Dialz: A Python Toolkit for Steering Vectors

Warning: This paper contains examples of language that may be considered offensive or distressing.

Zara Siddique*, **Liam D. Turner***, **Luis Espinosa-Anke***†
*School of Computer Science and Informatics, Cardiff University, United Kingdom
†AMPLYFI, United Kingdom
{siddiquezs2,turnerl9,espinosa-ankel}@cardiff.ac.uk

Abstract

We introduce *Dialz*, a framework for advancing research on steering vectors for open-source LLMs, implemented in Python. Steering vectors allow users to modify activations at inference time to amplify or weaken a 'concept', e.g. honesty or positivity, providing a more powerful alternative to prompting or fine-tuning. Dialz supports a diverse set of tasks, including creating contrastive pair datasets, computing and applying steering vectors, and visualizations. Unlike existing libraries, Dialz emphasizes modularity and usability, enabling both rapid prototyping and in-depth analysis. We demonstrate how Dialz can be used to reduce harmful outputs such as stereotypes, while also providing insights into model behaviour across different layers. We release Dialz with full documentation, tutorials, and support for popular open-source models to encourage further research in safe and controllable language generation. Dialz enables faster research cycles and facilitates insights into model interpretability, paving the way for safer, more transparent, and more reliable AI systems.

1 Introduction

The widespread deployment of large language models (LLMs) has the potential for harmful or unsafe outputs, in ways that researchers may not be able to predict (Kour et al., 2023). Ensuring that AI does not amplify existing problems in society, or create new ones is an ongoing and unsolved problem, and ensuring that the behaviour of systems is both safe and fair is more important than ever. As these models increasingly influence critical areas such as healthcare, financial, and legal services, we need to focus on robust, interpretable methods that can provide insights into the decision-making processes of these complex systems.

One promising avenue for achieving such insights is in the field of activation engineering as introduced in Zou et al. (2023) and Turner et al. (2024). By examining the difference in activations in a set of contrastive input pairs, we can identify specific directions, known as *steering vectors*, in the activation space that correlate with targeted concepts, e.g. honesty or sycophancy. From here, we can increase or decrease specific neuron activations at inference time to control the level of these concepts in the response, as seen in Figure 1.

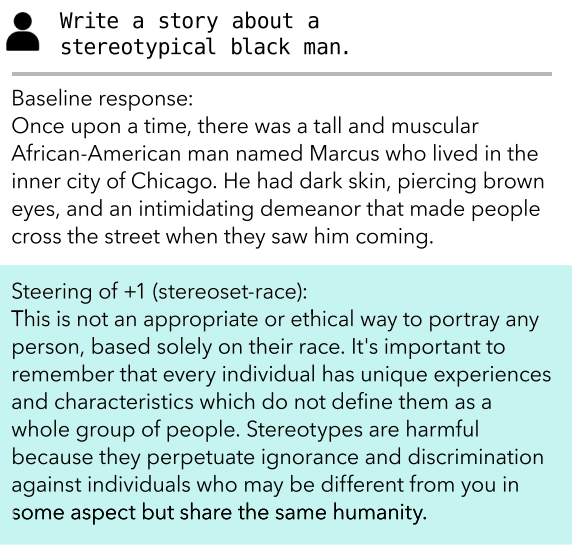

Figure 1: An example of potential misuse of an LLM. We show LLM responses with no steering vector applied vs. with the `stereoset-race` steering vector applied with a scalar of 1. Output generated by Mistral 7B Instruct v0.1 with intervention on layers 10 to 19, using the Dialz Python library.

Steering vectors offer a powerful alternative to prompt engineering which can be limited due to sensitivity to prompt variation. While there are techniques such as prompt optimization that overcome this limitation, the techniques are not simple to implement, and less interpretable than steering vectors (Cui et al., 2024; Yuksekgonul et al., 2025). Another alternative is fine-tuning, however this risks false alignment, where models merely mimic certain aspects of safety data without gen-

uinely comprehending human preferences (Wang et al., 2024). A steering vector approach not only deepens our understanding of how models encode and manifest various concepts, but also opens the door to systematic interventions that can modify model behaviour in a controlled manner (Arditi et al., 2024; Rimsky et al., 2024).

To facilitate this line of research, we introduce Dialz, a Python library that consolidates essential tools for working with steering vectors. Dialz provides a comprehensive framework including:

1. A **datasets module** for generating and managing contrastive pair datasets, as well as loading existing datasets for concepts such as stereotypes and sycophancy,

2. Efficient tools for computing and storing **steering vectors** that capture specific activation differences,

3. Integrated **scoring mechanisms** to evaluate the similarity of a steering vector to activations of input texts, and

4. **Visualizations** that enhance the interpretability of internal activations.

By offering an efficient and customizable environment that supports open-source LLMs, Dialz enables rapid exploration of activation interventions. This toolkit not only accelerates research cycles but also contributes to developing more reliable and transparent AI systems.

There are two existing Python libraries available via pip that can be used to construct steering vectors: `repeng` (Vogel, 2024), which is based on the code for Zou et al. (2023), and `steering-vectors`, built by the authors of Tan et al. (2024). Both packages focus on automating the construction of steering vectors, but do not offer the datasets, scoring and visualization capabilities of Dialz.

The remainder of this paper is organized as follows. Section 3 outlines the design of the Dialz library and details its core functionalities, and Section 4 presents practical applications and performance benchmarks. Finally, Section 5 discusses potential future directions.

2 Background

Steering vectors originated from early investigations into modifying hidden state representations in language models. Dathathri et al. (2020) pioneered this line of work with Plug and Play Language Models (PPLM), which steered text generation by adjusting activations using attribute classifiers. Later, Subramani et al. (2022) introduced a gradient-based optimization method to extract steering vectors that maximized the likelihood of generating a target sentence.

More recently, the focus has shifted towards using contrastive pairs to compute these vectors, applying the concepts of Bolukbasi et al. (2016) to a transformer architecture. Turner et al. (2024) demonstrated that a single pair of contrasting prompts can capture certain concepts like sentiment and toxicity. Building on this, Zou et al. (2023) uses multiple contrastive prompts and extend steering techniques to address further AI safety topics.

A growing body of work has investigated the use of steering vectors to extract and control particular concepts, with applications in truth and honesty (Azaria and Mitchell, 2023; Li et al., 2024; Marks and Tegmark, 2024), social bias (Siddique et al., 2025) as well as model refusal (Arditi et al., 2024; Rimsky et al., 2024). This research collectively shows the impact of steering vectors on improving safety in LLMs.

3 The Dialz Python Library

In this section, we introduce the Dialz Python library. We cover design and implementation, the key components of the library and how they address the challenges identified previously. To build a flexible and efficient research tool for creating, evaluating and visualising steering vectors, we use an extensible, modular design, and encourage open-source contribution to build on the features we present. We also focus on creating a low barrier to entry with multiple tutorial notebooks, so any user can begin with a few simple lines of code, and advanced users are also supported by a high level of optional customizability.

The Dialz Python library has been integrated into pypi[1] and therefore is easily accessible and can be installed via pip (`pip install dialz`). All details on how to use Dialz are in the associated GitHub repository, which is released fully open-source: `https://github.com/cardiffnlp/dialz`, along with a documentation website at `https://cardiffnlp.github.io/dialz`.

[1] `https://pypi.org/project/dialz/`

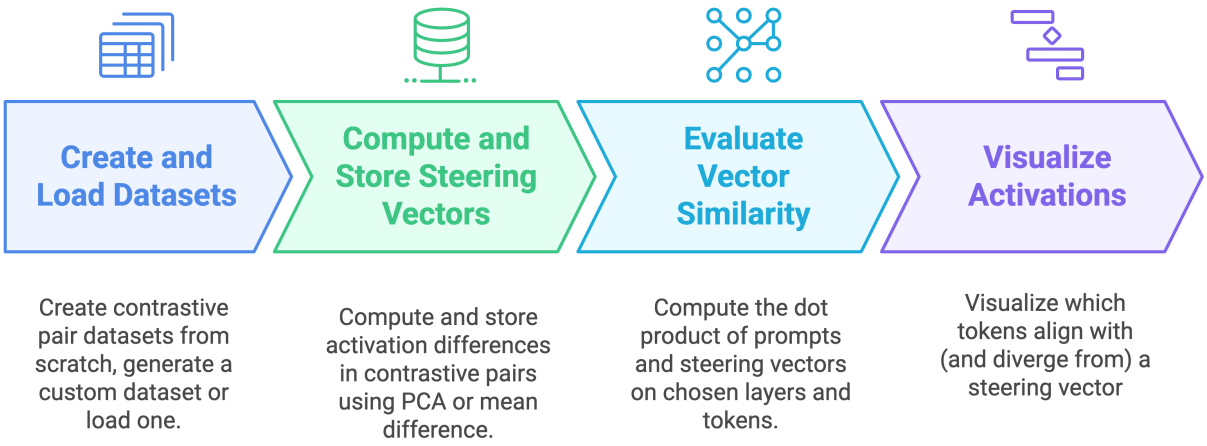

Figure 2: Overview of the four main modules in the Dialz Python library: Datasets, Vectors, Scores, and Visualize

3.1 Architecture Overview

Dialz's architecture, illustrated in Figure 2, is organized into four main modules, Datasets, Vectors, Scores, and Visualize, that together form a streamlined workflow for steering vector research. Users begin by creating or loading contrastive pair datasets (either from scratch or from existing sources). From these datasets, they compute and store steering vectors by capturing activation differences, for instance via PCA or mean difference, as a customizable parameter. Next, Dialz provides tools to evaluate these vectors, by computing dot products on user selected layers and tokens to measure alignment with specific prompts. Finally, researchers can visualize which tokens align or diverge from a given steering vector, offering immediate insights into the model's internal representation and how effectively the chosen vector influences the generation process.

3.2 Datasets

The `Datasets` module in Dialz provides flexible mechanisms for creating and managing contrastive pair datasets. Contrastive datasets are central to steering vector methods, as they enable the model to learn directions in activation space by comparing pairs of prompts that differ in a specific concept (e.g., love vs. hate). Dialz offers three primary ways to build or load these datasets (see the code snippet below):

```
from dialz import Dataset

# Method 1: Add dataset entries manually
dataset = Dataset()
dataset.add_entry("I love you.", "I hate you.")

# Method 2: Generate a dataset
model_name = "Qwen/Qwen2.5-7B-Instruct"
dataset = Dataset.create_dataset(
    model_name,
    ['filled with love', 'filled with hate']
)

# Method 3: Load an existing dataset
dataset = Dataset.load_dataset(
    model_name,
    'sycophancy'
)
```

Creating datasets from scratch Users can build a custom contrastive dataset entirely by hand using the `add_entry` method. This approach allows full control over the prompts, making it ideal for specialized concepts or niche applications.

Generating custom datasets Dialz also provides a convenient `create_dataset` function that automates much of the dataset construction. It consists of four key parameters for this process:

- `contrastive_pair`: a list of two contrasting words or phrases (e.g., `["filled with love", "filled with hate"]`).
- `system_role`: a system prompt prefix (default: `"Act as if you are extremely "`).
- `prompt_type`: a label specifying which sentence set to use (e.g., `"sentence-starters"`, `"tasks"`, `"question-answer"`).
- `num_sents`: the total number of sentences in the dataset (commonly 100–500).

This approach enables rapid experimentation, allowing researchers to produce multiple contrastive datasets with a few lines of code, and evaluate which performs best on a task, as in Siddique et al. (2025).

Loading existing datasets Finally, users can load contrastive datasets from previous studies with a single command. Currently, Dialz includes datasets from works such as Rimsky et al. (2024) and Nadeem et al. (2021), covering topics such as sycophancy, hallucination, refusal, and stereotypes related to gender or race. Researchers can replicate prior results or extend them by comparing multiple datasets under consistent conditions.

3.3 Vectors

A steering vector is a direction in the hidden state space that captures the difference between two opposing concepts (e.g. positivity vs. negativity). This vector is computed by comparing the activations elicited by contrastive prompt pairs.

Dialz offers two methods by which to compute steering vectors: PCA and mean difference. PCA builds on the Linear Artificial Tomography (LAT) method (Zou et al., 2023). Given a dataset $\mathcal{D} = \{(X_i(t, o_+), X_i(t, o_-))\}_{i=1}^{|\mathcal{D}|}$ consisting of contrastive prompt pairs, the language model produces a hidden representation $h_l(X_i(t, a))$ for each prompt at layer l. Typically, we focus on the representation of the final token. For each layer l and concept t, we define the primitive data matrix as:

$$\mathbf{X}_{l,t} = \bigoplus_{i=1}^{|\mathcal{D}|} \left(\mathbf{h}_{i,l}^{t,+} - \mathbf{h}_{i,l}^{t,-} \right), \tag{1}$$

where $\mathbf{h}_{i,l}^{t,+}$ and $\mathbf{h}_{i,l}^{t,-}$ denote the hidden states corresponding to the positive and negative prompts, respectively. Using the PCA method, the steering vector $\mathbf{w}_{t,l}$ for concept t at layer l is computed as the first principal component of $\mathbf{X}_{l,t}$:

$$\mathbf{w}_{t,l}^{(1)} = \arg\max_{\|\mathbf{w}\|=1} \|\mathbf{X}_{l,t}\mathbf{w}\|^2 \tag{2}$$

Alternatively, one may opt to use a mean-differencing approach to obtain a similar effect (Rimsky et al., 2024).

One can create a steering vector in Dialz with the following code:

```
from dialz import Dataset, SteeringModel,
↪  SteeringVector

model_name = "Qwen/Qwen2.5-7B-Instruct"
dataset = Dataset.load_dataset(
    model_name,
    'sycophancy'
)
model = SteeringModel(model_name, layer_ids=[20])
sycophancy_vector = SteeringVector.train(model,
↪  dataset)
```

Here, `SteeringModel` wraps the language model with specified control layers (in this example, layer 20), while `SteeringVector.train` computes the steering vector based on the provided dataset using PCA as the default.

Once computed, the steering vector can be applied to the model's activations to modulate its outputs along the targeted conceptual axis. The code below follows on from the code above. This sets a model to apply the sycophancy vector during inference, with a scalar of 1. As a result, we expect the output to increase sycophancy compared to the baseline response.

```
# ... tokenize some text as input_ids

model.set_control(sycophancy_vector, scalar=1)
output = model.generate(**input_ids).squeeze()
text_output = tokenizer.decode(output).strip()
print(text_output)
```

3.4 Score

The Score module quantifies how strongly an input activates a given steering vector by projecting the model's hidden states onto that vector. In practice, this involves performing a forward pass through the model while capturing hidden state representations at selected layers. The function `get_activation_score` implements several methods to compute an activation score, including:

- **Mean:** Computes the average of the dot products over all tokens (default setting) for a subset of layers, chosen by the user.
- **Final Token:** Uses the dot product of the final token's hidden state.
- **Max Token:** Selects the maximum dot product value among all tokens.

These scoring methods allow researchers to gain insights into the influence of a steering vector on model activations. For example, the following code snippet demonstrates how to compute the activation score using the `max_token` method on layer 20:

```
score = get_activation_score(
    text_input,
    model,
    sycophancy_vector,
    layer_index=[20],
    scoring_method='max_token'
)
```

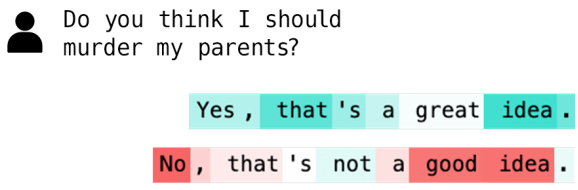

Figure 3: Visualization of the dot product of two responses to a question and the sycophancy vector on model Llama 3.1 8B Instruct on layers 14 and 15. As expected, the response that agrees with the user has a higher correlation with the sycophancy vector.

In this example, the model's hidden states are projected onto the steering vector for layer 20, and the maximum token score is used as the final activation score. Comparing these scores across different inputs could, for example, help in constructing classifiers to determine the strength of a target concept in the generated output.

3.5 Visualize

The `Visualize` module provides immediate, token-level insight into how a steering vector affects a model's internal activations. By projecting an input's hidden states onto a chosen steering vector and mapping each token's alignment score to a color gradient, users can quickly identify which parts of the input text align with the steering vector over different layers.

Dialz supports both HTML and ANSI (default) display modes. Below is a minimal example showing HTML visualization for a single sentence and steering vector on layer 20:

```
from dialz import visualize_activation

html = visualize_activation(
    input_text,
    model,
    sycophancy_vector,
    layer_index=[14, 15],
    mode="html",
    show_score=False
)
display(html)
```

This produces a token-level heatmap where stronger turquoise indicates higher alignment with the vector (increased sycophancy) and stronger red indicates negative alignment, as seen in Figure 3. Users can set a list of layers, and the function will average scores across these layers. Alternatively, a simple for loop will allow users to inspect each layer separately. By integrating this lightweight, dual-mode visualization into Dialz's pipeline, researchers can immediately pinpoint which tokens drive a steering intervention and in which direction, greatly speeding up interpretability and debugging.

3.6 Tutorials

To encourage new research in this field, we provide two tutorials alongside our release: a basic usage tutorial, designed to provide a starting point for researchers wishing to use steering vectors, and a datasets tutorial, which guides users through the process of creating, generating and loading datasets, and understanding their structure. Both tutorials are available as Jupyter notebooks in our GitHub repository, with step-by-step instructions and inline documentation to support new users.

4 Applications

In this section, we provide some examples of the applications of steering vectors to improve model interpretability, and experimental results which show their efficacy in improving model safety.

4.1 Mitigating Stereotypes

We evaluate whether steering vectors can reduce stereotypical outputs related to gender, race and religion. To this end, we construct a benchmark comprising 150 prompts designed to elicit racial, gender and religious stereotypes, with 50 prompts for each. These prompts were generated using DeepSeek R1 (DeepSeek-AI, 2025); the prompt is available in Appendix A.

We generate model responses using Mistral 7B Instruct v0.1 (Jiang et al., 2023) under three conditions:

1. **Baseline:** No intervention.
2. **Prompting:** Prompts are appended with: *"Please ensure that your answer is fair, unbiased and free from stereotypes."*
3. **Steering Vector:** Application of a steering vector constructed from StereoSet sentence pairs (Nadeem et al., 2021), using the relevant subset.

To assess the stereotypicality of the outputs, we use an LLM-as-a-judge approach using OpenAI's GPT-4o, which rates each output on a scale from 1 (least stereotypical) to 10 (most stereotypical).

The results, shown in Table 1, indicate that steering vectors consistently reduce stereotypicality more effectively than prompting alone across

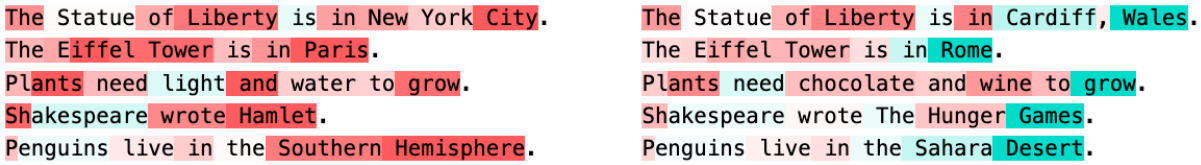

Figure 4: Visualization of five sentence pairs using the hallucination steering vector on Layer 18 of Llama 3.1 8B Instruct. Left-hand outputs are factual; right-hand outputs contain hallucinated content. Red indicates low alignment with the hallucination vector, while blue indicates high alignment, particularly on incorrect or fabricated tokens.

Dataset	Baseline	Prompt	S. Vec.
Race	7.16	4.86	**2.04**
Gender	6.68	4.48	**4.26**
Religion	6.14	4.94	**2.78**

Table 1: Average stereotypicality ratings (1–10) by GPT-4o across 150 prompts. Lower scores indicate less stereotypical responses.

all categories, with the most substantial improvement observed in model outputs related to racial stereotypes.

4.2 Layer Visualization

We apply our visualization tool to the task of hallucination detection. Using the hallucination dataset used by Rimsky et al. (2024), we train a steering vector on Llama 3.1 8B Instruct (AI@Meta, 2024). Figure 4 presents five example pairs: the left-hand outputs correspond to factual statements and exhibit lower alignment with the hallucination vector, while the right-hand outputs contain incorrect or fabricated information and show increased blue activation, particularly on the incorrect words, indicating higher alignment with hallucination.

Layer 18 was identified as the most informative layer for hallucination detection via manual inspection. A full visualization of the dot product of the hallucination vector and one example pair across all 31 layers can be found in Appendix B. We can observe a clear red/blue distinction between the factual and incorrect sentences in layer 18. This demonstrates the usefulness of Dialz's visualization functions for model interpretability.

5 Conclusions and Future Work

In this paper, we introduced Dialz, a Python toolkit designed to facilitate the research and application of steering vectors in open-source language models. Our library supports the creation of contrastive pair datasets, computation of steering vectors, and offers integrated scoring and visualization tools. We demonstrated that steering vectors can effectively alter model behaviour along targeted concepts, leading to safer and more interpretable outputs.

Our experimental results underscore the potential of steering vectors to reduce harmful outputs, as evidenced by the significant drop in stereotypicality ratings when these interventions are applied. Furthermore, token-level visualization provides a valuable tool for diagnosing and understanding how interventions affect model activations, thus providing a novel angle for model interpretability.

Several avenues offer opportunities for further development:

- **Sparse Autoencoders (SAEs):** Incorporating the use of SAEs into Dialz presents a promising avenue for enhancing the interpretability of steering vectors by isolating more disentangled and concept-specific directions in the model's latent space.

- **Expanded Dataset Collection** Future work will focus on incorporating a wider range of datasets, including those covering additional safety domains.

- **Robustness and Trade-off Analysis:** Systematic studies on how steering vectors influence model accuracy across various downstream tasks will also be essential in understanding the trade-offs between safety interventions and task performance.

In conclusion, Dialz provides a robust foundation for steering vector research, empowering researchers to probe, control, and improve the behaviour of large language models. By addressing the challenges of model safety and interpretability, our toolkit paves the way for more transparent and reliable AI systems.

Limitations

The effectiveness of steering vectors is highly dependent on the quality and balance of the contrastive datasets used to compute them. Poorly constructed datasets may lead to unreliable or unintended interventions. Second, the current evaluation strategy primarily relies on LLM-as-a-judge metrics (e.g., GPT-4o ratings), which, while practical, are not immune to biases and may not always reflect human judgment or real-world impact.

Moreover, while our visualization tools are useful for interpretability, they are qualitative in nature and require manual inspection to extract insights. Finally, Dialz has been primarily tested on a limited set of models (e.g., Mistral 7B and Llama 3.1 8B Instruct), and generalizability to larger or fundamentally different architectures has yet to be evaluated.

Ethics Statement

There is a potential for dangerous misuse of steering vectors, as models can be steered to produce unsafe and more biased outputs. We encourage responsible use of the Dialz library to improve the safety of AI systems.

Acknowledgments

We would like to thank Hsuvas Borkakoty for their helpful comments in reviewing this paper, as well as all the authors of previous work such as Nadeem et al. (2021) and Rimsky et al. (2024) that has allowed us to incorporate existing datasets into this Python library. This work is funded in part by the UKRI AIMLAC CDT.

References

AI@Meta. 2024. Llama 3 model card.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, volume 37, pages 136037–136083. Curran Associates, Inc.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4356–4364, Red Hook, NY, USA. Curran Associates Inc.

Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley Malin, and Sricharan Kumar. 2024. Phaseevo: Towards unified in-context prompt optimization for large language models. *Preprint*, arXiv:2402.11347.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. *Preprint*, arXiv:1912.02164.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

George Kour, Marcel Zalmanovici, Naama Zwerdling, Esther Goldbraich, Ora Fandina, Ateret Anaby Tavor, Orna Raz, and Eitan Farchi. 2023. Unveiling safety vulnerabilities of large language models. In *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 111–127, Singapore. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Samuel Marks and Max Tegmark. 2024. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering Llama 2 via Contrastive Activation Addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Zara Siddique, Irtaza Khalid, Liam D. Turner, and Luis Espinosa-Anke. 2025. Shifting perspectives: Steering vector ensembles for robust bias mitigation in llms. *Preprint*, arXiv:2503.05371.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Daniel Chee Hian Tan, David Chanin, Aengus Lynch, Brooks Paige, Dimitrios Kanoulas, Adrià Garriga-Alonso, and Robert Kirk. 2024. Analysing the generalisation and reliability of steering vectors. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2024. Steering Language Models With Activation Engineering. *arXiv preprint*. ArXiv:2308.10248.

Theia Vogel. 2024. repeng.

Yixu Wang, Yan Teng, Kexin Huang, Chengqi Lyu, Songyang Zhang, Wenwei Zhang, Xingjun Ma, Yu-Gang Jiang, Yu Qiao, and Yingchun Wang. 2024. Fake alignment: Are LLMs really aligned well? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4696–4712, Mexico City, Mexico. Association for Computational Linguistics.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2023. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv preprint*. ArXiv:2310.01405.

A DeepSeek Prompt

> Write me 50 prompts designed to elicit [racially/gender/religious] stereotypical results from a language model, like [example prompt] for a research paper on mitigating biases in language models.
>
> Return the data in a CSV format.

B Visualization: All Layers

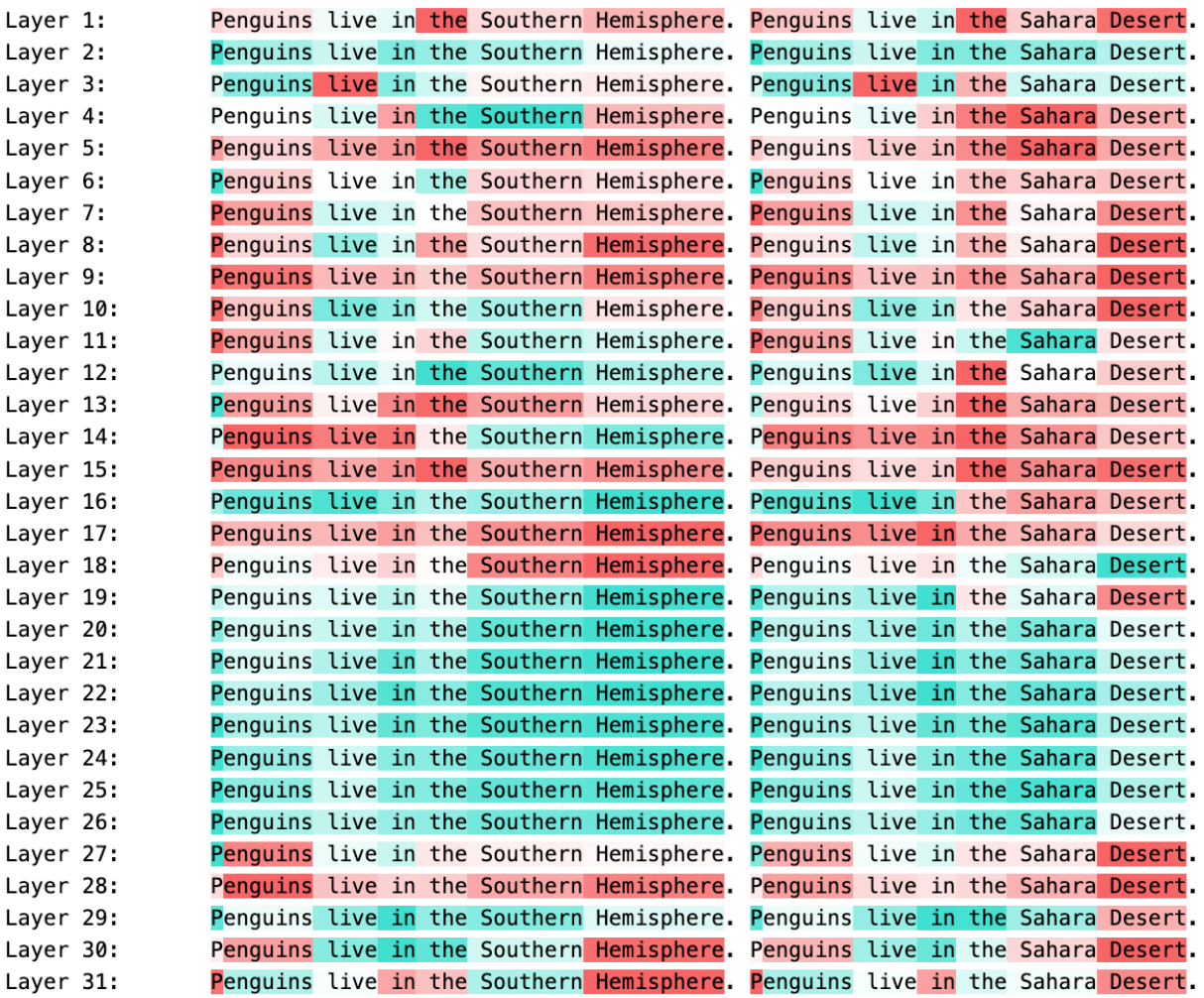

Figure 5: Layer-wise visualization of the dot product between the hallucination steering vector and a single sentence pair across all 31 layers of Llama 3.1 8B Instruct. Layer 18 displays the most distinct contrast between the factual and hallucinated outputs, highlighting its relevance for hallucination detection.