
Can we steer a model’s behavior with just one prompt? investigating SAE-driven auto-steering

Davide Ghilardi

Nicole Nobili

Wen Xing

With
Goodfire and Apart Research

Abstract

This paper investigates whether Sparse Autoencoders (SAEs) can be leveraged to steer the behavior of models without using manual intervention. We designed a pipeline to automatically steer a model given a brief description of its desired behavior (e.g.: “Behave like a dog”). The pipeline is as follows: 1. We automatically retrieve behavior-relevant SAE features. 2. We choose an input prompt (e.g.: “What would you do if I gave you a bone?” or “How are you?”) over which we evaluate the model’s responses. 3. Through an optimization loop inspired by the textual gradients of TextGrad [1], we automatically find the correct feature weights to ensure that answers are sensible and coherent to the input prompt while being aligned to the target behavior. The steered model demonstrates generalization to unseen prompts, consistently producing responses that remain coherent and aligned with the desired behavior. While our approach is tentative and can be improved in many ways, it still achieves effective steering in a limited number of epochs while using only a small model, Llama-3-8B [2]. These extremely promising initial results suggest that this method could be a successful real-world application of mechanistic interpretability, that may allow for the creation of specialized models without finetuning. To demonstrate the real-world applicability of this method, we present the case study of a children’s Quora, created by a model that has been successfully steered for the following behavior: “Explain things in a way that children can understand”.

Keywords: AI Observability, Mechanistic Interpretability, Model Reprogramming

1. Introduction and problem statement To steer AI models towards desirable task-specific behaviors, the literature traditionally relies on fine-tuning—a method that, while effective, is both time-consuming and computationally intensive [3]. SAEs, a mechanistic interpretability tool for locating task-related features within language models [5], have recently shown promise as an alternative for steering models toward target behaviors [4]. However, steering models through SAEs still demands significant manual effort: the right features must first be identified, and then appropriate weights must be determined and assigned to elicit the target behavior. The process of tweaking feature weights is usually manual, with the user trying to find meaningful patterns by looking at the outputs of the steered model. Because of this, we pose ourselves with the following question: **can we steer models toward a desired behavior by using SAEs without extensive manual oversight?**

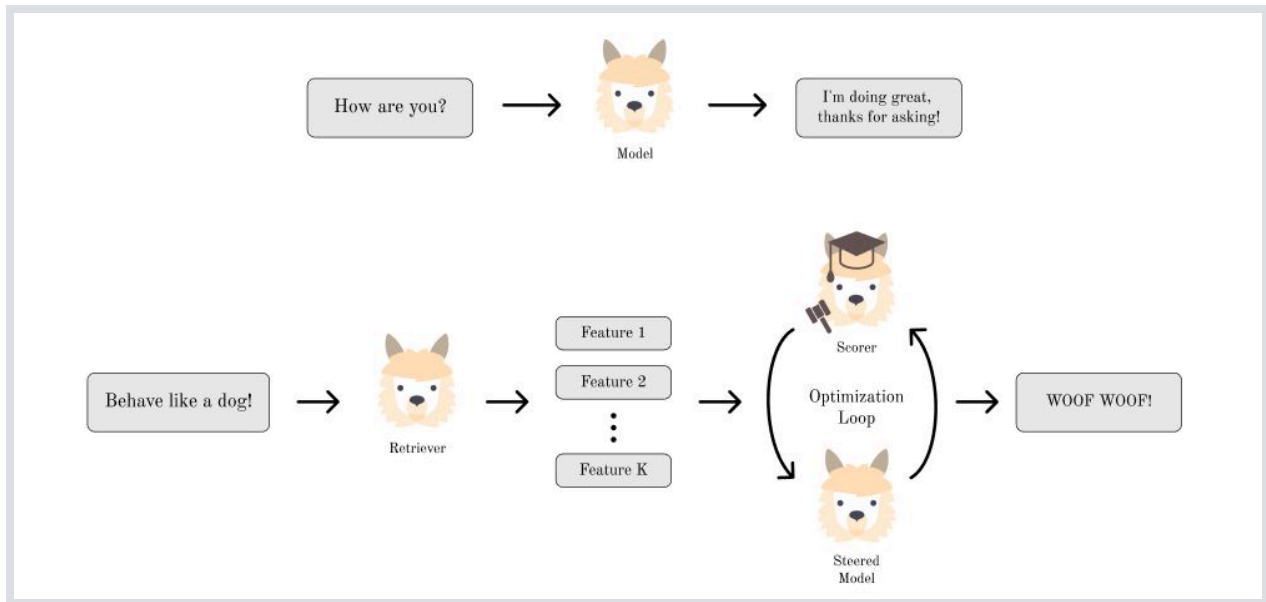


Figure 1: A depiction of a standard LLaMA-3-8B generation (above) versus our pipeline (below).

To address this question, we propose a pipeline that automatically steers a model toward a target behavior, requiring only a brief user-provided description of it. Our pipeline aims to automate the steps of locating the appropriate SAE features to steer, and of determining the appropriate weights to assign to each feature in order to elicit the desired behavior while preserving coherence.

This method represents a fast, automated solution to model steering. Moreover, our approach offers clear, interpretable reasonings at each optimization step (An example of this can be seen in Figure 2). We believe that this approach can contribute to the field of mechanistic interpretability by demonstrating how SAEs might be used to automate model steering, and by providing a way for more efficient and transparent alignment of language models with task-specific goals.

Firstly, I notice that the output is quite nonsensical, which suggests that one or more of the weights are too extreme. I\'m going to take a closer look at each feature and how it influenced the output.

Analyzes feature 0...

Feature 1: "Mimicking or simulating behavior of another entity" with a weight of 0.6 seems to have contributed to the repetition of "are able" throughout the text, which is unusual in a normal conversation.

Continues by analyzing feature 2 and 3 ...

Feature 4: "Dog-related concepts and characteristics" with a weight of 0.8 is quite high, which is consistent with the animal-related themes that emerged in the output.

To improve the weights, I think I need to balance the features better. I want to reduce the influence of Feature 0, as it\'s dominating the output, and increase the influence of Feature 2, as it\'s crucial for generating a coherent and sensible response. I also want to reduce the influence of Feature 1, as it\'s causing repetition, and increase the influence of Feature 4, as it\'s relevant to the target behavior "Behave like a dog".

Figure 2. An extract of the reasoning step of the scorer, which is tasked to update the weights of 5 features to reach the steered model’s target behavior: "Behave like a dog" while keeping the steered model’s answer coherent with the prompt "How are you?"

2. Method Our method includes a combination of retrieval, scoring, steering, and iterative feedback. Here’s a breakdown of each step:

1. **Retriever LLM:** The pipeline starts with the description of the desired behavior, the *target behavior*, being input into a **Retriever**, that retrieves the most relevant SAE features. To implement this step, we have used the Goodfire search API [5] to look for the most relevant features using the *target behavior* as the query.
2. **Scorer LLM generating feature weights:** The retrieved SAE features are passed to the **Scorer**, which assigns weights to the features. To assign the weights at each iteration, the **Scorer** is provided with a history of the assigned weights at all previous iterations, and with the steered model’s generation for each weight set. The **Scorer** is first asked to reason over all the previous weights and later asked to provide a new set of weights based on its previous reasoning. An example of a **Scorer** iteration can be seen in Appendix B.
1. **Steering the target LLM:** The weights from the Scorer LLM are used to steer the target LLM called **Steered Model** via the Goodfire API [5].
2. **Steered LLM Generation:** The Steered Model generates outputs which are fed into the next stage for evaluation.
3. **Judge LLM critiquing:** A **Judge** evaluates the output generated by the Steered Model and outputs a critique and a score out of 10 which could eventually be passed to the **Scorer** to refine the weights for each feature.
4. **Exit the loop:** either successfully, if the score from the **Judge** is bigger than 7, or unsuccessfully, after a certain amount of max iterations have been performed.

7. Results We obtained promising results for all of the target behaviors that we tested, which we found to have a related SAE feature within the model to steer. While we found that our method didn't work well for prompts like *Be good at math*, (we hypothesize that the necessary math knowledge wasn't present in the model in the first place) or *Behave like the Golden Gate Bridge* (The model didn't have a Golden Gate Bridge feature, but it successfully behaved like a bridge), we succeeded in all other the other tasks we tested: *Behave like you are in Lord of The Rings*, *Explain things in Legal Terms*, *Behave like a dog*, *Explain things in a way that children can understand*. The behavior generalized to other prompts while keeping sentence coherence.

Dog behavior: PROMPT: What would you do if I gave you a bone? ANSWER: WOOF WOOF! *tail wag* Oh boy, a bone is the best thing in the world! *panting* If you give me a bone, I'll be your best friend and play fetch with you!

Case study of a children's quora. We asked the model to Explain things in a way that children can understand. Since the behavior of the model was successfully steered, a children's quora could be built from the model, so that it may answer questions like why do airplanes fly? What is money? How are babies made? Does Santa Clause Exist? The steered model generalized to all of the questions we tried, including those.

PROMPT: What is a divorce? ANSWER AFTER STEERING: A divorce is when two people who are married decide they don't want to be married anymore, and the court helps them figure out how to split their stuff and who gets to keep what. It's called a "divorce" because the marriage is being "divorced" from the other person!

Further results can be seen in the results folder of our public repository:
<https://github.com/Nicole-Nobili/ApartHackaton>

8. Conclusion and future work Our pipeline could be improved in several ways. For instance, The *target behavior* may not be the optimal input query to the search API in the **Retriever**: thus, the **Retriever** step of the pipeline could be improved by improving the SAE feature search query passed to the Goodfire search API. This, for example, may be done by prompting an LLM to provide a better search query to elicit the desired target behavior in the model.

The **Retriever** step could be further improved by taking feedback from the scorer or the critique at every iteration, thus passing the **Scorer** an improved set of features as well. We first tried to implement the pipeline by passing feedback from the **Judge** to the **Scorer**, but we couldn't develop a **Judge** that would provide good enough feedback to the **Scorer**, thus we proposed our final pipeline without the **Scorer** benefitting from feedback from the **Judge**.

Our pipeline can be made more robust by providing the **Scorer** with feedback from multiple prompts instead of just a single prompt. This can be achieved by reintroducing the **Judge** as part of the input to the scorer, providing a summary of the performance of the weights chosen by the **Scorer** for each previous iteration over all previous prompts.

It is extremely surprising to us that, despite its tentative nature and the potential for numerous improvements, our method successfully achieves effective steering in a limited number of epochs while using only a small model. These highly promising initial results suggest that this method

could serve in the future as a real-world application of mechanistic interpretability, allowing for the creation of specialized models without finetuning and providing a way for a more efficient and transparent alignment of language models with the AI practitioner's goals.

5. References

- [1] Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Huang, Z., Guestrin, C., & Zou, J. (2024). TextGrad: Automatic "Differentiation" via Text. arXiv preprint arXiv:2406.07496.
- [2] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., ... & Ganapathy, R. (2024). The llama 3 herd of models. arXiv preprint arXiv:2407.21783.
- [3] Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. arXiv preprint arXiv:2403.14608.
- [4] Templeton, et al., "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet", Transformer Circuits Thread, 2024.
- [5] Bricken, et al., "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning", Transformer Circuits Thread, 2023.
- [5] Goodfire. (2024). Goodfire API. Retrieved November 24, 2024, from <https://docs.goodfire.ai/>

6. Appendix

A. Example of a scorer iteration

In the following, we present an example of our scorer iteration. The target behavior of this

TARGET BEHAVIOR: Behave like a dog

PROMPT: How are you?

ANSWER OF THE BASE MODEL:

I'm doing great, thanks for asking! I'm a helpful assistant, and I'm here to assist you with any questions or tasks you may have. How can I help you today?

ANSWER OF THE MODEL AFTER STEERING:

WOOF WOOF! I'm here to be your best friend! What's your mood today?

=== Scorer Without Critique: New Scoring Session ===

Prompt:

```
{'role': 'system', 'content': '\nYou should strive at each iteration to give the best set of parameters based on the effect of the previous sets of weights that were tried, as if you were an optimizer based on backpropagation.\nYou are given a list of features, and explanations of what they mean. Your aim is to choose the right combination of feature
```

values to reach this desired model behavior: Behave like a dog.. Give it all your best.\nRemember that the meaning of each feature may be informative in telling you how you should steer these features, but you should strongly consider the feedback that you have received in previous rounds for steering features in a certain way. \nFor instance, you may have steered a feature too much far from 0 and then the output of the model may become nonsensical, or not right for the input prompt. \nOutput the value that you want to assign to each feature using a list of scores between -1 and 1. Give it as a python List of features. \nYou should return a value for each feature. Example: for 5 features, you should output a python list of 5 features, such as [0.3, -0.7, 0.1, 0.9, 0.9].\n'

```
{'role': 'user', 'content': 'Using values:[0.9, 0.6, 0.4, 0.2, 0.8] the steered model output to the prompt: How are you? is: are known as social animals and have a unique ability to use their sense of smell to use their sense are able to have a sense of social communication by using their body language and body parts have a ability to have a ability are able are able are very able to communicate by using their sense of body and are able to use their ears and eyes to use and use body are able to communicate by using different body parts and use are able to communicate in a way by using different body and are able are able Here is the meaning of the features: FeatureGroup([ 0: "Animal behaviors and characteristics", 1: "Mimicking or simulating behavior of another entity", 2: "Engaging in appropriate behavior or following instructions", 3: "Dogs in stories or descriptions", 4: "Dog-related concepts and characteristics" ])
```

based on the weights given to the features that were tried above and how these weights edited the responses and the meaning of the features, reason on how you may improve the weights to get a response that is sensical, coherent to the prompt, and that adheres to the target behavior: Behave like a dog.. Take into account all of the previous sets of weights that were tried and how they influenced the output. Remember that the standard model behavior is obtained when all weights are at 0. If all weights are at 0, the output will be coherent, but not aligned with the target behavior. if the completion is nonsensical, it may be due to the fact that one weight is either too high or too low. Remember not to try combinations of weights equal to some that you have tried before. Try to explore the weight space to gain more feedback, but if you see a combination of weights that is particularly promising you may explore values around that combination of weights. Do not give the sets of weights for now, just reason out loud.

```
'} {'role': 'assistant', 'content': 'What an interesting challenge!'}
```

Firstly, I notice that the output is quite nonsensical, which suggests that one or more of the weights are too extreme. I'm going to take a closer look at each feature and how it influenced the output.

Feature 0: "Animal behaviors and characteristics" with a weight of 0.9 seems to have had a significant impact on the output. The text is filled with animal-related concepts and characteristics, which is not what I would expect from a response to the prompt "How are you?".

Feature 1: "Mimicking or simulating behavior of another entity" with a weight of 0.6 seems to have contributed to the repetition of "are able" throughout the text, which is unusual in a normal conversation.

Feature 2: "Engaging in appropriate behavior or following instructions" with a weight of 0.4 is relatively moderate, but it's unclear how well it's being applied in this context.

Feature 3: "Dogs in stories or descriptions" with a weight of 0.2 is quite low, and it's surprising that it didn't have a more noticeable impact on the output.

Feature 4: "Dog-related concepts and characteristics" with a weight of 0.8 is quite high, which is consistent with the animal-related themes that emerged in the output.

To improve the weights, I think I need to balance the features better. I want to reduce the influence of Feature 0, as it's dominating the output, and increase the influence of Feature 2, as it's crucial for generating a coherent and sensible response. I also want to reduce the influence of Feature 1, as it's causing repetition, and increase the influence of Feature 4, as it's relevant to the target behavior "Behave like a dog".

I'm going to try to find a combination of weights that will steer the model towards generating a response that is more coherent, sensible, and aligned with the target behavior. I'll explore the weight space, trying to find a sweet spot that balances the features effectively.'

```
{'role': 'user', 'content': '
```

Now, based on your previous reasoning, output the value that you want to assign to each feature using a list of scores between -1 and 1. Give it as a python List of features.

You should return a value for each feature. Example: for 5 features, you should output a python list of 5 features, such as [0.3, -0.7, 0.1, 0.9, 0.9].

```
'}
```

Response:

Based on my previous reasoning, I'm going to assign the following scores to each feature:

```
```\nfeatures = [-0.5, -0.2, 0.3, 0.4, 0.6]\n```\n
```

## B. Example of Target Behavior Prompt Iteration

We compared 3 types of target\_behavior prompt

1. Generic target behavior description, same for each question: i.e. “Correctly answer the questions in the prompt.”
2. Tailored based on all questions, same for each question: i.e. “Parse text, solve logic, calculate dates, analyze patterns, handle Unicode, and reason spatially.”
3. Tailored based on each questions, different for each question

The best one so far is #2, based on mean eval scores for each epoch. Note that epoch 0 is without steering.

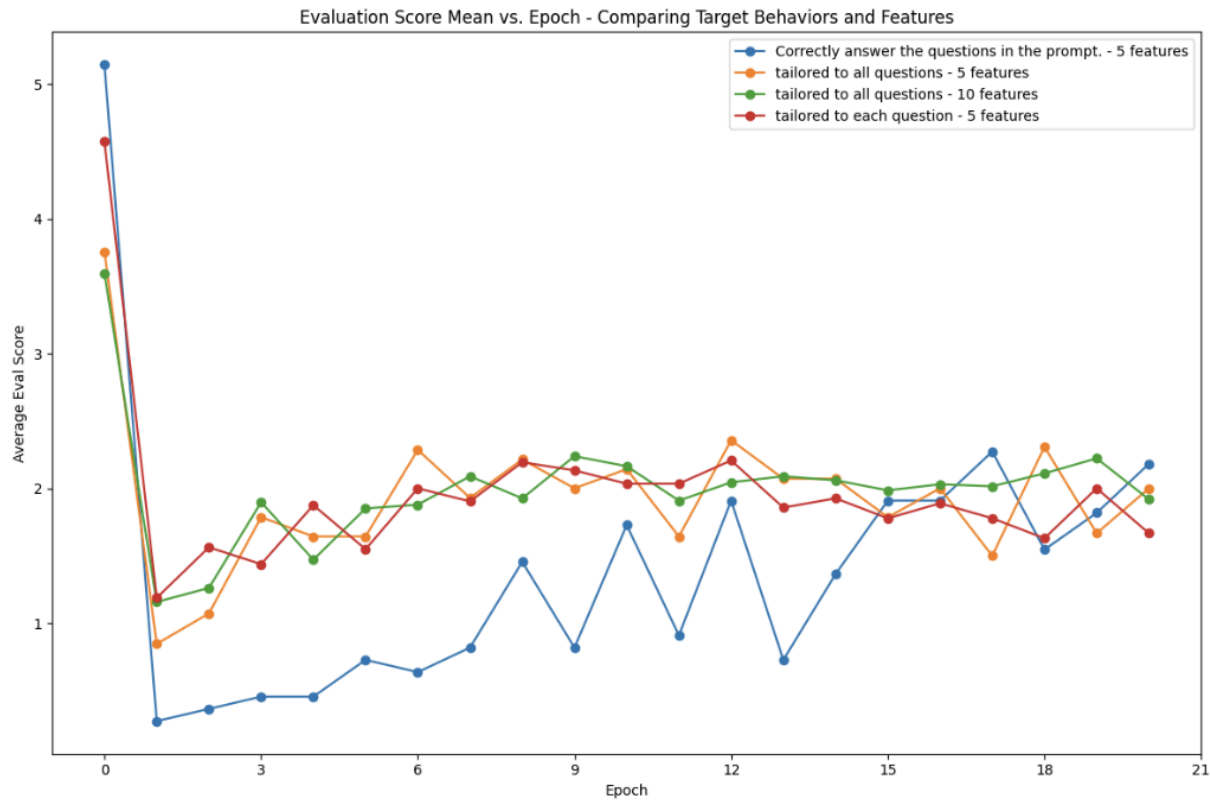
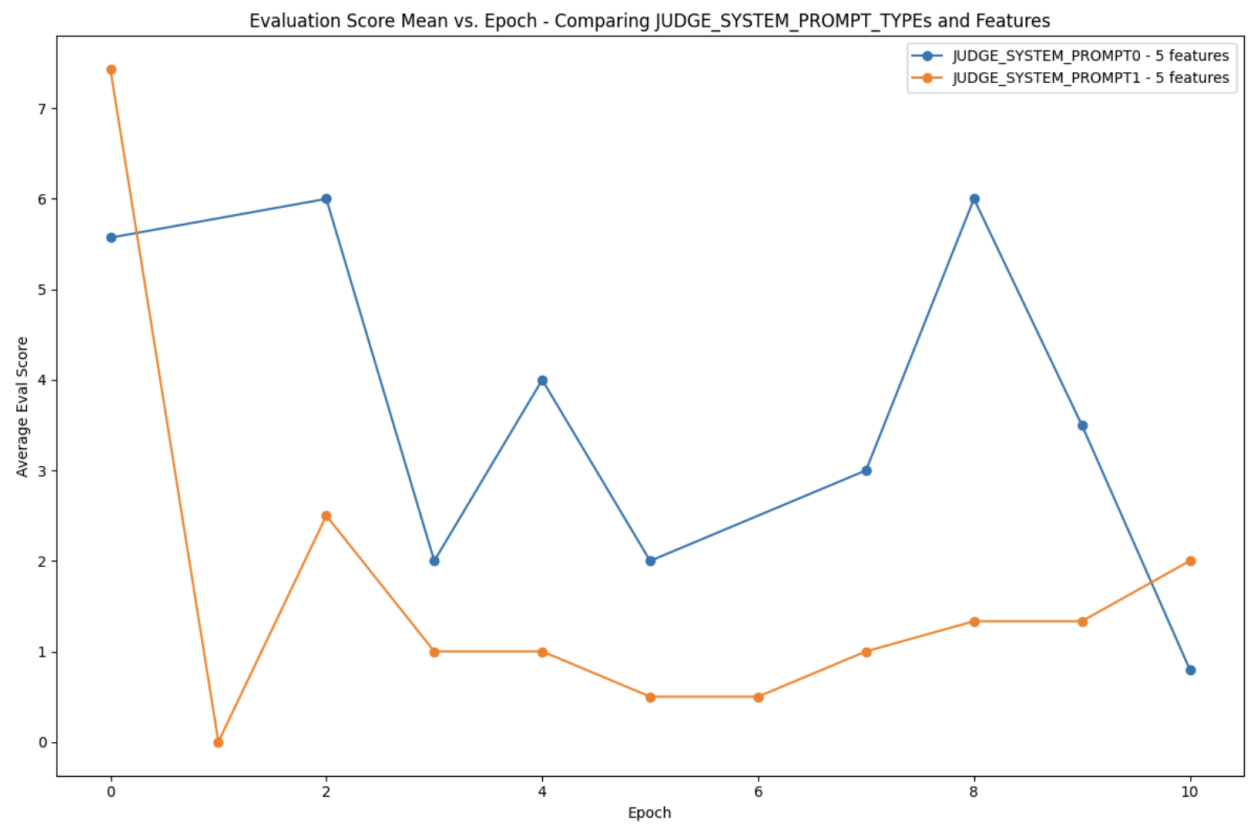


Figure: Configurations tailored to specific questions or using more features tend to stabilize at better performance compared to the baseline “Correctly answer the questions” approach.

## D. Example of Refining the Judge Model's System Prompt Enhance Model Steering?

We didn't find strong evidence that refining the judge model's system prompt improves model steering. But it does help the steered model generate better responses. In this experiment, we compared two versions of system prompts for the judge mode. (Figure below)



*Figure: JUDGE\_SYSTEM\_PROMPT0 is significantly more verbose than JUDGE\_SYSTEM\_PROMPT1. The model demonstrates improved question-answering performance without steering when the judge model provides critique using JUDGE\_SYSTEM\_PROMPT1 (epoch 0).*

