Evaluating Sparse Autoencoders for Controlling Open-Ended Text Generation

Aleksandar Makelov*
Guide Labs
amakelov@guidelabs.ai

Nathaniel Monson
Guide Labs
nathaniel.monson@guidelabs.ai

Julius Adebayo
Guide Labs
julius@guidelabs.ai

Abstract

Sparse autoencoders (SAEs) have emerged as a promising method for disentangling large language model (LLM) activations into human-interpretable features. However, evaluating SAEs remains challenging, as current metrics like reconstruction loss and sparsity provide limited insight into whether the extracted features are meaningful and useful.

We propose a novel evaluation methodology that measures SAEs' effectiveness at controlling open-ended text generation. Using the Tiny Stories dataset and models as a minimal yet realistic testbed, we develop an automated pipeline to extract steering vectors for concepts in an unsupervised way. We then evaluate how well these vectors can control text generation compared to SAE latents.

Our results show that individual SAE latents can often improve upon the Pareto front between steering success and generation coherence compared to supervised steering vectors. This suggests that SAEs can learn meaningful, disentangled features useful for model control, providing evidence for their effectiveness beyond standard reconstruction metrics.

1 Introduction

Despite significant progress in mechanistic interpretability, we still lack precise understanding of how large language models (LLMs) represent and process information internally. Sparse autoencoders (SAEs) have emerged as a promising unsupervised approach for disentangling LLM activations into human-interpretable 'features' through overcomplete sparse dictionary learning [Olshausen and Field, 1997, Yun et al., 2021, Bricken et al.]. While significant effort is being devoted to scaling and improving SAEs [Templeton et al., 2024, Gao et al., 2024, Rajamanoharan et al., 2024], meaningful evaluation beyond aggregate metrics like reconstruction loss and sparsity remains challenging. Only a handful of recent works have begun addressing this gap in restricted settings [Conmy and Nanda, 2024, Karvonen et al., 2024, Huang et al., 2024, Chaudhary and Geiger, 2024, Makelov, 2024, Makelov et al., 2024]. A key open question is whether SAEs succeed at their intended purpose: extracting atomic, independent, and causally-relevant interpretable features from LLM activations.

This paper focuses on one specific aspect of this problem: how effectively can SAE latents control model generation compared to supervised steering vectors [Turner et al., 2023, Rimsky et al., 2023]? Building on recent work [Conmy and Nanda, 2024], we develop an automated methodology for computing and evaluating steering vectors for single-word concepts in texts. We implement our

*Correspondence to: amakelov@guidelabs.ai

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

evaluation in the Tiny Stories setting [Eldan and Li, 2023], which provides a controlled yet realistic testbed.

Our main contributions are:

- **steering vector generation and evaluation methodology**: we propose and validate a principled pipeline to generate and evaluate steering vectors for open-ended text generation. By focusing on single-word concepts, we can automatically measure steering success through word statistics;
- **comparison to SAEs**: we compare steering vectors to SAE latents trained on the Tiny Stories model and find that individual SAE latents can often provide Pareto improvements over steering vectors in the trade-off between success and coherence.

2 Related Work

Unsupervised Sparse Autoencoder Evaluations. SAEs were initially developed for unsupervised feature discovery in neuroscience [Olshausen and Field, 1996, 1997] and speech recognition [Deng et al., 2013]. The sparse coding and compressed sensing literature [Zhang et al., 2015, Donoho, 2006, Candès et al., 2006] provides theoretical guarantees for when such unsupervised feature discovery succeeds. However, these results may not transfer directly to the modern setting where SAEs are applied to LLM representations [Yun et al., 2021, Bricken et al.].

Traditional SAE evaluations rely on aggregate metrics like reconstruction loss, sparsity, and fraction of variance explained [Gao et al., 2024, Braun et al., 2024]. While useful as basic sanity checks, these metrics provide limited insight into whether SAEs successfully disentangle model representations or capture causally relevant features. Recent work has begun addressing this gap by using steering as an evaluation tool. Conmy and Nanda [2024] compared supervised steering vectors to their SAE-processed versions in GPT-2 XL. Makelov et al. [2024] and Makelov [2024] evaluated SAE feature control on the indirect object identification task [Wang et al., 2022], finding that certain SAE variants can match supervised approaches. Gao et al. [2024] proposed evaluating feature reliability through ablation studies, though the connection between logit sparsity and steering effectiveness remains unclear.

Steering Vectors. Steering vectors were introduced by Turner et al. [2023] as a way to guide model generation by intervening on activation patterns at inference time. The original approach computed these vectors from contrastive prompt pairs (e.g., 'anger' vs 'calm'). Subsequent work [Rimsky et al., 2023, Zou et al., 2023] refined this methodology but typically relied on instruction-following capabilities. Steering vectors are generally evaluated on two axes: control effectiveness and coherence preservation. Our work adopts this evaluation framework while developing new automated metrics for the Tiny Stories setting.

Controlled Settings for Interpretability. Evaluating interpretability tools like SAEs is challenging precisely because they are most useful when little is known about the model. Previous work has approached this by using controlled tasks where ground truth is available, such as indirect object identification [Wang et al., 2022, Makelov, 2024], the RAVEL benchmark [Huang et al., 2024], Othello [Li et al., 2022], and NYC street maps [Vafa et al., 2024]. We use Tiny Stories [Eldan and Li, 2023] as our controlled setting because its data generation process is explicit and simple enough that key features can be enumerated, while still maintaining realistic language modeling challenges.

3 Methods

We develop a systematic methodology to evaluate SAE features by comparing their effectiveness at controlling text generation to supervised steering vectors. Our approach consists of three components: (1) computing steering vectors for single words, (2) applying these vectors to control text generation, and (3) evaluating the success and coherence of the generated text.

Unless otherwise specified, we refer to the Tiny Stories dataset and 33M-parameter model (4 transformer layers) from Eldan and Li [2023] (*not* the instruction-following version). We use standard

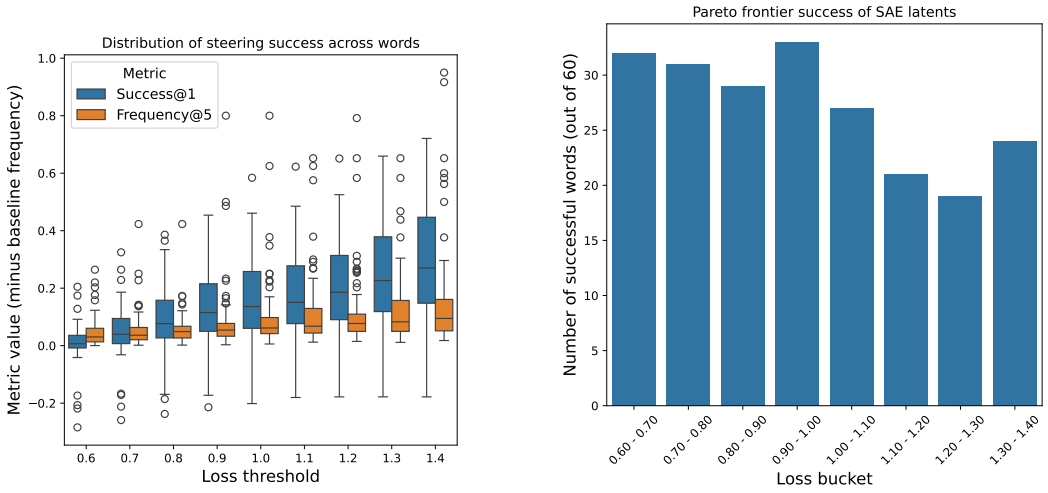

Figure 1: Summary of our main results. **Left**: trade-offs between steering success (y-axis) and generation coherence (measured by the loss of the model, x-axis) for our steering vectors. The success@1 metric measures the proportion of generations where the word we are steering for was successfully introduced in the text. The frequency@5 metric measures the proportion of the top 5 related words to the steering word that appear in the generated text. **Right**: over a set of 60 steering words and a range of loss buckets that correspond to coherent generations, we measure the number of words for which steering with SAE latents offers a Pareto improvement over steering vectors; see Appendix Figure **??** for individual examples.

terminology from Elhage et al. [2021] for decoder-only autoregressive transformers, notably the concept of the *residual stream*.

3.1 Computing Steering Vectors for Nouns

Brief motivation. Our goal is to compute steering vectors that make the model more likely to coherently include a specific noun (e.g. 'rain') and/or related concepts in its generation. While prior works have proposed various approaches for constructing such vectors, we develop our own methodology motivated by two key constraints: (1) the Tiny Stories model can generate simple coherent text but cannot follow general instructions; (2) we want to compute steering vectors for any *noun* word in the vocabulary without supervision. See Appendix A.1 for details on how we arrived at this methodology.

Steering vector computation. These constraints motivate the following procedure:

- given a noun word w from the vocabulary and a layer l of the model, we gather 200 sentences from the Tiny Stories training dataset which contain w (as indicated by the part-of-speech tagger of NLTK [Bird et al., 2009]);
- for each such sentence, compute the residual stream activation at layer l of the last token of the word as it appears in the sentence, using the Tiny Stories model;
- **average** these representations to obtain a single representation $\mathbf{z}_w$ for the word w;
- **subtract** the mean activation μ_l of the residual stream at layer l over the Tiny Stories training dataset from $\mathbf{z}_w$ to obtain the steering vector $\mathbf{v}_w = \mathbf{z}_w - \mu_l$. See Appendix A.1 on why we subtract the mean.

Note that $\mathbf{v}_w \in \mathbb{R}^d$ where d is the embedding dimension of the model. Furthermore, $\mathbf{v}_w$ comes with a 'canonical' magnitude, which reflects the average contribution of the word w to the residual stream at layer l over the Tiny Stories training dataset. This magnitude is a natural reference point when picking a coefficient α to scale $\mathbf{v}_w$ in order to steer the model towards generating text related to w.

We experimented with other methods to generate steering vectors, such as PaCE [Luo et al., 2024], but found them to be less effective in our setting.

What should we expect from these vectors? Finally, it is important to note that a priori there is no reason to expect that $\mathbf{v}_w$ will lead to the kind of steering we desire, and it is unclear if significantly better ways of computing steering vectors exist. We are simply using a heuristic that is motivated by prior work and the constraints of our setting; in particular, there are several reasonable and closely related hypotheses about what steering with $\mathbf{v}_w$ may do to the model:

- increase the likelihood of text **containing** w;
- increase the likelihood of text **related to** w;
- increase the likelihood of text that typically **follows** w;

among others.

3.2 Applying Steering Vectors to Control Text Generation

Method. Given a steering vector $\mathbf{v}_w$ at layer l and a coefficient α, we pick the first $N_{stories} = 500$ stories from the Tiny Stories validation set, and furthermore truncate each story to the first $N_{tokens} = 100$ tokens.

Given a truncated story S, we intervene on the forward pass of the model by adding $\alpha\mathbf{v}_w$ to the residual stream activation at layer l and token position $T_{steering} = 25$ (zero-indexed) in each story. The tokens up to and including $T_{steering} - 1$ are kept from S, while the tokens from $T_{steering}$ onwards are generated by the model (at temperature 0).

Motivation. The N_{tokens} and $T_{steering}$ values are chosen to balance two objectives: (1) ensuring sufficient context diversity to evaluate steering across varied settings [Tan et al., 2024], and (2) including enough generated tokens (in our case, $N_{tokens} - T_{steering} = 75$) to evaluate steering success and coherence over a reasonable context window.

We note that, when evaluating steering vectors for the word w, our approach requires computation proportional to the inverse of the frequency of w in the Tiny Stories distribution. The number $N_{stories}$ was chosen to ensure that in a sample of $N_{stories}$, each of the words we evaluate will appear at least ~ 20 times in expectation in the truncated stories, in order to ensure meaningful statistics.

3.3 Evaluating Steering Vectors: Success and Coherence Metrics

Once we have a candidate steering vector $\mathbf{v}_w$ for a noun word w, we need robust metrics to evaluate its effectiveness at steering generation. Following prior work [Turner et al., 2023, Conmy and Nanda, 2024, Rimsky et al., 2023], we focus on two key aspects:

- **generation success**: does w and/or its related concepts appear more frequently in the generated text when steering with $\mathbf{v}_w$?
- **generation coherence**: is the generated text grammatical and semantically meaningful, or does the model produce nonsensical or degenerate texts?

The details of how we implement these metrics in our specific setting are given below. In all cases, we assume we are given an original truncated story S spanning N_{tokens}, a steered story S', and a word w for which we want to evaluate steering success.

Generation success implementation. A straightforward way to measure generation success is to start with a truncated story S which *does not* contain w, and check if w appears in the generated text after steering with $\mathbf{v}_w$. A related metric is to check for the introduction of w-related words in an analogous way. Finally, we can also evaluate the frequency of w-related words in the generated text and compare it to the frequency of w-related words in the original truncated story S. This motivates the following metrics:

- **success@1**: the fraction of generations for which w appeared in the generated story, but not anywhere in the original truncated story S. The second part is important, because in some cases the prefix of S that we use to generate from may already contain w or lead up to w in a way that makes it more likely for w to appear in the generated text.
- **success@k**: same as success@1, but for the top k words most related to w (via TF-IDF) in the generated story. As we discussed previously in 3.1, it is unclear what steering with

$\mathbf{v}_w$ should do to the model, and this metric helps us understand the effect of steering on a broader set of concepts related to w. Note that there is some trade-off between capturing enough related words vs considering too many words as related; we chose 10 as a reasonable compromise. See Appendix A.3 for details.

- **frequency@k**: the fraction of the top k words most related to w in the generated story. We typically look at the difference between the value of this metric on the steered story and the value of this metric on the original story S to understand the effect of steering.

Generation coherence implementation. We use two complementary approaches to measure coherence:

1. The loss of the model on generated text: we compute the per-token cross-entropy loss of the same Tiny Stories model on the steered story S', and take the average loss over the generated tokens. See Appendix A.2 for an evaluation of this metric's validity.

2. Expert evaluation: we use Claude 3.5 to judge story quality, providing three subscores on a 1-9 scale for grammaticality, semantic coherence, and overall quality. On a small checked sample, human evaluators' blinded ratings aligned well with these automated scores. See Appendix A.6 for methodology details.

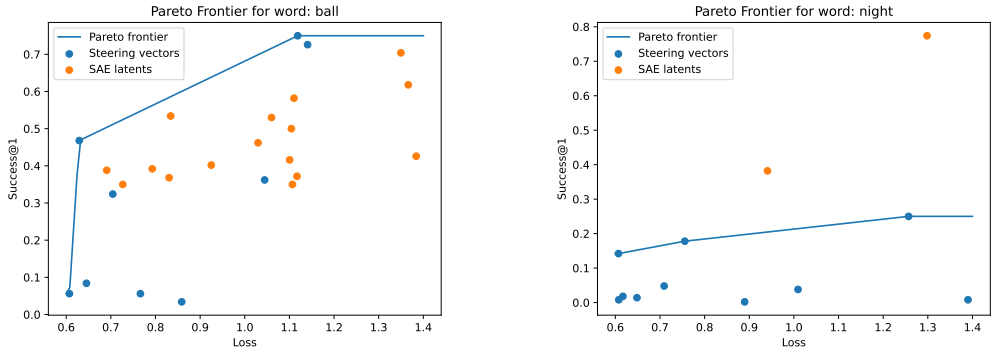

Figure 2: Example Pareto frontiers comparing steering vectors and SAE latents for two words: 'ball' (left) and 'night' (right). Each point represents a generation, with the x-axis showing model loss (lower is better) and y-axis showing steering success rate (higher is better). Points closer to the top-left corner indicate better performance. In blue, we draw the Pareto frontier for steering vectors; this allows us to see where SAE latents dominate steering vectors.

4 Results

4.1 Steering Vector Hyperparameters and Evaluation

We evaluated steering vectors for the top 100 most common nouns in the Tiny Stories training set (see Appendix A.3 for details). For each word, we computed steering vectors across all four layers of the model using coefficients $\alpha \in (0.5, 0.75, 1.0)$; we found that $\alpha > 1$ consistently led to degenerate text generation.

To evaluate overall steering success, we consider a generation successful if: (1) the loss is below a threshold t that we vary over a reasonable range, and (2) it satisfies success@1, meaning the steering word appears in the steered text but not in the original text. For each word w and loss threshold t, we compute success rates across all strength-layer combinations and take the maximum to identify the best configuration. We then subtract the base rate for this word to account for its natural frequency in the dataset. We apply the same methodology to summarize the frequency@5 metric, subtracting the base rate of the top 5 related words in the unsteered text[2].

[2]Note that, after our preprocessing (Appendix A.3) the generated text has on average ~ 25 words, so while the frequency numbers may seem low, they are over a very small amount of text; a frequency of 0.1 corresponds to about 2 related words.

We chose $t \in [0.6, 1.4]$ as our range of loss thresholds, corresponding to one standard deviation around the average loss value of ~ 0.98 over the corresponding range of tokens in the Tiny Stories validation set (see Appendix A.2 for justification).

We plot the distributions of these metrics in Figure 1 (adjusted for the number of generated tokens). Notably, we observe that steering sometimes *decreases* the success rate compared to random, consistent with findings from Tan et al. [2024]. While we also analyzed the success@5 metric similarly, we found high variation across steering words, with many words showing negative effects from steering.

4.2 Comparison to SAE Latents

We evaluated a set of 4 TopK SAEs [Gao et al., 2024] (one per model layer) adapted from EleutherAI [2024]. Each SAE had $d_{hidden} = 8192$ hidden latents and $k = 32$ active latents per activation (see Appendix A.4 for training details). Due to computational constraints, we evaluated 200 randomly chosen features from each SAE.

Our evaluation methodology for SAE latents parallels that of steering vectors, with two key adaptations:

- Without a canonical way to assign words to SAE latents, we consider success@1 for all vocabulary words and take the top 10 words as potential labels for each latent
- For steering strength, we scale each unit-vector SAE latent by its maximum activation over the validation set, then multiply by $\alpha \in (0.5, 0.625, 0.75, 0.875, 1.0)$

This process identified SAE latents corresponding to 60 of our 100 steering words. For comparison, we computed Pareto frontiers between success@1 and loss across all configurations for both methods over the loss range $[0.6, 1.4]$.

Remarkably, for 54 out of 60 words, SAE latents improved upon the steering vectors' Pareto front. Figure 2 shows one success and one failure, while Figure 3 in the appendix shows additional successes and failures. The improvements appear well-distributed across loss values, suggesting SAE success is not limited to particular coherence regimes (Figure 1, right).

5 Discussion, Limitations, and Future Work

Our results suggest that SAEs can learn features that are not just interpretable, but also useful for controlled text generation. However, several important limitations and open questions remain:

Limitations of Current Work

- Our steering vector methodology, while effective, may not be optimal. It remains unclear whether better approaches exist for computing steering vectors in our setting.
- The Tiny Stories model's limited capabilities - particularly its inability to follow instructions - restrict our ability to compare with more sophisticated steering approaches from prior work.
- Due to computational constraints, we only evaluated a small subset of SAE latents. A more comprehensive evaluation might reveal different patterns.

Future Directions

- Scaling this evaluation methodology to larger language models would help validate whether our findings generalize beyond the minimal Tiny Stories setting.
- Investigating whether multiple SAE latents can be combined effectively for steering could reveal additional capabilities.
- Developing automated methods to map SAE latents to interpretable concepts remains an important challenge for making SAE-based steering practical.

Conclusions. Our work provides evidence that SAE latents can effectively control text generation, sometimes outperforming supervised steering vectors. This suggests that SAEs do more than just compress model activations - they can extract meaningful, causally relevant features. While our evaluation is limited to a simple setting, it provides a foundation for more comprehensive investigations of SAE capabilities in model control and interpretability.

References

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. *arXiv preprint arXiv:2405.12241*, 2024.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning, 2023. *URL https://transformer-circuits. pub/2023/monosemantic-features/index. html.→ p*, 9.

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

Maheep Chaudhary and Atticus Geiger. Evaluating open-source sparse autoencoders on disentangling factual knowledge in gpt-2 small. *arXiv preprint arXiv:2409.04478*, 2024.

Arthur Conmy and Neel Nanda. Activation steering with SAEs. *Alignment Forum*, 2024. URL https://www.alignmentforum.org/posts/C5KAZQib3bzzpeyrg/progress-update-1. Progress Update #1 from the GDM Mech Interp Team.

Jun Deng, Zixing Zhang, Erik Marchi, and Björn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 humaine association conference on affective computing and intelligent interaction*, pages 511–516. IEEE, 2013.

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

EleutherAI. Sparse autoencoders. https://github.com/EleutherAI/sae, 2024. Accessed: 2024-09-25.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Jing Huang, Zhengxuan Wu, Christopher Potts, Mor Geva, and Atticus Geiger. Ravel: Evaluating interpretability methods on disentangling language model representations. *arXiv preprint arXiv:2402.17700*, 2024.

Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. *arXiv preprint arXiv:2408.00113*, 2024.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Jinqi Luo, Tianjiao Ding, Kwan Ho Ryan Chan, Darshan Thaker, Aditya Chattopadhyay, Chris Callison-Burch, and René Vidal. Pace: Parsimonious concept engineering for large language models. *arXiv preprint arXiv:2406.04331*, 2024.

Aleksandar Makelov. Sparse autoencoders match supervised features for model steering on the ioi task. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

Aleksandar Makelov, George Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. *arXiv preprint arXiv:2405.08366*, 2024.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

Daniel Chee Hian Tan, David Chanin, Aengus Lynch, Adrià Garriga-Alonso, Dimitrios Kanoulas, Brooks Paige, and Robert Kirk. Analyzing the generalization and reliability of steering vectors. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

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

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Keyon Vafa, Justin Y Chen, Jon Kleinberg, Sendhil Mullainathan, and Ashesh Rambachan. Evaluating the world model implicit in a generative model. *arXiv preprint arXiv:2406.03689*, 2024.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Zeyu Yun, Yubei Chen, Bruno A Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. *arXiv preprint arXiv:2103.15949*, 2021.

Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. A survey of sparse representation: algorithms and applications. *IEEE access*, 3:490–530, 2015.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Appendix / supplemental material

A.1 Motivation and Details for Steering Vector Generation

Motivation. We are working with a very small model (33M parameters and 4 transformer layers) which is barely able to generate coherent text, and is not good at general instruction-following tasks

(note that Eldan and Li [2023] also trained instruction-based versions of their models, but they can only respond to very specific instructions about generating stories). This rules out the relatively more sophisticated approach of Zou et al. [2023], which requires a model that can follow instructions in order to extract concept vectors. Similarly, it rules out the multiple-choice question approach for constructing contrastive pairs from Rimsky et al. [2023].

Furthermore, we want to compute steering vectors for any *noun* word in our vocabulary. While sometimes it is straightforward to construct antonyms of noun words (such as the 'anger-calm' pairing from Turner et al. [2023]), this approach fails for most nouns, e.g. 'tree' or 'hanger' do not have clear antonyms.

Why do we subtract the mean activation? Subtracting the mean is crucial to ensure that adding $\mathbf{v}_w$ to an existing residual stream activation (which is how steering vectors are used) does not take the activation out of distribution by 'doubling' the bias in the residual stream, which can often have significant magnitude.

As another point of view, we can think of $\mathbf{z}_w$ and μ_l as a contrastive pair of activations, where $\mathbf{z}_w$ represents w and μ_l approximately represents 'not w' (i.e, everything else).

Alternative approaches. We considered several alternative approaches for computing steering vectors, but found them to be perform poorly in our early experiments.

A.2 Details on using Loss as a Proxy for Coherence

By manually inspecting generations in buckets of loss values of width 0.2 spanning the range $[0.0, 4.0]$ (10 generations per bucket), we find that generations with loss ≤ 1.0 are generally reasonable, but around a loss of ~ 1.2, the generations start to increasingly suffer from poor semantic and grammatical coherence, and, eventually, from increasingly difficult to recover from bursts of mode collapse. This agrees well with the observation that the corresponding average loss over the validation set is $\sim 0.98 \pm 0.4$ (standard deviation) over the original set of stories we are working with. We further observe that coherence generally correlates with the loss value, which increases our confidence in using the loss as a proxy for coherence.

A.3 Details on the Vocabulary Used for Tiny Stories

TD-IDF details. We determine the related words by running TF-IDF [Schütze et al., 2008] on the Tiny Stories training set, and taking the top 10 words according to cosine similarity to the word embedding of w;

Computing a vocabulary. Throughout this work, we use a shared vocabulary for the Tiny Stories dataset consisting of 5,676 words derived from the most common words in the training set.

Text preprocessing used throughout this work. In order to obtain more meaningful word statistics, we use the following standard text preprocessing pipeline for both original and generated stories:

- we make the text lowercase;
- we tokenize it into words using the `nltk.tokenize.word_tokenize` function;
- we remove stop words using the NLTK stop words list for English;
- we remove non-alphabetic characters (which do occur with some frequency in the Tiny Stories dataset);
- we lemmatize the words using the NLTK WordNet lemmatizer.

Computing the top 100 most common nouns.

For completeness, here is the full list of 100 nouns that we picked:

```
 mom, girl, day, time, bird, man, fun, mommy, dog, toy, dad,
park, love, car, friend, tree, home, ball, cat, water, share,
lot, way, house, family, room, rabbit, goodbye, bit, fish,
bunny, sky, door, mum, story, truck, garden, noise, food,
place, boat, voice, thing, ice, cake, dress, wind, book,
brother, rock, picture, baby, doll, work, head, flower, frog,
```

```
end, mother, forest, paper, bed, hole, lion, world, game,
ground, squirrel, bag, duck, swing, kitchen, surprise, hat,
grass, rain, daddy, farmer, pond, lesson, owner, rest, window,
piece, adventure, night, year, monster, child, hair, cream,
purple, bug, moment, spot, magic, grandma, teacher, bite,
animal
```

A.4 SAE training details and metrics

We train our SAEs on the first 2×10^6 stories from the Tiny Stories training set, comprising a total of about 250×10^6 tokens. Our SAEs achieve a recovered cross-entropy loss of $\sim 0.96+$ against a mean ablation of the corresponding residual stream, and a fraction of variance unexplained in activations that ranges from ~ 0.1 to ~ 0.27, increasing over the 4 layers of the model.

A.5 Selected generations from steering vectors

```
Loss: 1.81
Word: rain, Strength: 0.75, Layer: 2
Once upon a time, there was a little girl named Lily. She loved to eat oatmeal
every morning for breakfast. One day, she woke up and poured herself a bowl of
oatmeal. She was so happy because she was so lucky to have such a yummy treat.
One day, she poured herself a whole bowl of cereal and poured herself a whole
bowl. She poured and poured until the cereal was all gone. When she stopped, she
poured herself even more. Her clothes
```

A.6 Automated coherence evaluation methodology

Ideally, we would have a ground truth to how coherent or otherwise desirable a story is. Lacking this, we might hope that several imperfect measures correlate sufficiently to be useful. One such imperfect measure is the judgement of a powerful LLM. After some exploration, we note the following observations[3]:

1. When asked to grade a story on a scale from 1-5 or 1-10, LLMs are very unlikely to use the full range of grades available to them (plausibly reflecting the pretraining distribution—human reviewers also have a tendency to use only a subset of the available scoring range).
2. If asked to grade a batch of 20 stories with a single prompt with scores of 1-5, the model displays order effects, with the first 2 stories getting significantly higher scores than they would receive in other positions.
3. Holding the words of a grading-request nearly constant, the same story can receive many grades. Using variants (on the level of "adding an extra blank line before beginning the story" or "enclosing the story in quotes or not") of the grading-request prompt used in TinyStories, the same story can receive grades of "87/100", "7.6/10", or "A-". Similarly, a story introduced with:

   "The exercise tests the student's language abilities and creativity.
   41) """Once upon a time,"

and the same story introduced with:

   "The exercise tests the student's language abilities and creativity.
   4) """Once upon a time,"

can receive repetition and creativity subscores of "8, 7" for the first prompt and "B, B+" for the second.

[3] We do not claim to have more than anecdotal evidence for these points.

4. Allowing a model to give textual descriptions before numerical scores can somewhat increase the stability of the scores, at the cost of more time and money. We did not find the tradeoff worth it for the present work.
5. Evaluation of this can be exceptionally frustrating as neither Anthropic nor OpenAI's APIs give deterministic answers for a fixed prompt, even with 0 temperature.

Note that effects 1-3 (and plausibly 4) are also likely to be true of *human* scorers as well. Comparing the stability or consistency of human scoring with automated LLM scoring is beyond the scope of this work. For now, we merely note that these issues do not invalidate LLM scoring as a useful measurement of coherence.

After some exploration, we used a prompt of the following form:

> "Evaluate this story for kindergarteners. Assign scores between 17 (poor) and 25 (excellent). Use the full range of scores for each of the following metrics: Flow (the story should flow naturally), Repetition (the story should not repeat words or phrases unnecessarily), Self-Consistency (the story should not contradict itself). Please be blunt and objective. Only include the scores; no text analysis.
>
> `<begin_stories>`
> " + {actual stories} + "
> `<end_stories>`
> `<begin_ratings>`
> `<a,b,c>`
> `<d,e,f>`
> `<end_ratings>`"

With this prompt, we found sufficient score diversity to be useful—on a small sample, the most frequently used score (24) was used under 25% of the time. In contrast, one of our earlier attempts with a 1-5 scoring system returned a score of 4 approximately 80% of the time.

The model also displayed reasonably high self-consistency. Running a set of stories through in several different orders demonstrated a sample standard deviation of ~ 1.5 for each story's subscores, which is low enough that we can have over 90% confidence a given run is within 1 point of the "true score." It also did not show major order effects.

A.7 Pareto fronts for steering vectors and SAE latents

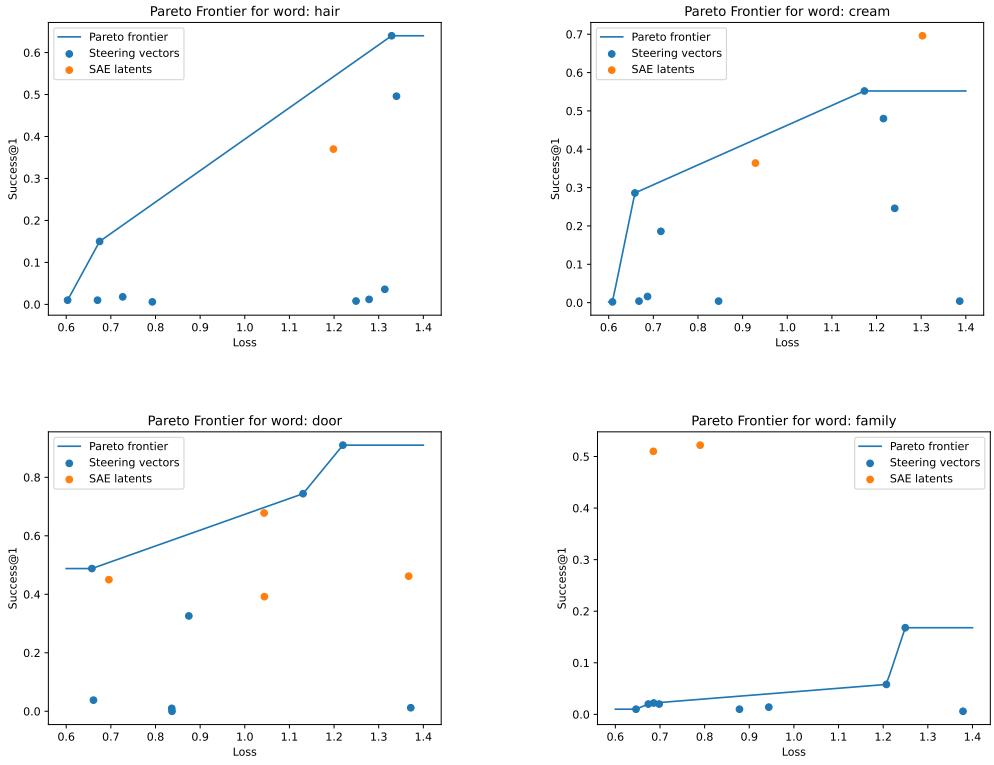

Figure 3: Additional examples of Pareto frontiers comparing steering vectors and SAE latents for four more words.