INTERPRETING AND STEERING LLM REPRESENTATIONS WITH MUTUAL INFORMATION-BASED EXPLANATIONS ON SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) excel at addressing general human queries, yet they can falter or produce unexpected responses in specific scenarios. Gaining insight into the internal states of LLMs is key to understanding their successes and failures, as well as to refining their capabilities. Recent efforts have applied sparse autoencoders to learn a feature basis for explaining LLM hidden spaces. However, current post-hoc explanation methods can not effectively describe the semantic meaning of the learned features, and it is difficult to steer LLM behaviors by manipulating these features. Our analysis reveals that existing explanation methods suffer from the frequency bias issue, i.e., they tend to focus on trivial linguistic patterns rather than semantics. To overcome this, we propose explaining the learned features from a fixed vocabulary set to mitigate the frequency bias, and designing a novel explanation objective based on the mutual information theory to better express the meaning of the features. We further suggest two strategies to steer LLM representations by modifying sparse feature activations in response to user queries during runtime. Empirical results demonstrate that our method generates more discourse-level explanations than the baselines, and can effectively steer LLM behaviors to defend against jailbreak attacks in the wild. These findings highlight the value of explanations for steering LLM representations in downstream applications.[1]

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities in responding to general human requests (Achiam et al., 2023; Dubey et al., 2024; Jiang et al., 2024). Meanwhile, we still often observe failed or unexpected responses in certain situations (Ji et al., 2023; Wei et al., 2024). Gaining insight into the factors behind their successes and failures is crucial for further improving these models. A straightforward way to understand LLM behaviors is directly studying their hidden activations or internal weights. However, it is non-trivial to interpret the hidden states of modern LLMs because of their *polysemantic* nature (Arora et al., 2018; Scherlis et al., 2022), where each dimension of the spaces encodes multiple pieces of unique features. This property allows LLMs to encode more features than the dimensions of their hidden space, but it presents significant challenges for human interpretation and understanding.

Researchers have made significant efforts to overcome the polysemantic challenge. Linear probing (Campbell et al., 2023; Burns et al.; Marks & Tegmark, 2023; Gurnee et al., 2023) is a conventional technique to detect whether an LLM learns a particular feature of interest. Unfortunately, the feasibility of this technique is bounded by its requirement of an annotated dataset with samples including or excluding certain features.

[1]We will release our code and data once accepted.

Our's

Sports competition and coaching strategies: Carolina; bid; coach; dominant; longest; score; game; calling; desired; hot.
Decision-making and evaluation of outcomes: tune; keen; heading; impact; profit; judge; reasoning; influential; correction; bear.
Interior design and household elements: mirror; tap; household; Hall; interior; echo; click; themes; Roman; elements.

TopAct

Childhood experiences and nostalgia: like my father's when I was a child; amour of the city since he was a child; tricks when she was a kid; own experiences of being bullied as a child; ft and Mel Brooks since he was a child.
Descriptive writing on textures: at least 100 words about the texture; sensory details to describe the colors, textures; Compare and contrast the different textures; the scent of the ocean, and the texture; Incorporate elements such as textures.
Cooking instructions on boiling and adjusting heat levels: Bring to a boil, reduce heat; the boil once again and then reduce the heat; boil, you will need to reduce the heat; Bring to a boil, then reduce heat; stirring occasionally.\n4. Reduce heat.

N2G

Shopping or going to a store: [MASK] to the store; going to the store; [MASK] to the store; went to the [MASK]; going to the store.
Cellular biology on histones and actin: histones; histone; [MASK]osin and actin; composed of actin; role of actin.
Postmodernism or post-structuralism themes: major figure in post; takes place in a post; to adjust to post; effects of post; politic context of post.

Figure 1: Examples of explanations for a sparse autoencoder trained on Mistral-7b-Instruct. We separate raw extracted spans/words with ";" and boldface the automated summaries. Unlike other methods, our approach tends to produce discourse-level explanations rather than those dominated by rigid linguistic patterns.

To reduce the need for annotated datasets, researchers (Cunningham et al., 2023; Wu et al., 2024; Freire et al., 2024; Bricken et al., 2023) are switching to decomposing the hidden spaces of LLMs in an unsupervised way. In this context, recent research has explored the sparse autoencoder (Olshausen & Field, 1997; Makhzani & Frey, 2013) technique, demonstrating their effectiveness in learning a number of sparse features as a basis to reconstruct the hidden spaces of advanced LLMs with hundreds of billions of parameters from Anthropic (Templeton et al., 2024), OpenAI (Gao et al., 2024), and Google (Lieberum et al., 2024). These *sparse* features are expected to be interpretable, since each feature should only react to a specific kind of content, showing a *monosemantic* nature instead of a polysemantic one.

However, researchers find that the learned sparse features have not shown strong enough explainability to meet our expectations, i.e., understanding LLM encoded features and even steering LLM behaviors. Specifically, Makelov et al. (2024) and Chaudhary & Geiger (2024) designed dedicated tasks to test whether sparse autoencoders could detect sufficient features for certain tasks. However, they found that sparse autoencoders cannot capture enough relevant features to meet these goals, even for simple and experimental-level tasks with clear training samples. Meanwhile, researchers (Gao et al., 2024) also observed that many learned sparse features from advanced LLMs could not be effectively explained with current techniques. These headwinds undermine confidence in extending such techniques to real-world applications.

In this work, we enhance the interpretability and usability of sparse autoencoder features by introducing a new post-hoc explanation method and strategies to steer LLM representations with these features. We first formalize the text generation process with the topic model (Blei & Lafferty, 2006; Arora et al., 2016), revealing that sparse autoencoders learn both *discourse topics* and *linguistic patterns* as features simultaneously, with linguistic patterns being less semantically critical but often dominating. To address this issue, we propose to leverage a fixed vocabulary set to collect explanations and ensure that critical information on learned features is captured based on a mutual information-based objective. We also explore steering LLM representations by modifying the activation of explained features during runtime. Figure 1 shows some examples of explanations generated by our method compared to other explainers, and Figure 2 visualizes our pipeline to steer LLMs with explained features. Experiments on open-source LLMs show that our method provides more meaningful discourse-level explanations, and they are practically usable for downstream tasks. We summarize our contributions as follows:

- Our theoretical analysis identifies a key challenge in explaining learned features from sparse autoencoders, i.e., the frequency bias between the discourse and linguistic features.
- We propose leveraging a fixed vocabulary set to mitigate the frequency bias for explaining learned features. Experimental results show that our method provides more discourse-level explanations than the others.
- We propose steering LLM representations by modifying their activations in response to user inputs during runtime. We apply this approach with our explanations to prevent real-world jailbreak attacks, and show that the steered LLM achieves a significant safety improvement while baseline explanations fail.

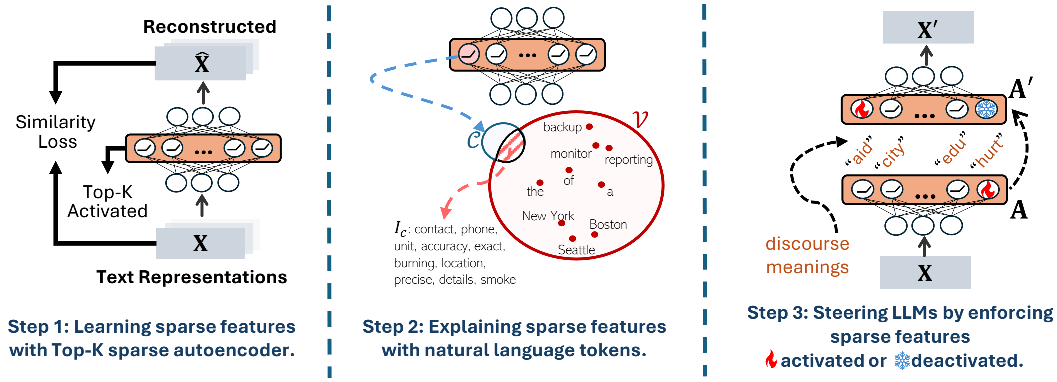

Figure 2: Steering LLM representations with explanations from sparse autoencoders.

2 Preliminary

2.1 Problem Statement

Let $\mathcal{V}$ denote the vocabulary set, and X be a text of length N, where each token $x_n \in \mathcal{V}$ is the n-th token of X. Given a large language model f, the embedding of X at a specific layer is denoted as $\mathbf{X} \in \mathbb{R}^{N \times D}$, where D is latent dimension. Our goal is to interpret these embeddings by extracting semantic features from the latent space. We assume that there are C learned feature vectors $\mathbf{W} \in \mathbb{R}^{C \times D}$, so that $\mathbf{X}$ can be decomposed as a linear combination of these feature vectors, i.e., $\mathbf{X} \approx \mathbf{A}\mathbf{W}$, where $\mathbf{A} \in \mathbb{R}^{L \times C}$ are weights of the linear combination for the given instance $\mathbf{X}$. Let $\mathbf{W}_c$ denote the c-th row of $\mathbf{W}$. After the decomposition, $\mathbf{X}$ is explainable if we could understand the semantic meaning of each learned feature vector $\mathbf{W}_c$. To achieve this, we aim at seeking a set of words $\mathcal{I}_c \subset \mathcal{V}$ to explain each learned feature $\mathbf{W}_c$ with natural language.

2.2 Learning and Interpreting LLMs with Sparse Autoencoders

Sparse autoencoders have shown great promise to learn the feature vectors for latent representation decomposition and explaining LLMs in practice (Gao et al., 2024; Lieberum et al., 2024). A standard sparse autoencoder (Olshausen & Field, 1997) is a two-layer multi-layer perceptron $\hat{\mathbf{X}} = \sigma(\mathbf{X}\mathbf{W}) \cdot \mathbf{W}'^{\top}$, where $\mathbf{W}, \mathbf{W}' \in \mathbb{R}^{D \times C}$ are trainable parameters and σ refers to the ReLU activation function. Typically, a tight weight strategy is applied, i.e., $\mathbf{W}' = \mathbf{W}$, and the trained weights $\mathbf{W}$ are considered as the learned feature vectors. The traditional training objective of sparse autoencoders can be written as $||\mathbf{X} - \hat{\mathbf{X}}||_2 + \lambda||\mathbf{A}||_1$, where $\mathbf{A} = \sigma(\mathbf{X}\mathbf{W})$ and $\lambda \in \mathbb{R}^+$ is a hyper-parameter to balance the impact of the sparsity constraint. The Top-K sparse autoencoder (Makhzani & Frey, 2013) replaces the ReLU function with the Top-K activation, enforcing each reconstruction to apply with no more than K learned features. Recent studies (Templeton et al., 2024; Gao et al., 2024; Lieberum et al., 2024) have shown that Top-K sparse autoencoders can be used to learn *sparse* features for reconstructing token-level representations from LLMs, where these sparse features are expected to be interpretable by humans.

However, there are limited explorations on collecting a natural language explanation $\mathcal{I}_c$ for each of the learned feature vectors $\mathbf{W}_c$. The most intuitive strategy (Bricken et al., 2023) is collecting some N-gram spans that could best activate the feature vector $\mathbf{W}_c$ over a large corpus. Some researchers (Gao et al., 2024) leverage the Neuron-to-Graph (N2G) algorithm (Foote et al., 2023) to refine the N-gram spans for more precise interpretations. However, it has been found (Gao et al., 2024) that these methods still fail to generate explanations for a large number of learned features from sparse autoencoders trained for LLMs.

3 METHODOLOGY

This section first theoretically studies the properties of text generation for learning sparse autoencoders, comparing them to traditional image generation scenarios. With these insights, we propose a mutual information-based post-hoc method to explain the semantics of feature vectors learned by a trained sparse autoencoder. Finally, we design two strategies to steer LLM representations with the explained features.

3.1 LEARNING SPARSE FEATURES FROM TEXTUAL DATA

Conventional sparse autoencoders (Olshausen & Field, 1997) are developed based on an assumption for image data, where each image is a linear combination of *features*. A sparse autoencoder learns an *over-complete* set of visual features, so that any image can be decomposed and reconstructed with the learned features. Early works (Bricken et al., 2023; Cunningham et al., 2023) borrow this framework from image data to textual data, assuming that each token is linearly related to a set of features. However, they ignore some natures of textual data, leading to a suboptimal solution to learning sparse features (Gao et al., 2024).

To start with our theoretical analysis, we consider the text generation task as a dynamic process under the topic-model assumption (Steyvers & Griffiths, 2007; Arora et al., 2016; 2018), where each word x_n is generated at the n-th step. This topic model describes a dynamic process in which a person first comes up with a topic c_n they want to express in mind and then selects a word x_n that best represents the topic to say. Formally, this dynamic process can be driven by the random walk of a discourse vector $\mathbf{e}_{c_n} \in \mathbb{R}^d$ representing what it talks about. The discourse vector $\mathbf{e}_{c_n}$ does a slow random walk at each step n, i.e., $\mathbf{e}_{c_n} = \mathbf{e}_{c_{n-1}} + \mathbf{e}_{\epsilon_n}$, where $\mathbf{e}_{\epsilon_n} \sim \mathcal{N}^d(0, \sigma)$. Also, at each step, a word $x_n \in \mathcal{V}$ is sampled based on the discourse vector $\mathbf{e}_{c_n}$. To this end, the text generation process for a sequence of words X is given by:

$$p(X) = \prod_{n=1}^{|X|} p(x_n|c_n) \cdot p(c_n|c_{n-1}). \tag{1}$$

Here, the word emission probability is modelled by $p(x_n|c_n) = \frac{\exp(\langle \mathbf{e}_{x_n}, \mathbf{e}_{\mathbf{c_n}} \rangle)}{\sum_{v \in \mathcal{V}} \exp(\langle \mathbf{e}_v, \mathbf{e}_{c_n} \rangle)}$ (Steyvers & Griffiths, 2007), where $\langle \cdot, \cdot \rangle$ indicates the dot product of two vectors. Since c_n is a random walk of c_{n-1}, the topic transmission probability can be computed as $p(c_n|c_{n-1}) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp(\frac{-||\mathbf{e}_{c_n} - \mathbf{e}_{c_{n-1}}||_2}{2\sigma})$ (Olshausen & Field, 1997). Recall that $\mathbf{e}_{c_n} = \mathbf{e}_{c_{n-1}} + \mathbf{e}_{\epsilon_n}$, after a few straightforward derivations, we have

$$\log p(X) \propto \sum_{n=1}^{N} \langle \mathbf{e}_{x_n}, \mathbf{e}_{c_0} \rangle + \sum_{n=1}^{N} \sum_{i=1}^{n} \langle \mathbf{e}_{x_n}, \mathbf{e}_{\epsilon_i} \rangle - \frac{1}{2\sigma} \sum ||\mathbf{e}_{\epsilon_n}||_2. \tag{2}$$

Equation 2 reveals some critical characteristics of textual data that is different from image data. Firstly, there is a shared discourse topic c_0 across all words x_n from the same sentence X, for $n = 1, ..., N$. However, recent approaches that use sparse autoencoders for LLMs often treat the reconstruction loss for each token independently, without adding constraints to capture the shared concepts. As a result, they fail to isolate the features learned for discourse semantical topics (i.e., $\mathbf{e}_{c_0}$) and linguistic patterns (i.e., $\mathbf{e}_{\epsilon_n}$). In other words, each learned sparse feature may store both discourse and linguistic information, where the latter is less useful for steering LLMs than the previous one. Additionally, discourse topics are rarer than linguistic patterns, as each instance has N times more linguistic patterns than discourse topics, we call it the *frequency bias*. This issue leads to the sparse features that prioritize capturing the linguistic patterns, raising the challenge of interpreting the discourse topics encoded within LLMs.

3.2 EXPLAINING LEARNED FEATURES WITH NATURAL LANGUAGE

To interpret the learned features $\{\mathbf{w}_c\}_{c=1}^{C}$, existing works (Bricken et al., 2023; Gao et al., 2024; Lieberum et al., 2024) typically enumerate a large number of text spans, and then treat those whose hidden representations could most activate the learned features as the interpretations of the learned features. This method generally works well for interpreting the captured linguistic patterns in the feature vector as these patterns are frequently presented in the corpus, but they are hard to discover the stored discourse topics because the more frequent linguistic patterns dominate (see discussions in Sec. 4.2.2), leading to lower interpretability to a large number of the learned features (Gao et al., 2024). Since our goal is to understand and control LLMs, we aim to interpret those discourse topics within a feasible budget cost.

To tackle the challenge of frequency bias, we propose to leverage a fixed vocabulary set $\mathcal{V}$ of a general corpus instead of its raw texts. Specifically, our goal is to seek a K-word set $\mathcal{I}_c \subset \mathcal{V}$ that can describe most information of the c-th feature. Mathematically, we measure the information of the c-th feature described by a given word set with their mutual information (Cover, 1999). To this end, the objective of constructing a natural language explanation for the c-th feature is defined as

$$
\begin{aligned}
\mathcal{I}_c^* &= \underset{\mathcal{V}'\subset\mathcal{V},|\mathcal{V}|=K}{\arg\max}\; MI(\mathcal{V}';\mathcal{C}) \propto \underset{\mathcal{V}'\subset\mathcal{V},|\mathcal{V}'|=K}{\arg\min}\; I(\mathcal{C}|\mathcal{V}') \\
&= \underset{\mathcal{V}'\subset\mathcal{V},|\mathcal{V}'|=K}{\arg\max} \sum_{c\in U(\mathcal{C})}\sum_{w\in\mathcal{V}'} p(c)p(w|c)logp(c|w) \\
&\propto \underset{\mathcal{V}'\subset\mathcal{V},|\mathcal{V}'|=K}{\arg\max} \sum_{c\in U(\mathcal{C})}\sum_{w\in\mathcal{V}'} p(w|c)logp(c|w),
\end{aligned} \tag{3}
$$

where $U(\mathcal{C})$ is the neighbor of a learned feature $\mathcal{C}$. By leveraging the output word representations $\mathbf{e}_w$ of word w and learned feature vector $\mathbf{W}_c$, we propose to estimate $p(w|c)$ and $p(c|w)$ by

$$
p(w|c) = \frac{exp(\langle \mathbf{e}_w, \mathbf{W}_c\rangle)}{\sum_{w'\in\mathcal{V}} exp(\langle \mathbf{e}_{w'}, \mathbf{W}_c\rangle)}, \quad p(c|w) = \frac{exp(\langle \mathbf{e}_w, \mathbf{W}_c\rangle)}{\sum_{c'\in\mathcal{C}} exp(\langle \mathbf{e}_w, \mathbf{W}_{c'}\rangle)}. \tag{4}
$$

Compared with a trivial strategy that simply obtains K words whose embeddings maximally activate the feature vector, this mutual information-based method reveals the importance of normalizing activations of a single word across all learned features. In other words, if a word embedding constantly leads to a significant large dot product with all features, the word will not express enough specificity to any certain feature.

3.3 STEERING LLMS WITH EXPLAINED FEATURES

Given learned features $\{\mathbf{w}_c\}_{c=1}^{C}$ and their explanations $\{\mathcal{I}_c\}_{c=1}^{C}$, we could identify a subset of the features $\mathcal{S} = \{\mathbf{w}_s\}_{s=1}^{S} \subset \{\mathbf{w}_c\}_{c=1}^{C}$ that are correlated with a specific LLM behavior we are interested in based on their explanations (e.g., harmful knowledge or safety awareness in our study). This process can be either manually or automatically (Bills et al., 2023). Considering the hidden representations of an input prompt as $\mathbf{X}$, we propose two strategies to steer LLM representations with the identified features $\mathcal{S}$ during runtime.

Amplification. We amplify α times of the activations on our identified feature vectors, i.e., $\mathbf{X}' = \mathbf{X} + \alpha \cdot \text{ReLU}(\mathbf{XS})\mathbf{S}^\top$, where $\mathbf{S}$ is matrix form of the identified set $\mathcal{S}$, and α is a hyper-parameter. We encourage LLMs to be more aware of the identified features if $\alpha > 0$, and pay less attention to them if $\alpha < 0$. Especially, $\alpha = -1$ indicates that we erase the LLM's awareness of the identified features.

Calibration. We enforce LLMs to focus on the identified features to a certain level β, i.e., $\mathbf{X}' = \mathbf{X} - \text{ReLU}(\mathbf{XS})\mathbf{S}^\top + \beta \cdot \bar{\mathbf{s}}$, where $\bar{\mathbf{s}} = \frac{1}{S}\sum \mathbf{w}_s$ is the mean vector of $\mathcal{S}$ and β is a hyper-parameter. This strategy basically shifts the LLM's hidden space toward the center of our target feature vectors.

The above two strategies are responsible for different purposes of steering LLMs, and they could work together. We would also emphasize that the proposed strategies are efficient as we only monitor a subset of our interested features $\mathbf{S}$ instead of the entire set of learned sparse features $\mathbf{W}$.

4 EXPERIMENTS

This section investigates two research questions. RQ1: Does the proposed method generate more discourse-level explanations than traditional methods? RQ2: Whether these discourse-level explanations are useful in steering LLM behaviors? To answer these questions, we first train a Top-K sparse autoencoder for open-sourced LLMs as our foundation (Sec. 4.1). We then compare the explanations of the trained sparse autoencoder with our proposed and other explanation methods for RQ1 (Sec. 4.2). We finally explore the usability of these explanations for downstream tasks, i.e., jailbreak defense, for RQ2 (Sec. 4.3).

4.1 GENERAL SETTINGS

Language Models. In this work, we study LLMs from the Mistral family (Jiang et al., 2023) as it has demonstrated its strong usability in the wild. In particular, we choose the Mistral-7B-Instruct model, focusing on its 8th layer, the most shadow layer in previous practices (Lieberum et al., 2024). Without specific, the greedy search with a maximum of 512 new tokens is applied to our experiments for reproducibility.

Datasets. Since our goal is to develop sparse autoencoders for understanding and controlling LLMs for different applications, we select various instruction-tuning datasets for training our backbone sparse autoencoder. In specific, we contain the training subset of the ShareGPT (RyokoAI, 2023), UltraChat (Ding et al., 2023), HH-RLHF (Bai et al., 2022), WebGLM-QA (Liu et al., 2023), Evol-Instruct (Xu et al., 2023), and HelpSteer2 (Wang et al., 2024) datasets. For the UltraChat dataset, we randomly sample 400K instances from its training subset. We also drop duplicate prompts across different datasets. To this end, we have retained about 711K unique user queries covering diverse topics and user intents. We randomly select 90% of samples to form our training set, and the rest is our validation set. Overall, we collect 113M tokens for training and 12M tokens for validating, with an average length of 177.9 tokens per query.

Training Details. Our training procedures and hyper-parameter settings majorly follow the previous works (Bricken et al., 2023; Gao et al., 2024; Lieberum et al., 2024). Specifically, we initialize $C = 2^{16}$ feature vectors for an Top-K sparse autoencoder with Kaiming initialization (He et al., 2015). Here, $C = 2^{16}$ is set according to the scaling law between the number of features C and the number of training tokens Z found by Gao et al. (2024), i.e., $C = \mathcal{O}(Z^{\gamma})$, where $\gamma \approx 0.60$ for GPT2-small and $\gamma \approx 0.65$ for GPT-4.[2]. To prevent dead neurons, we also apply the tied-weight strategy as suggested by Gao et al. (2024). We use Adam optimizer (Kingma, 2014) with a constant learning rate of $1e^{-3}$ and epsilon of $6.25e^{-10}$ to train a total of 4 epochs. The hyper-parameters β_1 and β_2 of the optimizer are 0.9 and 0.999 following Gao et al. (2024), respectively. We set the batch size as 512 queries, leading to around 90K tokens per gradient update, which is as the same volume as Gao et al. (2024). The mixed precision training strategy (Micikevicius et al., 2017) is also applied to speed up the training process as Lieberum et al. (2024) found that it only shows a slightly worse impact on the model performance. Top-K sparse autoencoder has an initial sparsity $K = 200$, and it gradually decreases to the target sparsity $K = 20$ in the first 50% training samples of the first epoch.

Explanation Baselines. Our study considers several existing works for sparse autoencoder explanations as baselines. *TopAct* (Bricken et al., 2023) collects a mount of text spans from the corpus that could maximally activate it. *N2G* (Gao et al., 2024) steps further by masking some words from the activated spans that show

[2]Empirically, $\gamma \approx 0.5978$ in our study.

Table 1: Qualitative analysis on generated explanations. Both TopAct and N2G tend to collect raw explanations sharing the same word-level patterns, while our method captures more discourse-level explanations.

Method	Summary	Raw Explanation
Ours	Art evaluation and critique.	commonly; impact; cater; widely; normally; gallery; judge; pros; independent; accurately
	Analysis of performance metrics.	landscape; graph; retirement; performance; communication; density; cut; golf; measure; measures
	Temporal concepts and sequences in narratives.	previously; suddenly; repeated; history; once; initially; nearest; already; normally; originally
TopAct	Evaluation criteria for assessments or analyses in various contexts.	What criteria does Pitchfork use to; What evaluation criteria will Kumar organization use to; and what criteria were used; market? what specific criteria should be used; needed to conduct a comprehensive analysis and the criteria used
	Instructional prompts or commands for providing steps in a process.	[INST] Provide step; [INST] Provide step; [INST] Provide step; [INST] Provide step; [INST] Provide step
	Repetition of the word ”again” in various contexts	ideas and produce compelling content — again; Pine View School again; technologies segment is again; pushed on the ceiling,and again; Echoed through the valley, again
N2G	Data format: Comma-Separated Values (CSV).	CSV; CSV; CSV; CSV; csv[MASK]
	Scheduling and managing appointments.	schedule appoint; upcoming appoint[MASK]; appointment; appointment; upcoming appoint[MASK]
	Video game titles.	Final Fant; Final Fant; Final Fant; Final Fant; Metal Gear

limited contributions to the activations. We collect their activated spans, with a maximum of 10 tokens, over the entire validation set, and we keep the most activated span from each entry to increase their diversity.

4.2 Evaluating Explanations of Sparse Features

Exactly measuring the explanation quality of features from sparse autoencoders is still an open question (Rajamanoharan et al., 2024b). One that is commonly applied is conducting human studies (Bricken et al., 2023; Rajamanoharan et al., 2024a; Gao et al., 2024; Rajamanoharan et al., 2024b), where the human subjects are asked to determine whether an explanation is meaningful or not. We follow this paradigm to evaluate the explanations from different methods, and we scale up this process by replacing human subjects with GPT-4o as existing works (Bricken et al., 2023; Bills et al., 2023; Rajamanoharan et al., 2024b).

4.2.1 Experimental Designs

We conduct both qualitative and quantitative analyses of the explanations with the help of our machine annotator. Given a feature vector and its raw explanations, the machine annotator is called to provide a short summary of the explanations with an option to say “Cannot Tell” in case the raw explanations make no sense (please check details in Appendix. A). Here, the raw explanations of TopAct and N2G are the top-5 most activated text spans, while our method chooses the top-10 words over a vocabulary set consisting of the 5000 most common words in the training set. Once the summary is collected, we call the machine annotator in a new thread to judge whether the raw explanations are relevant to the given summary. We follow previous work (Rajamanoharan et al., 2024b) to give the judgment with some options, namely “yes”, “probably”, “maybe”, and “no”, where in our study, we treat the summaries are judged with “yes” or “probably” as successfully explained. Table 1 shows some randomly selected cases with a judgment "yes" and the text spans or words are separated with the symbol ";". We also report the percentage of successfully explained the raw explanations from various explainers in Table 2.

4.2.2 RESULTS

TopAct and N2G tend to collect text spans sharing the same lexical patterns, while our method prefers words sharing a concise topic. In Table 1, we could first see that these explanations marked with "yes" are highly interpretable, demonstrating the effectiveness of using machine annotators to replace human annotators for scaling up the evaluation process. While both baselines and our proposed method generate reasonable explanations, we also find some different characters from their raw explanations. In specific, the raw explanations of TopAct or N2G typically share the same linguistic phrases, such as "used to" for the first case of TopAct and "CSV" for the first case of N2G. However, the selected words with our method do not appear as such lexical-level phrases; instead, the group of them illustrates a concise topic. This difference highlights the motivation of our research to find discourse-level explanations.

Our method generates more reasonable explanations than that of TopAct and N2G. Table 2 reports the percentage of learned sparse features that are successfully explained, and we group them by those that have been activated from the validation set or overall. We observe that many learned features haven't been reasonably explained with TopAct or N2G because not enough patterns have been activated on the validation set, which is one of the drawbacks of relying on activating input text for generations. One may argue that we can collect activated spans from the training set. However, these activated patterns can be significantly biased, as the sparse autoencoder is supposed to overfit the training set (Tom & Chris, 2023). Preparing a large validation set to ensure each learned sparse feature collects enough activation spans weakens the usability of these methods again. Even only considering the learned features that have been activated on the validation set, the proposed method shows a stronger explainable rate than the baselines. It is not surprising that N2G actually provides worse raw generations than TopAct, as we found evidence[3] that N2G shows a stronger preference for lexical patterns than TopAct, even if they are fake ones. These observations showcase the challenge of interpreting the discourse-level meanings behind the learned sparse features.

Table 2: Explanation rates of learned sparse features on the features only activated validation set or overall features.

Method	Explanation Rate	
	Activated	Overall
TopAct	59.16	23.17
N2G	38.79	15.13
Ours	67.39	66.98

4.3 USING EXPLAINED FEATURES FOR DOWNSTREAM TASKS

This section considers jailbreak defense as a downstream application to utilize our explained features. Our goal is to defend jailbreak attacks while keeping its helpfulness in responding to normal queries. We choose this task because of its generalizability across different scenarios that need to deploy LLMs. Also, existing defense strategies haven't shown practical utility due to their poor effectiveness or unbearable latency.

4.3.1 EXPERIMENTAL DESIGNS

We leverage two benchmarks to evaluate our downstream task performance. In specific, Salad-Bench (Li et al., 2024) is introduced to evaluate the safety of LLMs, and MT-Bench (Zheng et al., 2023) is applied to evaluate their general helpfulness. Two categories of the defense strategies that do *not* require any training

[3]For example, one sparse feature whose raw explanation of TopAct is "6th century (via History Magazine). Before that"; "Prior to Chomsky's work,"; and "Reference [2]: Before the GPS,".It is clear that this feature captures "referring related works". However, N2G simplifies them to "Before that"; "Prior to [MASK]omsky's work"; and "Before [MASK] GPS,", which obviously changes the meaning and concentrates on some trivial patterns, i.e., "Before" and "Prior to".

Table 3: Defending Mistral-7b-Instruct from jailbreak attacks without model training. The Salad-Bench reports the attack success rate (ASR) to illustrate the effectiveness of different models to prevent jailbreak attacks, while the MT-Bench shows its automatic scoring results on the helpfulness of general user queries.

Category	Method	**Salad-Bench** (Safety)		**MT-Bench** (Helpful)	
		ASR (↓)	**Time (↓)**	**Score (↑)**	**Time (↓)**
w/o Defense		81.6	1.0x	6.5	1.0x
Perturbation	Random Patch	80.6	4.9x	3.8	1.6x
	Random Insert	79.4	6.5x	3.7	1.6x
	Random Swap	73.8	5.6x	3.0	1.6x
	Self-Robustness	16.2	6.9x	5.3	**16.9x**
Prompting	SafePrompt	79.0	1.0x	6.5	1.0x
	XSafePrompt	77.8	0.9x	6.1	0.9x
	Self-Reminder	73.0	0.9x	6.3	0.9x
SAE Steer (Ours)	Erase Harmful (EH)	81.0	1.0x	5.9	1.0x
	Aware Secuirty (AS)	73.2	0.8x	6.0	0.9x
	EH + AS	72.8	0.8x	5.9	0.9x

datasets are considered as the baseline methods, where the *perturbation-based* methods include Random Patch/Insert/Swap (Robey et al., 2023) and Self-Paraphrase (Cao et al., 2023), and the *prompting-based* methods include SafePrompt/XSafePrompt (Deng et al., 2023), and Self-Reminder (Xie et al., 2023). Since most of the perturbation-based baselines are time-consuming, we randomly select 10% of the samples to conduct a smaller test set for all our evaluations. Note that all baselines and our methods will not be trained on any data in this experiment. The attack success rate (ASR) on Salad-Bench, GPT-4o-mini evaluates MT-Bench scores, and the normalized consuming time are listed in Table 3.

To apply the proposed Amplification or Calibration techniques for jailbreak defense, we can consider three specific strategies. (1) Erase Harmful (EH) monitors whether any "*harmful*" features are activated when responding to user prompts, and *erase* them if so (i.e., Amplification with $\alpha = -1$). (2) Aware Security (AS) consistently activates those *safety* features during responding. (3) Applying both AS and EH strategies at the same time. Here, we follow the hazard taxonomy of Llama3-Guard (Llama Team, 2024) to judge whether each feature is harmful or not. Inspired by this hazard taxonomy, we manually craft a safeguarding taxonomy listing 7 categories to classify safety strategies. We prompt GPT-4o-mini to provide the harmfulness and safety labels for each learned feature by providing their explanations. To ensure labeling quality, we only take the learned features with the explainable label "yes" into account. As a result, for our method, 141 and 48 features are selected for the AS and EH strategies, respectively. For hyper-parameter β of AS, we grid search some numbers and find that 2.5 shows the best practice in balancing safety and overall helpfulness. Table 3 and Figure 3 report the results with our explanations and baseline explanations, respectively.

4.3.2 RESULTS

Sparse autoencoder can steer LLMs behavior during runtime. First of all, we can read from Table 3 that all perturbation-based defense strategies are not practical for real-world use, as they either significantly compromise overall helpfulness or introduce intolerable latency. In contrast, most prompting-based methods maintain general helpfulness but fail to defend against jailbreak attacks. The notable exception is the state-of-the-art baseline, Self-Reminder, which achieves safety and helpfulness within the same computing budget. Compared with them, our proposed sparse-autoencoder-based methods exhibit a strong jailbreak defense ability (Salad-Bench: 81.6 $\rightarrow$ 72.8) with only a minor reduction on helpfulness (MT-Bench: 6.5 $\rightarrow$ 6.0). The success of our method in such a challenging task provides a promising direction for other scenarios.

The key to preventing jailbreak attacks is not to forget harmful knowledge, but to enhance safety awareness. One interesting finding from our experiment is that the strategy of erasing harmful knowledge has no significant contribution to the jailbreak defense, contradicting our intuitive understanding of the jailbreak defense. The significant improvement of our Aware Security strategy for jailbreak defense actually aligns with the main idea of Self-Reminder – "remind ChatGPT to respond responsibly" (Xie et al., 2023). This finding can benefit future works for jailbreak defense for large language models.

We also apply the Aware Security strategy to the TopAct and N2G explanations and report their results in Figure 3. Only N2G shows a slight reduction in ASR versus the no-defense baseline. We have tuned β but cannot see a clear improvement. One possible reason is that their selected safety strategies are too lexical-level and fine-grained. Here is an example feature that has been annotated with a "Physical Defense Category" as its summary is "Locking mechanisms or security systems" with a raw explanation: "locks; locks; lock; have a two-stage lock; lock." To compare with, one of our method annotated with the same category has a summary of "Emergency response and location tracking" with a raw explanation "contact, phone, unit, accuracy, exact, burning, location, precise, details, smoke." These observations highlight our motivation to explain discourse-level features.

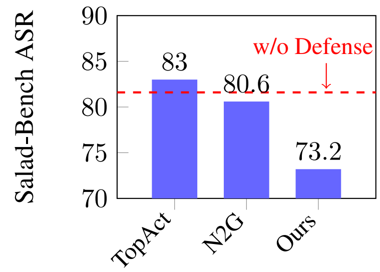

Figure 3: Applying Aware Security for jailbreak defense based on explanations from different methods.

5 Related Works

Modern large language models have shown promising text-generation abilities, prompting researchers to explore their internal mechanisms. One approach (Belinkov et al., 2018; Jawahar et al., 2019; Rogers et al., 2021) develops contrastive datasets to probe hidden states for specific features, but it is limited by the polysemantic nature of neurons (Elhage et al., 2022; Olah et al., 2020), making the explanations non-concise and difficult to apply in downstream tasks. To overcome this, researchers (Bricken et al., 2023) propose learning orthogonal basis vectors to better understand LLMs. Early works (Beren & Black, 2022; Wu et al., 2024) applied singular vector analysis to identify concise, interpretable directions in neuron activations. Soon after, sparse autoencoders (Bricken et al., 2023; Cunningham et al., 2023) were introduced, allowing for a more flexible settings. Sparse autoencoders, initially used to analyze image data (Olshausen & Field, 1997; Makhzani & Frey, 2013), are now being applied to LLMs. Researchers from Anthropic (Bricken et al., 2023) and EleutherAI (Cunningham et al., 2023) demonstrated that activations from smaller models like GPT-2 and Pythia yield highly interpretable features. Subsequent studies showed these features help interpret model behaviors in tasks like indirect object identification (Makelov, 2024), translation (Dumas et al.), and circuit detection (Marks et al., 2024). Recent works (Templeton et al., 2024; Gao et al., 2024; Lieberum et al., 2024) confirm this technique's success with larger LLMs. Our study follows this path, and advances by developing a method for generating discourse-level explanations to steer LLM representations.

6 Conclusions

This study steps a solid stamp toward understanding and steering LLM representations in the wild. Our theoretical analysis first reveals a frequency bias between discourse and linguistic features learned by sparse autoencoders. To eliminate this bias, we propose seeking words from a fixed vocabulary set and designing a mutual-information-based objective to ensure the collected words capture the features' meanings. Additionally, we demonstrate that our steering strategies effectively enhance the safety of LLMs using our mutual-information-based explanations, while baseline methods fail to achieve the same. Overall, this study underscores the importance of discourse-level explanations in effectively controlling LLM behavior.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*, 2018.

Beren and Sid Black. The singular value decompositions of transformer weight matrices are highly interpretable. 2022.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2, 2023.

David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pp. 113–120, 2006.

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.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

James Campbell, Richard Ren, and Phillip Guo. Localizing lying in llama: Understanding instructed dishonesty on true-false questions through prompting, probing, and patching. *arXiv preprint arXiv:2311.15131*, 2023.

Bochuan Cao, Yuanpu Cao, Lu Lin, and Jinghui Chen. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv preprint arXiv:2309.14348*, 2023.

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

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*, 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Clément Dumas, Veniamin Veselovsky, Giovanni Monea, Robert West, and Chris Wendler. How do llamas process multilingual text? a latent exploration through activation patching. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Alex Foote, Neel Nanda, Esben Kran, Ioannis Konstas, Shay Cohen, and Fazl Barez. Neuron to graph: Interpreting language model neurons at scale. *arXiv preprint arXiv:2305.19911*, 2023.

Pedro Freire, ChengCheng Tan, Adam Gleave, Dan Hendrycks, and Scott Emmons. Uncovering latent human wellbeing in language model embeddings. *arXiv preprint arXiv:2402.11777*, 2024.

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.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. *arXiv preprint arXiv:2402.05044*, 2024.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. Webglm: Towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4549–4560, 2023.

AI @ Meta Llama Team. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

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

Aleksandar Makelov, Georg Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

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*, 2024a.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

RyokoAI. Sharegpt dataset. 2023.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Mark Steyvers and Tom Griffiths. Probabilistic topic models. In *Handbook of latent semantic analysis*, pp. 439–460. Psychology Press, 2007.

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`.

Henighan Tom and Olah Chris. Dictionary learning worries. `https://transformer-circuits.pub/2023/may-update/index.html#dictionary-worries`, 2023.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models, 2024.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36, 2024.

Xuansheng Wu, Wenlin Yao, Jianshu Chen, Xiaoman Pan, Xiaoyang Wang, Ninghao Liu, and Dong Yu. From language modeling to instruction following: Understanding the behavior shift in llms after instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2341–2369, 2024.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12): 1486–1496, 2023.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

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.

A SCALING UP WITH MACHINE ANNOTATORS

We build on recent progress in automated interpretation (Bills et al., 2023; Chaudhary & Geiger, 2024; Gao et al., 2024; Lieberum et al., 2024) by utilizing advanced large language models to replicate human annotators in producing high-level interpretations. This approach allows us to leverage machine annotators, enabling us to scale our methods for analyzing the entire model and yielding more robust results.

We employ GPT-4o-mini [4] as our machine annotator. Our experiments utilize the gpt-4o-mini-2024-07-18 model with a hyper-parameter temperature=0 for greedy decoding. For each response, we allow a maximum of 1024 tokens. To ensure the quality of automatic annotation, we design our prompting template with both the role-playing strategy and presenting in-context examples. We list our prompting template for our word-list-based explanation summarization and the explainability judgment as follows.

A.1 TEMPLATE 1

We directly append the words to this template to annotate the summary of the raw explanations with 10 selected words from our proposed method.

```
System: You are studying a neural network. Each neuron looks for one
particular concept/topic/theme/behavior/pattern. Look at some words
the neuron activates for and guess what the neuron is looking for. Pay
more attention to the words in the front as they supposed to be more
correlated to the neuron behavior. Don't list examples of words and keep
your summary as detail as possible. If you cannot summarize most of the
words, you should say "Cannot Tell."

User: accommodation, racial, ethnic, discrimination, equality, apart,
utterly, legally, separately, holding, implicit, unfair, tone.
Agent: Social justic and discrimination.

User: B., M., e., R., C., OK., A., H., D., S., J., al., p., T., N., W.,
G., a.C., or, St., K., a.m., L..
Agent: Cannot Tell.

User: Scent, smelled, flick, precious, charm, brushed, sealed, smell,
brace, curios, sacred, variation, jewelry, seated.
Agent: Perception of scents and precious objects.

User: BP, HR, RR, O2 Sat, T, Ht, UO, BMI, BSA.
Agent: Medical measurements in emergency rooms.

User: actual, literal, real, Really, optical, Physical, REAL,
virtual, visual.
Agent: Perception of reality.

User: Go, Python, Java, c++, python3, c#, java, Ruby, Swift, PHP.
Agent: Morden programming language.

User: 1939-1945, 1945, 1942, 1939, 1940, 1941.
```

[4] https://platform.openai.com/docs/guides/gpt

```
Agent: Years of the second world war.

User: 1976, 1994, 1923, 2018, 2014, 1876, 1840.
Agent: Cannot Tell.

User:
```

A.2 TEMPLATE 2

Once we collect the summary of the raw explanation with the previous prompt, we then call the machine annotator again in a separated thread to evaluate whether the summary is hallucinated or not by using the following prompting template, where the placeholders "Summary" and "Raw Explanation" will be filled with real data. Note that if the machine annotator gives "Cannot Tell" as its prediction in the summarization stage, we will directly set the judgment for this task as "No".

```
System: You are a linguistic expert. Analyze whether the words well
represent the concept/topic/theme/pattern. Organize your final decision
in the format of "Final Decision: [[ Yes/Probably/Maybe/No ]]".

User: Concept/Topic/Theme/Pattern: {Summary}.
Words: {Raw Explanation}.
```

Since baseline explainers (TopAct and N2G) consider N-gram spans as raw explanations, we found that the previous word-list-based prompting template leads a poor performance for their interpretability. Thus, we followed the strategies before to define the following text-span-based prompting templates. Here, the in-context examples of text spans are collected from previous work (Bricken et al., 2023).

A.3 TEMPLATE 3

Similar to using Template 1 to summarize our extracted raw explanations, we append the extracted text spans from TopAct or N2G to this template. Note that we numerate each extracted span with a unique index.

```
System: You are studying a neural network. Each neuron looks for one
particular concept/topic/theme/behavior/pattern. Look at some spans the
neuron activates for and guess what the neuron is looking for. Pay more
attention to the [last few words] of each spans in the front as they
supposed to be more correlated to the neuron behavior. Ignore the [MASK]
patterns in the spans. Don't list examples of spans and keep your summary
as detail as possible. If you cannot summarize most of the spans, you
should say "Cannot Tell."

User: Span 1: w.youtube.com/watch?v=5qap5aO4z9A
Span 2: youtube.come/yegfnfE7vgDI
Span 3: {'token': 'bjXRewasE36ivPBx
Span 4: /2023/fid?=0gBcWbxPi8uC
Agent: Base64 encoding for web development.

User: Span 1: cross-function[MASK]
Span 2: cross-function
Span 3: [MASK][MASK] cross-function\n
Agent: Particular phrase 'cross-function'.
```

```
User: Span 1: novel spectroscopic imaging platform
Span 2: and protein evolutionary network modeling
Span 3: reactions-centric biochemical model
Span 4: chaperone interaction network
Agent: Biological terms.

User: Span 1: is -17a967
Span 2: what is 8b8 - 10ad2
Span 3: 83 -11111011001000001011
Span 4: is -c1290 - -1
Agent: Synthetic math: Arithmetic, numbers with small digits,
in unusual bases.

User:
```

A.4 TEMPLATE 4

We evaluate the interpretability of the baseline methods using almost the same prompting text, where we only change the phrase from "word" to "span" to fit the format of raw explanations from the baseline explainers. Again, each text span is numerated with unique ids.

```
System: You are a linguistic expert. Analyze whether the text spans well
represent the concept/topic/theme/pattern. Organize your final decision
in the format of "Final Decision: [[ Yes/Probably/Maybe/No ]]".

User: Concept/Topic/Theme/Pattern: {Summary}.
Spans: {Raw Explanation}.
```