Enhancing Automated Interpretability with Output-Centric Feature Descriptions

Yoav Gur-Arieh[1] **Roy Mayan**[1*] **Chen Agassy**[1*] **Atticus Geiger**[2] **Mor Geva**[1]
[1]Blavatnik School of Computer Science and AI, Tel Aviv University
[2]Pr(Ai)²R Group
{yoavgurarieh@mail,roymayan@mail,chenagassy@mail,morgeva@tauex}.tau.ac.il, atticusg@gmail.com

Abstract

Automated interpretability pipelines generate natural language descriptions for the concepts represented by features in large language models (LLMs), such as *plants* or *the first word in a sentence*. These descriptions are derived using *inputs* that activate the feature, which may be a dimension or a direction in the model's representation space. However, identifying activating inputs is costly, and the mechanistic role of a feature in model behavior is determined both by how inputs cause a feature to activate and by how feature activation affects *outputs*. Using steering evaluations, we reveal that current pipelines provide descriptions that fail to capture the causal effect of the feature on outputs. To fix this, we propose efficient, output-centric methods for automatically generating feature descriptions. These methods use the tokens weighted higher after feature stimulation or the highest weight tokens after applying the vocabulary "unembedding" head directly to the feature. Our output-centric descriptions better capture the causal effect of a feature on model outputs than input-centric descriptions, but combining the two leads to the best performance on both input and output evaluations. Lastly, we show that output-centric descriptions can be used to find inputs that activate features previously thought to be "dead".

1 Introduction

Understanding how language models represent concepts in a real-valued vector space has long been a central challenge in NLP (Mikolov et al., 2013; Karpathy et al., 2015; Bau et al., 2019; Mu and Andreas, 2020; Dai et al., 2022; Park et al., 2024a). Recent efforts to scale this process use automated interpretability pipelines, where large language models (LLMs) describe the concepts encoded by features, i.e., small model components such as neurons or directions in activation space, *based on inputs that activate them* (Bills et al., 2023; Bricken et al., 2023; Paulo et al., 2024; Choi et al., 2024). However, despite its wide adoption, solely relying on the inputs activating a feature to describe it has practical limitations and theoretical pitfalls.

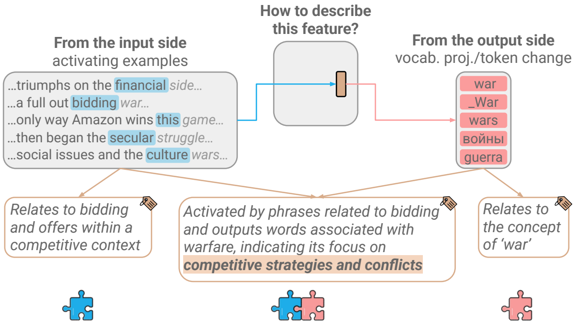


Figure 1: We posit that a faithful description of a feature should consider both model inputs that activate it (**left**, marked words cause the highest activations) and the effect it introduces to the model's outputs (**right**).

First, given the large corpora modern LLMs are trained on, obtaining these examples can be costly and nearly impossible in cases when features are described by data instances that are not publicly available. This practical limitation increases the compute and data needed for automated interpretability. Second, the concept represented by a feature is determined by the causal role of that feature in model behavior, namely, how model inputs cause the feature to activate and how a feature causes model outputs to change (Mueller et al., 2024). Using only inputs to characterize a feature is ungrounded in the causal mechanisms driving model behavior, which introduces pitfalls. For example, different datasets can lead to inconsistent feature descriptions (Bolukbasi et al., 2021) or to classifying features as "dead" due to lack of activation (Gao et al., 2024; Templeton et al., 2024). Last, a common use of feature descriptions is controlling model behavior through "steering", i.e., stimulating a feature to control the model's outputs (Upchurch et al., 2017; Li et al.,

---

[*] Equal contribution

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5757–5778
July 27 - August 1, 2025

2023; Rimsky et al., 2024; Templeton et al., 2024; O'Brien et al., 2024a). Therefore, good feature descriptions for steering should be output-centric.

To overcome these limitations, we propose two output-centric methods for enhancing automated interpretability pipelines (see Figure 1 for illustration). The first method, called `VocabProj`, uses the prominent tokens in the projection of a feature to the model's vocabulary space (Geva et al., 2022b; Bloom and Lin, 2024). The second method, called `TokenChange`, considers the tokens whose probabilities in the model's output distribution change the most when the feature is amplified. Notably, these methods are substantially more computationally efficient than generating descriptions based on activating inputs; `VocabProj` requires a single matrix multiplication, and `TokenChange` involves running the model on a few inputs.

We compare the descriptions generated by these methods with those generated based on maximum activating inputs (dubbed `MaxAct`) using two evaluations: *input-based* and *output-based* (see Figure 2 for illustration). The input-based evaluation assesses how accurately a description identifies what triggers the feature, whereas the output-based evaluation measures how effectively the description captures the causal impact of the feature's activation on the model's output.

Experiments over neuron-aligned and sparse autoencoder (SAE) features from both the residual and MLP layers of multiple LLMs reveal substantial differences between the methods and the descriptions they yield. While `MaxAct` typically outperforms `VocabProj` and `TokenChange` on the input-based evaluation, it is generally worse in capturing the feature's effect on the model's generation. Moreover, the gap between `MaxAct` and `VocabProj` in describing the inputs activating a given feature is sometimes small, suggesting that the latter can serve as a cheap replacement in such cases. Last, ensembles of the three methods consistently achieve the best performance across both evaluations, providing strong empirical evidence for the benefits of incorporating output-centric methods into automated interpretability pipelines.

Further analysis sheds light on those benefits. We observe that descriptions generated by output-centric methods are often abstractions of their input-centric counterparts, and that the composition of the input- and output-centric descriptions of a feature can in some cases provide a new meaning (e.g. Figure 1). Additionally, experiments with Gemma-2 SAEs show that output-centric methods can be used to efficiently discover inputs that activate "dead" features, for which no activating inputs had previously been identified.

To summarize, our work makes the following contributions: (a) we propose a two-faceted evaluation framework for feature descriptions, examining them through complementary input and output lenses (b) we highlight key drawbacks of using `MaxAct`, the common method used today in automated interpretability pipelines, to obtain feature descriptions in LLMs, (c) we propose output-centric methods to mitigate these limitations, (d) our experiments demonstrate the effectiveness of each approach and that their combination yields more faithful feature descriptions, (e) our analysis provides insights into the benefits in combining input- and output-centric methods. By producing more faithful and complete feature descriptions, our approach can enhance downstream applications such as model editing, machine unlearning, and circuit analysis (e.g., Wu et al., 2023; Farrell et al., 2024a; Marks et al., 2025). We release our code and generated feature descriptions at `https://github.com/yoavgur/Feature-Descriptions`.

2 Problem Setup

We focus on the problem of automatically describing atomic units of computation in LLMs called *features*. As the exact nature of features is a hotly debated topic, we adopt the general framework of Geiger et al. (2024a) which we limit to real-valued features. Let $\mathcal{M}$ be our target LLM. Any hidden vector $\mathbf{v} \in \mathbb{R}^d$ in $\mathcal{M}$ can be transformed with an invertible *featurizer* $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^k$ that maps the vector into a space of k features. A single feature $f \in \mathbb{R}^k$ is simply a one-hot encoding which can be vectorized using $\mathbf{v}_f = \mathcal{F}^{-1}(f)$. This framework supports a variety of features, including neurons (axis-aligned dimensions) in MLPs (Geva et al., 2022b), sets of orthogonal directions (Geiger et al., 2024b; Huang et al., 2024; Park et al., 2024b), sparse linear features from SAEs (Bricken et al., 2023; Templeton et al., 2024; Huben et al., 2024), or even non-linear features, e.g. "onion" representations with a magnitude-based features (Csordás et al., 2024).

During inference, the LLM constructs the vector $\mathbf{v}$ from the input, which can then be passed through $\mathcal{F}$ to determine the activation for each feature $\mathcal{F}(\mathbf{v})$. The possible values for activations are a result of the feature space, e.g. SAE features produced with a ReLU only have positive activations.

In this work, we consider the problem of automatically labeling the concept represented by a feature f. Namely, producing a human-understandable description text s_f of the feature f. Importantly, we want the method producing s_f to be scalable, i.e. automatic and efficient, such that it can be integrated into large-scale pipelines that interpret millions of features in LLMs. This additional requirement excludes approaches that rely, for example, on manual human labeling.

A key question that arises is how to evaluate whether a description faithfully describes its corresponding feature. Here we observe that describing a feature is practically *a two-faceted problem*; one can describe what inputs activate the feature, i.e. what inputs yield high feature activations, but they can also describe what this feature promotes in the model's output. Consider for example the feature illustrated in Figure 1. The input side indicates that the feature activates mainly on competitive financial and business related sentences. Conversely, the output side shows that the feature amplifies the concept of war when activated. Only when considering the two sides together we see that the feature promotes the concept of war in social and business related scenarios, e.g., *trade war*, *bidding war*, and *culture war*. Notably, this formulation was also discussed in prior works; Geva et al. (2021, 2022a,b) characterized MLP as key-value memories that promote specific concepts, and Antverg and Belinkov (2022); Huang et al. (2023) contended the importance of differentiating between the information encoded by the feature versus used by the model.

Despite the dual nature of this problem, existing automated interpretability pipelines (e.g., Bills et al., 2023; Paulo et al., 2024; Choi et al., 2024) have focused on one side of the problem. Namely, describing the inputs that activate the feature, while disregarding the feature's influence on the model's output. For example, Huang et al. (2023) showed that neurons interpreted by Bills et al. (2023) lack causal influence on the concepts expressed in their generated descriptions. Therefore, we offer a more holistic approach, accounting for both the input and output of the model.

3 Evaluation of Feature Descriptions

We propose to evaluate how faithful a description is to its corresponding feature with the following complementary metrics, illustrated in Figure 2.

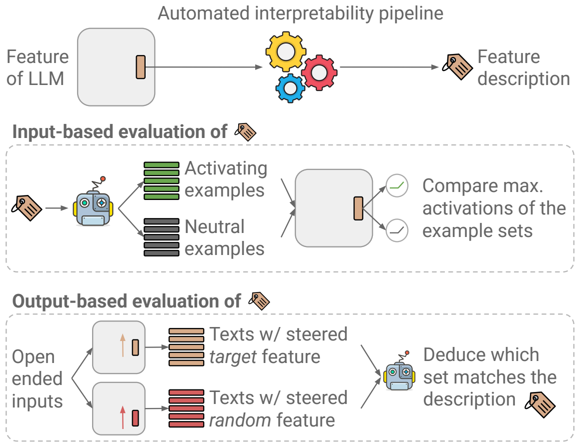


Figure 2: Illustration of our feature description evaluation, considering the description's faithfulness with respect to both the input (middle panel) and output (lower panel) of the model.

Input-based Evaluation Following Huang et al. (2023); Caden Juang et al. (2024), we evaluate how well the description captures the inputs triggering the feature. Given a feature f, we feed its description s_f generated by some method into an LLM, which is tasked to generate two sets of k examples each: *activating* and *neutral*. These examples are expected and not expected to activate f according to s_f, respectively (see §A for examples and details regarding prompts). We then pass the generated examples through $\mathcal{M}$ and obtain f's activation for each example, calculated as the max activation over all token positions in that example. We take the max over all token positions since it's reasonable to expect f to be activated highly even for just a single token, and not at all for the rest, following prior work that treats strong localized activation as meaningful (Bills et al., 2023; Choi et al., 2024; Paulo et al., 2024; Voita et al., 2024). Let $\bar{m}_{\text{activating}}$ and $\bar{m}_{\text{neutral}}$ be the mean activations obtained for the activating and neutral examples, respectively. The description s_f is considered faithful if the mean activation for the activating examples exceeds that of the neutral examples, namely:

$$\bar{m}_{\text{activating}} > \bar{m}_{\text{neutral}}$$

This evaluation is similar to those implemented in existing automated pipelines, which essentially measure *how accurately the description captures the inputs that activate the feature*.

Output-based Evaluation To assess how faithful s_f is with respect to f's influence on the model's outputs, we evaluate s_f against outputs generated by $\mathcal{M}$ when steering f versus when steering another feature f'. Concretely, we feed $\mathcal{M}$ open-ended prompts, such as "`<BOS> I think`" (Chalnev et al., 2024), and let the model generate n tokens three times – one time while amplifying f and two other times by amplifying two different random features f' and f''. Amplification of a feature is done by clamping its activation to a high value m (Templeton et al., 2024). Since finding an effective yet not destructive amplification level is challenging (Bhalla et al., 2024; Templeton et al., 2024), we run each input with varying levels of amplification while fixing the KL-divergence between the outputs of the steered model and the non-steered model (Paulo et al., 2024), as calculated on a single next token prediction, averaged over all open ended prompts. This way we generate three sets of texts $\mathcal{T}_f$, $\mathcal{T}_{f'}$ and $\mathcal{T}_{f''}$. Next, we feed s_f concatenated with $\mathcal{T}_f$, $\mathcal{T}_{f'}$ and $\mathcal{T}_{f''}$ to a judge LLM (see justification in §E), and task it to indicate which of the three sets matches s_f. The description s_f is faithful if the LLM selects $\mathcal{T}_f$. Namely, we evaluate *how well the description captures the feature's impact on the model's output*. For details, example generations and prompts used, see §A.

4 Interpretability Methods

We describe the methods used for automatically describing features in LLMs. These include the input-centric method prevalent today, two output-centric methods that describe a feature f using its corresponding vector $\mathbf{v}_f$, and their ensembles.

Max Activating Examples (`MaxAct`) Using the inputs that maximally activate a given feature to understand its function has been used extensively (Dalvi et al., 2018; Na et al., 2019; Bolukbasi et al., 2021). More recently, this method has been widely adopted and refined for automatically interpreting features at scale (Bills et al., 2023; Bricken et al., 2023; Paulo et al., 2024; Choi et al., 2024; He et al., 2024a; Huben et al., 2024). The method involves collecting feature activations in $\mathcal{M}$ across a large dataset. For each feature, k examples are sampled from the dataset, prioritizing those with the highest activations, along with some examples from other activation quantiles (Bricken et al., 2023). These examples are then fed to an explainer model, which is tasked with generating a description of the feature by the examples that activate it.

Vocabulary Projection (`VocabProj`) Building on Geva et al. (2021, 2022a,b), we propose to view the feature f as an update to the model's output distribution. To interpret f's contribution, we compute the feature vector $\mathcal{F}^{-1}(f) = \mathbf{v}_f \in \mathbb{R}^d$ and project it to the vocabulary space to obtain a vector of logits $\mathbf{w} \in \mathbb{R}^{|\mathcal{V}|}$ such that:

$$\mathbf{w} = W_U \texttt{LayerNorm}(\mathbf{v}_f)$$

where $\mathcal{V}$ is $\mathcal{M}$'s vocabulary, `LayerNorm` is the final layer norm, and $W_U \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the model's unembedding matrix. We then examine the tokens corresponding to the top- and bottom-scoring entries in $\mathbf{w}$, interpreting them as the tokens most promoted or suppressed, respectively. These tokens are then fed to an explainer model that generates a description for the feature. For more details and other variants of this method, see §B.1.

Token Change (`TokenChange`) This method describes the tokens whose logits in the model's output were most affected by amplifying the feature. Specifically, we pass k random prompts sampled from some dataset through the model and collect the output logit values for each token position. Next, the feature is clamped to activation value m, and we collect the new logit values (Templeton et al., 2024). We then calculate the mean change in logit value per token across all positions and prompts. The list of tokens most affected by amplifying the feature is provided to an explainer model, which generates a description for the feature.

While both `VocabProj` and `TokenChange` are output-centric methods, `VocabProj` is correlative and `TokenChange` causally intervenes in the model's generation.

Ensembles To capture both the input and output sides of a feature, we propose combining the above approaches in two ways: (a) `Ensemble Raw`: the raw data used by the methods is concatenated and fed to the explainer model. For example, in `Ensemble Raw (MaxAct+VocabProj)` we would feed the explainer model the activating examples and top tokens in the vocabulary projection. (b) `Ensemble Concat`: the description is simply a concatenation of the descriptions generated by the methods. We also attempted to summarize the descriptions by the different methods with an LLM to produce a more cohesive description, but these ensembles performed worse across the board.

5 Experiments

In this section, we evaluate the above methods on our input- and output-based evaluations. Additional human evaluations are reported in §E.

5.1 Experimental Setting

Features We analyze both features learned through SAEs and neurons in MLP layers, covering four LLMs of different sizes and families: Gemma-2 2B (Team et al., 2024b), Llama-3.1 8B and Llama-3.1 8B Instruct (Dubey et al., 2024), and GPT-2 small (Radford et al., 2019). For Gemma-2, Llama-3.1 and GPT-2 small, we evaluate descriptions of SAE features trained on residual stream and MLP layers: Gemma Scope 16K and 65K (Lieberum et al., 2024), Llama Scope 32K (He et al., 2024b), and OpenAI SAE 32K and 128K (Gao et al., 2024). The activation function used by Gemma Scope is JumpReLU (Rajamanoharan et al., 2024), while both Llama Scope and OpenAI SAE use TopK-ReLU (Makhzani and Frey, 2014). We randomly sample $n = 40$ features per layer from every SAE, resulting in a total of 4,160 features for Gemma-2, 2,560 for Llama-3.1 and 2,880 for GPT-2 small. For Llama-3.1 Instruct we inspect a sample of $n = 80$ MLP features per layer, with 2,560 features in total.

Description Generation We use the methods described in §4 and generate descriptions for each feature, using GPT-4o mini (Hurst et al., 2024) as our explainer model to ensure consistency with descriptions from Neuronpedia (Lin and Bloom, 2023) and Transluce (Choi et al., 2024). For `MaxAct`, we utilize the publicly available feature descriptions from these repositories. To validate these descriptions are comparable to those generated by us, we sampled 1,080 features and found their descriptions match those we generate for `MaxAct` (see §B.3).

When generating ensembles from raw data (`Ensemble Raw`), we rely on feature activation data from these same sources, using the top five activating sentences to keep in line with existing methods. Notably, Transluce generated descriptions for Llama-3.1 8B Instruct through a more complex process than `MaxAct` (Choi et al., 2024), creating multiple descriptions from activating examples and selecting the best one using simulation scoring (Bills et al., 2023). For clarity, we refer to this method as `MaxAct++` and generate the `MaxAct` descriptions for Llama-3.1 8B Instruct ourselves using the feature activation data from Transluce.

For `VocabProj` and `TokenChange`, we pass the top and bottom t tokens to the explainer model GPT-4o mini (see prompts in §B.2). We set $t = 50$ for `VocabProj` and $t = 20$ for `TokenChange`. For `TokenChange` we use $k = 32$ random prompts of 32 tokens each from The Pile (Gao et al., 2020).

Description Evaluation For the input-based evaluation, we instruct Gemini 1.5 Pro (Team et al., 2024a) to generate five activating and five neutral sentences with respect to a given feature description. For the output-based evaluation, we prompt the model with three open-ended prompts, letting it generate up to 25 tokens while clamping the feature's activation value to m for all token positions. For each prompt, we run the model four times with increasing clamping values, making the generations progressively more affected by the feature's output. This process results in 12 text generations for each of the sets $\mathcal{T}_{\mathbf{v}_f}$, $\mathcal{T}_{\mathbf{v}'_f}$, and $\mathcal{T}_{\mathbf{v}''_f}$, which we provide to GPT-4o mini (Hurst et al., 2024) as a judge (see §A for more details and exact prompts). We select this model to minimize costs, given the lengthy prompts induced by the text sets.

5.2 Results

Table 1 shows the results averaged across layers, and Figure 3 provides a breakdown for layer groups for features from Gemma-2 and both Llama-3.1 models. Similar trends are shown for all other features in §C.

Combining input- and output-centric methods yields better feature descriptions Table 1 shows that across all models and feature types, `MaxAct` outperforms `VocabProj` and `TokenChange` on the input-based evaluation and vice versa on the output-based evaluation, often by large margins of up to 15%-30%. This also holds for `MaxAct++` on Llama-3.1 8B Instruct, demonstrating that input- and output-centric methods capture different feature information. Second, ensembling input- and output-centric methods boosts performance on both evaluations, with the ensembles combining all three methods consistently outperforming the single-methods. For instance, for Gemma-2 the ensembles yielded an improvement of 6%-10% over the next best single-method on both metrics. One exception to this trend is `MaxAct++`, which performs better than all other methods on the input metric, with `Ensemble Raw` in close second. This is probably due to `MaxAct++` being optimized for describing what activates a given feature. Overall, this

	Gemma-2 Res. SAE		**Gemma-2 MLP SAE**		**Llama-3.1 Res. SAE**		**Llama-3.1 Inst. MLP**	
	Input	Output	Input	Output	Input	Output	Input	Output
`MaxAct`	56.6 ± 2.2	49.2 ± 2.2	50.4 ± 2.2	35.1 ± 2.1	30.3 ± 2.7	71.8 ± 2.6	85.6 ± 1.4	36.9 ± 1.9
`MaxAct++`	-	-	-	-	-	-	**89.8** ± 1.2	39 ± 1.9
`VocabProj`	50.1 ± 2.2	56.5 ± 2.2	20.9 ± 1.8	37.2 ± 2.1	18.2 ± 2.2	64.2 ± 2.8	71.2 ± 1.8	**45.8** ± 1.9
`TokenChange`	44.7 ± 2.2	54.9 ± 2.2	22.3 ± 1.8	40.3 ± 2.2	21.4 ± 2.4	72.0 ± 2.6	74 ± 1.7	43.8 ± 1.9
`EnsembleR (MA+VP)`	**66.9** ± 2.1	52 ± 2.2	**56.6** ± 2.2	38.6 ± 2.1	**36.9** ± 2.8	68.9 ± 2.7	86.7 ± 1.3	40.7 ± 1.9
`EnsembleR (MA+TC)`	**67** ± 2.1	61.9 ± 2.1	**56.4** ± 2.2	46.2 ± 2.2	**37.2** ± 2.8	68.0 ± 2.7	87.2 ± 1.3	41.7 ± 1.9
`EnsembleR (VP+TC)`	53.1 ± 2.2	63 ± 2.1	24.3 ± 1.9	46.6 ± 2.2	20.9 ± 2.3	67.4 ± 2.7	72.4 ± 1.7	**44.3** ± 1.9
`EnsembleR (All)`	**66.6** ± 2.1	**64.9** ± 2.1	**55.7** ± 2.2	**48.7** ± 2.2	**36** ± 2.8	71.2 ± 2.6	86.2 ± 1.3	41.8 ± 1.9
`EnsembleC (All)`	57.7 ± 2.2	**66.9** ± 2.1	31.6 ± 2.1	**49.9** ± 2.2	28.5 ± 2.6	**75.4** ± 2.5	84.9 ± 1.4	**44.6** ± 1.9

Table 1: Input- and output-based evaluation results of the methods and their ensembles, over different feature types and models, averaged across model layers, along with their respective 95% confidence intervals. For SAE features we take the average over features from SAEs of all sizes. We denote `MA` for `MaxAct`, `VP` for `VocabProj`, `TC` for `TokenChange`, and `EnsembleR` and `EnsembleC` for the raw and concatenation based ensembles.

input-output integration not only better describes the causal roles of features but also improves performance on the widely-used input-based evaluation.

Performance varies by layer and feature type Comparing the results for residual versus MLP features and neurons versus SAE features, we find that output-based performance is substantially lower for MLP features compared to residual features (reaching 45-50 points for MLP vs. ∼66 points for residual). This might be explained by the MLP layers introducing gradual changes to the residual stream (Geva et al., 2021, 2022b), potentially making them harder to steer. Additionally, output-based performance of `VocabProj` is worse in early layers but gradually improves, consistent with prior observations (Nostalgebraist, 2020; Geva et al., 2021; Yom Din et al., 2024).

`VocabProj` and `TokenChange` often provide efficient substitutes for `MaxAct` A major practical drawback of `MaxAct` is the computational cost required for comprehensively mapping the activating inputs of a feature. Considering the performance of `VocabProj`, `TokenChange`, and `EnsembleR (VP+TC)`, we observe that (a) they typically outperform `MaxAct` on the output-based evaluation, which is crucial for assessing the description's faithfulness to the feature's causal effect and its usefulness for steering, and (b) they often perform only slightly worse on the input-based evaluation, e.g. there's only a 3.5 point gap between `Ensemble Raw (VP+TC)` and `MaxAct` on residual stream SAE features in Gemma-2. These results suggest that `VocabProj` and `TokenChange`, which require only ≤2 inference passes, can often be a more efficient and sometimes higher-performing alternative to the widely-used `MaxAct` method. An analysis of the computational costs is in §D.

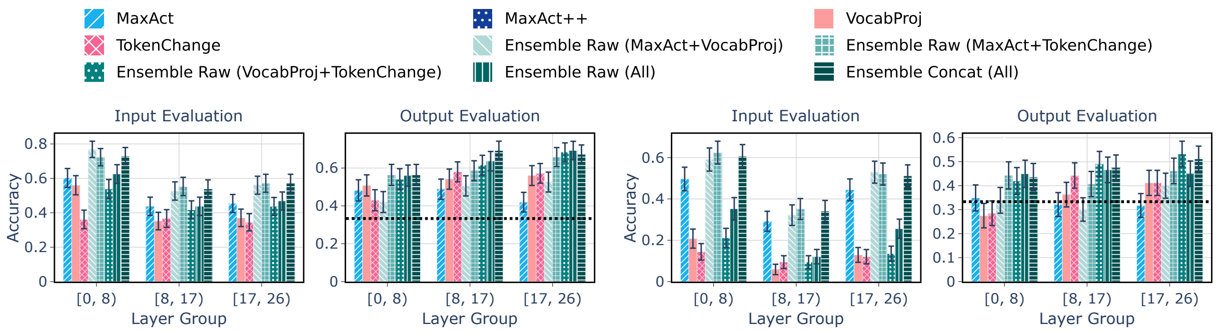
Description Format Affects Performance Comparing the top-performing ensembles, we observe that `Ensemble Raw` is generally better on the input-based evaluation while `Ensemble Concat` is consistently best on the output-side evaluation. We hypothesize that this could be due to the different description formats of the two ensembling approaches, i.e., concatenating raw outputs versus generated descriptions. For the input-based evaluation, a longer and more informative description may have a higher chance of enabling an LLM to generate sentences with at least one activating token, compared to a concise description. Similarly, a concise description could be matched to texts generated by the model more easily compared to a long and detailed description.

6 Analysis

In this section, we compare the feature descriptions obtained by `MaxAct`, `VocabProj` and `TokenChange` and analyze the utility in their combination.

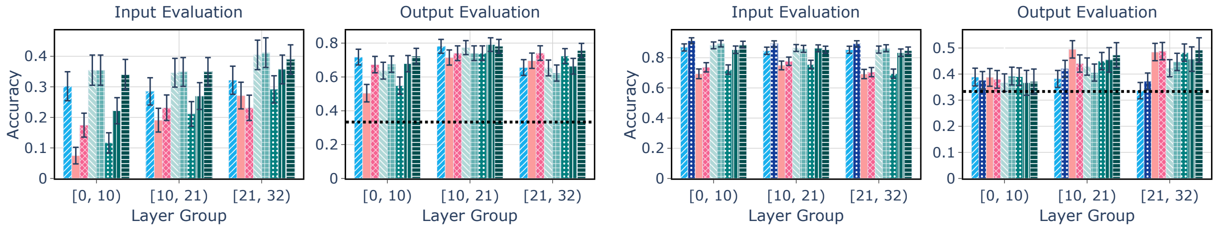
6.1 Qualitative Analysis

We manually analyze the descriptions by `MaxAct` and `VocabProj` for a random sample of 100 features from Gemma Scope 16K, 50 for the MLP layers and 50 for the residual stream. We exclude `TokenChange` here as we noticed that the descriptions it produces are often similar to those by `VocabProj` (see examples in §G). In the analysis, we consider descriptions that pass both our input- and output-based evaluations. We observed 4 main types of relations between the descriptions:



(a) Residual stream SAE features of width 65k from Gemma-2.

(b) MLP SAE features of width 65k from Gemma-2.



(c) Residual stream SAE features of width 32k from Llama-3.1.

(d) MLP features from Llama-3.1 8B Instruct.

Figure 3: Performance of the various methods on the proposed metrics, for Gemma-2 2B (upper row), Llama-3.1 8B (lower left), and Llama-3.1 8B Instruct (lower right). For the output metric, the baseline (dashed black line) is $1/3$ since the judge LLM picks between three sets of texts.

Relation	**Example feature** layer-type/id	**Description by `MaxAct`**	**Description by `VocabProj`**
Similar 41%	`3-MLP-16K/` `4878`	Terms and themes related to various genres of storytelling, particularly in horror, drama, and fantasy.	A blend of themes and genres commonly found in storytelling or media, with a specific focus on dramatic, horror, and suspenseful narratives.
Composition 23%	`19-MLP-16K/` `5635`	References to political events and milestones.	Concepts related to time measurement such as days, weeks, weekends, and months, indicating it likely pertains to scheduling or planning events.
Abstraction 23%	`21-RES-16K/` `10714`	Information related to bird species and wildlife activities.	Concepts related to birdwatching and ornithology, focusing on activities such as observing, spotting, and recording bird species in their natural habitats.
Different 13%	`19-MLP-16K/` `1450`	Mentions of notable locations, organizations, or events, particularly in various contexts.	Concepts related to self-reflection, purpose, and generalization in various contexts, focusing on the exploration of identity and overarching themes in literature or philosophy.

Table 2: Human evaluation results of descriptions by `MaxAct` and `VocabProj` for 100 SAE features from Gemma Scope, showing for each relation category the fraction of observed cases and the descriptions of an example feature.

- **Similar**: The tokens in the projection and are highly similar to the tokens in the activating examples, resulting in matching descriptions.
- **Composition**: The input- and output-centric descriptions refer to different aspects of the feature, while their composition provides a more holistic description of the feature.
- **Abstraction**: The tokens in the projection represent a more general or broad concept than the one observed in the activating examples.
- **Different**: The input- and output-centric descriptions refer to different aspects of the feature, which share no clear relation between them.

Table 2 shows the fraction of examples classified per category alongside representative feature descriptions. Overall, while input- and output-centric descriptions are often similar (41%), there are many cases where their composition provides a broader (23%) or more accurate (23%) description .

6.2 Reviving Dead Features

One drawback of describing features with `MaxAct` is the dependency on the dataset used to obtain activations (Bolukbasi et al., 2021). A particularly interesting case is the classification of “dead” features, which do not activate for any input from the dataset. Dead features can be prevalent (Voita et al., 2024; Gao et al., 2024; Templeton et al., 2024). For example, we observed they constitute up to 29% of the features in some SAEs in Gemma-2.

While dead features could potentially not represent meaningful features, it may be that the dataset used simply does not cover the "right" inputs for activating them. Here we conduct an analysis that shows that dead features can be "revived" (i.e. activated) with inputs crafted based on their `VocabProj` and `TokenChange` descriptions.

Analysis We sampled 1,850 SAE features from Gemma-2 2B equally distributed across layers and types (MLP / residual) and classified as "dead" based on Neuronpedia. For each feature, we create a set of candidate prompts for activating it by: (a) using the feature descriptions by `VocabProj` and `TokenChange` and letting Gemini generate 150 sentences that are likely to activate the feature, and (b) gathering the tokens identified by `VocabProj` and `TokenChange` and constructing 1,450 sequences of different lengths that randomly combine these tokens. Both the top and bottom tokens obtained using these methods could potentially activate the feature, as they might relate to concepts that the feature promotes or suppresses. We then feed all the generated prompts into the model and consider a feature as "revived" if any prompt successfully activated it. For implementation details, see §F.

Results The generated prompts successfully activated 9.1% (85) of MLP SAE features and 62% (491) of residual ones. In 12% (70) of cases, a feature was activated using an LLM-generated prompt, while 73% (423) were activated with a prompt composed of two tokens: '`<BOS>`' and a sampled token. Moreover, the revived dead features can often be easily interpreted using `VocabProj` and `TokenChange`, while considered faithful based on our output-based metric (see examples in §F). Overall, this demonstrates that output-centric methods can address potential oversights that may arise from focusing solely on activating inputs.

7 Related Work

Bills et al. (2023) introduced an automated interpretability pipeline that used GPT-4 to explain the neurons of GPT-2 based on their activating examples (`MaxAct`), while employing an input-based evaluation known as simulation scoring. This approach has become common practice for interpreting neurons and learned SAE features of LLMs at scale (Lin and Bloom, 2023; Cunningham et al., 2023; Bricken et al., 2023; Templeton et al., 2024; Gao et al., 2024; He et al., 2024a), which also extends to neuron description pipelines of visual models (Hernandez et al., 2022; Shaham et al., 2024; Kopf et al., 2024).

Recently, new methods for generating feature descriptions have been proposed, such as applying variants of activation patching (Kharlapenko et al., 2024), refining the prompt given to the explainer model (Paulo et al., 2024), and improving descriptions of residual feature activations via description selection (Choi et al., 2024) similarly to the algorithm by Singh et al. (2023). While all these prior works rely on input-centric, computationally intensive approaches, we propose output-centric efficient methods that require no more than two inference passes of the model. Furthermore, we show that combining input- and output-centric methods leads to improved overall performance.

More broadly, our work relates to growing efforts in understanding features encoded in neurons and SAE features. These include steering (Farrell et al., 2024b; Chalnev et al., 2024; O'Brien et al., 2024b; Templeton et al., 2024), circuit discovery (Marks et al., 2024; Makelov et al., 2024; Balcells et al., 2024), feature disentanglement (Huang et al., 2024; Cohen et al., 2024) and benchmarks like SAEBench.[1] However, evaluation of feature descriptions remains relatively underexplored. Rajamanoharan et al. (2024) evaluated latent interpretability for different SAE architectures using an input-centric approach which does not reflect downstream effect in model control. More recently, Paulo et al. (2024) have found negative correlation between multiple input-centric scoring methods and an intervention-based metric. Finally, Bhalla et al. (2024) concurrently evaluated feature descriptions in terms of their downstream effects on the model. However, they focus on evaluating methods for effectively steering models, as opposed to evaluating methods for generating descriptions.

8 Conclusion

While existing automated interpretability efforts describe features based on their activating inputs, we posit that describing a feature is a two-faceted challenge, requiring the comprehension of both its

---

[1] `https://www.neuronpedia.org/sae-bench/info`

activating inputs and influence on model outputs. To tackle this challenge at scale, we employ two evaluations – input-based and output-based – and propose two output-centric methods (`VocabProj` and `TokenChange`) for generating feature descriptions. Through extensive experiments we show that output-centric methods offer an efficient solution for automated interpretability, especially when geared towards model steering, and can substantially enhance existing pipelines which rely on input-centric methods.

Limitations

Although we observe clear trends in the results, the output-based evaluation is fairly noisy. We address this by sampling large numbers of features and using multiple prompts in the evaluation, but future work could focus on reducing this noise further and making the evaluation more efficient. Additionally, we find that the output-centric methods and ensembles are sensitive to the choice of prompt. Since generating feature descriptions using these methods is non-trivial and often involves long texts (especially for the ensembles), improving explainer model prompts to extract relevant information could potentially enhance performance. We also note that our input-based evaluation uses a binary threshold, which may oversimplify feature behavior. Nonetheless, it enabled us to efficiently identify trends across models and methods, and we leave refining this evaluation to future work.

Regarding the methods evaluated, while we focused on efficient approaches that can automatically scale to millions of features, exploring other methods, such as patching-based methods, could be valuable. Lastly, the output-centric methods we propose are tied to the model's vocabulary, which means they can only describe features that can be expressed with tokens from the vocabulary. These methods may struggle in describing features that are not easily or naturally expressed with words, such as positional features. For simplicity, we did not differentiate between whether concepts were being suppressed or promoted by a feature.

Acknowledgements

We thank the Transluce team, specifically Dami Choi, for sharing their neuron description pipeline data, as well as Johnny Lin from Neuronpedia for sharing their descriptions and model activations data. This work was supported in part by the Gemma 2 Academic Research Program at Google, the Edmond J. Safra Center for Bioinformatics at Tel Aviv University, a grant from Open Philanthropy, and the Israel Science Foundation grant 1083/24. Figures 1 and 2 use images from `www.freepik.com`.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Omer Antverg and Yonatan Belinkov. 2022. On the pitfalls of analyzing individual neurons in language models. In *International Conference on Learning Representations*.

Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. 2024. Evolution of sae features across layers in llms. *Preprint*, arXiv:2410.08869.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Usha Bhalla, Suraj Srinivas, Asma Ghandeharioun, and Himabindu Lakkaraju. 2024. Towards unifying interpretability and control: Evaluation via intervention. *Preprint*, arXiv:2411.04430.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. 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.

J Bloom and J Lin. 2024. Understanding sae features with the logit lens. In *AI Alignment Forum*.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. 2021. An interpretability illusion for bert. *Preprint*, arXiv:2104.07143.

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. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Gonçalo Paulo Caden Juang, Jacob Drori, and Nora Belrose. 2024. Open source automated interpretability for sparse autoencoder features. *EleutherAI Blog, July*, 30.

Nitay Calderon, Roi Reichart, and Rotem Dror. 2025. The alternative annotator test for llm-as-a-judge: How to statistically justify replacing human annotators with llms. *Preprint*, arXiv:2501.10970.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. 2024. Improving steering vectors by targeting sparse autoencoder features. *Preprint*, arXiv:2411.02193.

Dami Choi, Vincent Huang, Kevin Meng, Daniel D Johnson, Jacob Steinhardt, and Sarah Schwettmann. 2024. Scaling automatic neuron description. `https://transluce.org/neuron-descriptions`.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. 2024. Recurrent neural networks learn to store and generate sequences using non-linear representations. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 248–262, Miami, Florida, US. Association for Computational Linguistics.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James R. Glass. 2018. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. *CoRR*, abs/1812.09355.

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

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. 2024a. Applying sparse autoencoders to unlearn knowledge in language models. *Preprint*, arXiv:2410.19278.

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. 2024b. Applying sparse autoencoders to unlearn knowledge in language models. *arXiv preprint arXiv:2410.19278*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

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

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. 2024a. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Preprint*, arXiv:2301.04709.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. 2024b. Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning, 1-3 April 2024, Los Angeles, California, USA*, volume 236 of *Proceedings of Machine Learning Research*, pages 160–187. PMLR.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022b. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. 2024a. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *Preprint*, arXiv:2410.20526.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, et al. 2024b. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *arXiv preprint arXiv:2410.20526*.

Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. 2022. Natural language descriptions of deep features. In *International Conference on Learning Representations*.

Jing Huang, Atticus Geiger, Karel D'Oosterlinck, Zhengxuan Wu, and Christopher Potts. 2023. Rigorously assessing natural language explanations of neurons. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 317–331, Singapore. Association for Computational Linguistics.

Jing Huang, Zhengxuan Wu, Christopher Potts, Mor Geva, and Atticus Geiger. 2024. RAVEL: Evaluating interpretability methods on disentangling language model representations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8669–8687, Bangkok, Thailand. Association for Computational Linguistics.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Curt Tigges Joseph Bloom and David Chanin. 2024. Saelens. `https://github.com/jbloomAus/SAELens`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Dmitrii Kharlapenko, neverix, Neel Nanda, and Arthur Conmy. 2024. Self-explaining SAE features.

Laura Kopf, Philine Lou Bommer, Anna Hedström, Sebastian Lapuschkin, Marina M. C. Höhne, and Kirill Bykov. 2024. Cosy: Evaluating textual explanations of neurons. *Preprint*, arXiv:2405.20331.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300, Miami, Florida, US. Association for Computational Linguistics.

Johnny Lin and Joseph Bloom. 2023. Neuronpedia: Interactive reference and tooling for analyzing neural networks with sparse autoencoders. Software available from neuronpedia.org.

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

Alireza Makhzani and Brendan Frey. 2014. k-sparse autoencoders. *Preprint*, arXiv:1312.5663.

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

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Jesse Mu and Jacob Andreas. 2020. Compositional explanations of neurons. In *Advances in Neural Information Processing Systems*, volume 33, pages 17153–17163. Curran Associates, Inc.

Aaron Mueller, Jannik Brinkmann, Millicent L. Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. 2024. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability. *CoRR*, abs/2408.01416.

Seil Na, Yo Joong Choe, Dong-Hyun Lee, and Gunhee Kim. 2019. Discovery of natural language concepts in individual units of cnns. In *International Conference on Learning Representations*.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Nostalgebraist. 2020. interpreting GPT: the logit lens.

Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangde. 2024a. Steering language model refusal with sparse autoencoders. *Preprint*, arXiv:2411.11296.

Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangde. 2024b. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. 2024a. The geometry of categorical and hierarchical concepts in large language models. *Preprint*, arXiv:2406.01506.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024b. The linear representation hypothesis and the geometry of large language models. *Preprint*, arXiv:2311.03658.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. 2024. Automatically interpreting millions of features in large language models. *Preprint*, arXiv:2410.13928.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

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.

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. 2024. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*.

Chandan Singh, Aliyah R Hsu, Richard Antonello, Shailee Jain, Alexander G Huth, Bin Yu, and Jianfeng Gao. 2023. Explaining black box text modules in natural language with language models. *arXiv preprint arXiv:2305.09863*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. 2017. Deep feature interpolation for image content changes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6090–6099.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2024. Neurons in large language models: Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1288–1301, Bangkok, Thailand. Association for Computational Linguistics.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: Detecting and editing privacy neurons in pretrained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886, Singapore. Association for Computational Linguistics.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2024. Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, Torino, Italia. ELRA and ICCL.

A Additional Details on Feature Description Evaluations

Input-based We used the prompt in Figure 4 for generating activating and neutral sentences based on a feature description, as per the input metric.

Output-based We used the prompt in Figure 7 for tasking the judge LLM with telling the steered text generations apart using a feature description, as per the output metric. Figure 14 shows an example of a steered text set for the feature with the description "`urgent global issues such as epidemics and invasions`".

The clamping values for m were derived by fixing two target KL-divergence values, 0.25 and 0.5, providing two positive and two negative clamping values for m. These values, along with sequence length, balance generating text with sufficient feature effect, and producing long or degenerate text that is difficult to evaluate. To confirm this, we ran the output-based evaluation using target KL-divergence values of 0.5 and 1 on 800 features from Gemma-2 2B, obtaining similar results. However, the generated text became more degenerate (see Figure 16 for an example text). Therefore, we decided to retain the original target KL-divergence values, as higher values resulted in text that did not reflect probable model behavior.

B Additional Experimental Details

B.1 Variants of `VocabProj`

When implementing `VocabProj`, presented in §4, there are several variants that generate tokens we can choose from, which are determined by the weight matrices we utilize. There are two points of interest: (a) the projection destination in the model (*unembedding* matrix $W_U \in \mathbb{R}^{d\times|\mathcal{V}|}$ vs. *embedding* matrix $W_E \in \mathbb{R}^{|\mathcal{V}|\times d}$), (b) in the case of SAEs, the source of the feature vector we analyze when applying the SAE on the hidden representation (*encoding* matrix $W_{enc} \in \mathbb{R}^{d\times d_{sae}}$ vs. *decoding* matrix $W_{dec} \in \mathbb{R}^{d_{sae}\times d}$). We conducted experiments across all of our subject models (except Llama-3.1 8B Instruct), in order to choose the best variant of this method.

Decode vs. Encode We first wished to tackle the decision of (a). To do so, we conducted a small-scale experiment in which we took a random sample of SAE features, using the following SAE types: Gemma Scope 16K, Llama Scope 32K and OpenAI SAE 32K; considering both layers (MLP and residual), for each subject model. This resulted in 52 features from Gemma-2 2B, 64 from Llama-3.1 8B, and 24 from GPT-2 small. Due to the small sample size of features, we used bootstrap (9999 resamples of the data with replacements, 95% confidence) to estimate the accuracy of each variant. We used our chosen prompt (see §B.2 for more details), to generate descriptions given the tokens retrieved using each of the 4 combinations above. We evaluated the descriptions using our input metric presented in §3. Table 3 shows the confidence interval for each variant on each model. From the table we concluded that generally the decoding matrix variant outperforms the encoding one.

Variant	Gemma-2 2B (Gemma Scope 16K)	Llama-3.1 8B (Llama Scope 32K)	GPT-2 small (OpenAI SAE 32K)
Dec & Unembed	0.44 (0.31-0.58)	0.27 (0.17-0.39)	0.29 (0.12-0.50)
Enc & Unembed	0.38 (0.27-0.52)	0.14 (0.06-0.25)	0.25 (0.12-0.46)
Dec & Embed	0.52 (0.38-0.65)	0.20 (0.11-0.31)	0.25 (0.08-0.46)
Enc & Embed	0.29 (0.17-0.42)	0.16 (0.08-0.27)	0.21 (0.08-0.42)

Table 3: Confidence interval of mean input metric results on the descriptions generated by `VocabProj` using tokens retrieved by 4 different methods, to compare decoding vs. encoding variants.

Method	Estimated FLOPs		
	Gemma-2 2B	Gemma-2 9B	Gemma-2 27B
`MaxAct`	$3.9 \cdot 10^{16}$	$1.5 \cdot 10^{17}$	$5 \cdot 10^{17}$
`VocabProj`	$2.8 \cdot 10^{14}$	$1.1 \cdot 10^{15}$	$2 \cdot 10^{15}$
`TokenChange`	$9.9 \cdot 10^{13}$	$4.1 \cdot 10^{14}$	$1.3 \cdot 10^{15}$

Table 4: Estimated FLOPs for generating descriptions for all MLP features for models of different sizes, on a sample of 25k sequences of 128 tokens each, as done by Neuronpedia.

Unembed vs. Embed We then conducted a larger scale experiment to tackle decision (b). We used the same SAEs and models from our previous experiment, taking a random sample of 5 features per SAE, considering both layers (MLP and residual), for each subject model. This resulted in 260 features from Gemma-2 2B, 320 from Llama-3.1 8B, and 120 from GPT-2 small. Table 5 shows the confidence interval for each variant on each model. From the table we concluded that the unembedding variant outperforms the embedding one, therefore we chose the decoding-unembedding variant for `VocabProj`.

B.2 Description Generation

`VocabProj` We use the prompt in Figure 8 given to the explainer model for it to generate feature descriptions using `VocabProj`.

We tried different prompts, but didn't observe significant improvement. These include both generic prompts to be used for all subject models (Figures 15 and 17), and more fine-tuned prompts based on vocabulary projection demonstrations for each subject model (see the fine-tuned based prompt in Figure 13, for which we concatenate few-shot examples for each model as seen in Figure 11).

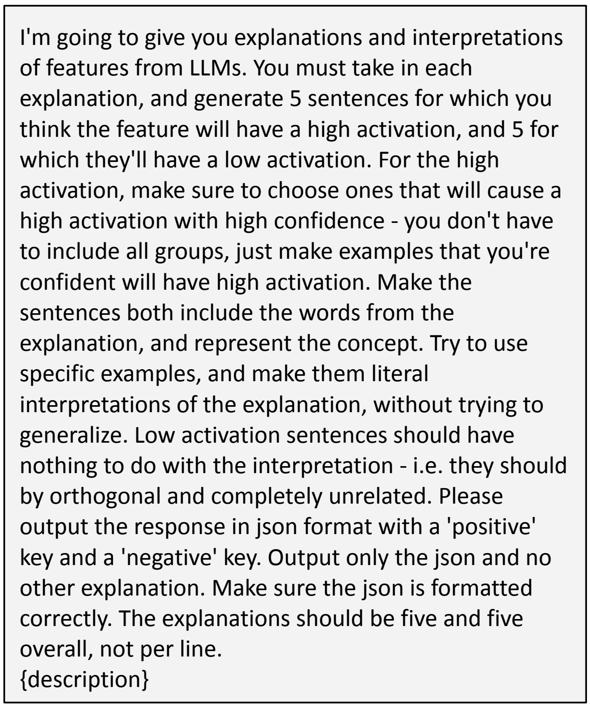

I'm going to give you explanations and interpretations of features from LLMs. You must take in each explanation, and generate 5 sentences for which you think the feature will have a high activation, and 5 for which they'll have a low activation. For the high activation, make sure to choose ones that will cause a high activation with high confidence - you don't have to include all groups, just make examples that you're confident will have high activation. Make the sentences both include the words from the explanation, and represent the concept. Try to use specific examples, and make them literal interpretations of the explanation, without trying to generalize. Low activation sentences should have nothing to do with the interpretation - i.e. they should by orthogonal and completely unrelated. Please output the response in json format with a 'positive' key and a 'negative' key. Output only the json and no other explanation. Make sure the json is formatted correctly. The explanations should be five and five overall, not per line.
{description}

Figure 4: Prompt given to the judge LLM for the input-based evaluation.

Ensembles To generate `Ensemble Raw` descriptions, we used variations of the prompt in Figure 12 when the ensemble included `MaxAct`. To generate `Ensemble Raw (VocabProj+TokenChange)` we simply concatenate the tokens generated by the two methods and use the `VocabProj` prompt.

B.3 Recreating Neuronpedia Descriptions using `MaxAct`

In order to compare our own generated descriptions to the ones provided in Neuronpedia, we conducted an experiment across all of our subject models (except Llama-3.1 8B Instruct) where we regenerated a description based on the activations data provided by Neuronpedia, fed to `MaxAct`, following their automatic pipeline based on Bills et al. (2023). For a given feature, the explainer model gets as input the 5 top-activating sentences in the format of token-activation pairs, and generates a description adapting their code[2] to our pipeline.

We took a random sample of 360 SAE features from each model, using the following SAE types:

---

[2] `https://github.com/hijohnnylin/automated-interpretability`

Variant	Gemma-2 2B (Gemma Scope 16K)	Llama-3.1 8B (Llama Scope 32K)	GPT-2 small (OpenAI SAE 32K)
Dec & Unembed	0.41 (0.35-0.47)	0.14 (0.11-0.19)	0.20 (0.13-0.28)
Dec & Embed	0.37 (0.31-0.43)	0.12 (0.08-0.16)	0.13 (0.08-0.21)

Table 5: Confidence interval of mean input metric results on the descriptions generated by `VocabProj` using tokens retrieved by 2 different methods, to compare unembedding vs. embedding variants with the decoding matrix.

Variant	Gemma-2 2B (Gemma Scope 16K)	Llama-3.1 8B (Llama Scope 32K)	GPT-2 small (OpenAI SAE 32K)	ALL
Neuronpedia	0.49 (0.44-0.55)	0.46 (0.41-0.52)	0.41 (0.36-0.47)	0.46 (0.43-0.49)
`MaxAct`	0.52 (0.46-0.57)	0.47 (0.42-0.53)	0.44 (0.39-0.50)	0.48 (0.45-0.51)

Table 6: Confidence interval of mean input metric results on the descriptions taken from Neuronpedia and those generated by `MaxAct`.

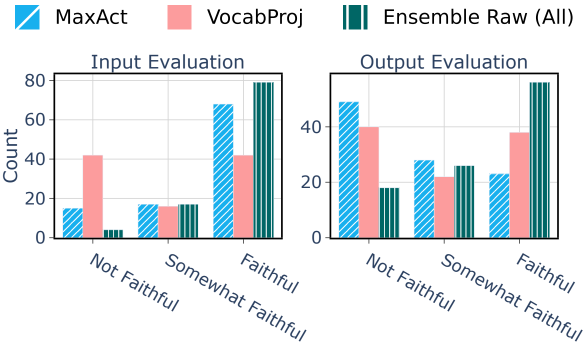


Figure 5: Human evaluation results for 100 features across three methods, for input- and output-faithfulness.

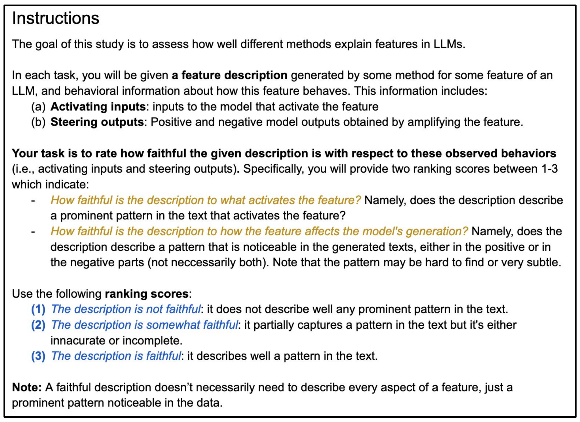
Gemma Scope 16K and 65K, Llama Scope 32K and OpenAI SAE 32K and 128K; considering both layers (MLP and residual). We evaluated both sets of descriptions using our input-based metric, and observed that they reach similar performance. Table 6 shows the confidence interval for the mean input metric evaluating both Neuronpedia's descriptions and our recreated descriptions.

C Additional Evaluation Results

See Figure 10 and Table 7 for additional results from Llama-3.1 8B and GPT-2 small SAE features, overall following the same trends observed in §5. Results for GPT-2 small are noisier than in other models. This may be due to the model's relatively small size and generally lower performance.

D Computational Cost Analysis

The computational cost of each method is a key factor to consider when selecting a method for generating descriptions. In our analysis, we computed the FLOPs required by each method to generate a description for every single MLP feature in Gemma-2 2B (results in Table 4). When calculating the FLOPs required for a single forward pass, we rely on the heuristic FLOPs $\approx$ 6N plus embedding FLOPs, where N is the total number of non-embedding model parameters (Kaplan et al., 2020; Anil et al., 2023). The results show that even when using a small sample for `MaxAct`—25k sequences of 128 tokens each, as used by Neuronpedia—alternative methods are 2-3 orders of magnitude more compute-efficient. When using larger samples that more accurately represent a model's training data, such as The Pile (Gao et al., 2020), the difference reaches 7-8 orders of magnitude. Lastly, computational cost for analysing SAE features results in an increase of roughly one order of magnitude across the board, while maintaining the same relative differences between methods.



Instructions

The goal of this study is to assess how well different methods explain features in LLMs.

In each task, you will be given **a feature description** generated by some method for some feature of an LLM, and behavioral information about how this feature behaves. This information includes:
(a) **Activating inputs**: inputs to the model that activate the feature
(b) **Steering outputs**: Positive and negative model outputs obtained by amplifying the feature.

Your task is to rate how faithful the given description is with respect to these observed behaviors (i.e., activating inputs and steering outputs). Specifically, you will provide two ranking scores between 1-3 which indicate:

- *How faithful is the description to what activates the feature?* Namely, does the description describe a prominent pattern in the text that activates the feature?
- *How faithful is the description to how the feature affects the model's generation?* Namely, does the description describe a pattern that is noticeable in the generated texts, either in the positive or in the negative parts (not neccessarily both). Note that the pattern may be hard to find or very subtle.

Use the following **ranking scores**:
(1) *The description is not faithful*: it does not describe well any prominent pattern in the text.
(2) *The description is somewhat faithful*: it partially captures a pattern in the text but it's either innacurate or incomplete.
(3) *The description is faithful*: it describes well a pattern in the text.

Note: A faithful description doesn't necessarily need to describe every aspect of a feature, just a prominent pattern noticeable in the data.

Figure 6: Instructions provided to human annotators for the evaluation of feature descriptions. These were accompanied with a few example annotations.

	Llama-3.1 MLP SAE		**GPT2 Res. SAE**		**GPT2 MLP SAE**	
	Input	Output	Input	Output	Input	Output
`MaxAct`	56.4 ± 2.9	49.6 ± 2.9	44.4 ± 2.3	44.1 ± 2.3	39.7 ± 2.9	34 ± 2.8
`VocabProj`	20.2 ± 2.3	48.2 ± 2.9	23.7 ± 2	42.8 ± 2.3	6.3 ± 1.4	**38.3** ± 2.9
`TokenChange`	25.4 ± 2.5	**53.1** ± 2.9	25.4 ± 2.1	43.4 ± 2.3	6.1 ± 1.4	36.5 ± 2.8
`EnsembleR (MA+VP)`	62.1 ± 2.8	45.8 ± 2.9	**59.6** ± 2.3	**47.2** ± 2.4	**51.2** ± 2.9	**38.1** ± 2.9
`EnsembleR (MA+TC)`	**65.8** ± 2.8	48.9 ± 2.9	**58.8** ± 2.3	**47.2** ± 2.4	**51.1** ± 2.9	**40.3** ± 2.9
`EnsembleR (VP+TC)`	22.6 ± 2.4	50.7 ± 2.9	29.2 ± 2.1	44.2 ± 2.4	7.1 ± 1.5	**40.9** ± 2.9
`EnsembleR (All)`	62.7 ± 2.8	51.6 ± 2.9	**60.4** ± 2.3	**47.2** ± 2.4	**50.2** ± 2.9	37.1 ± 2.8
`EnsembleC (All)`	39.1 ± 2.8	**55.5** ± 2.9	42.4 ± 2.3	**46.9** ± 2.4	24.4 ± 2.5	37.2 ± 2.8

Table 7: Input- and output-based evaluation results of the methods and their ensembles, over different feature types and models, averaged across model layers, along with their respective 95% confidence intervals. For GPT-2 small SAE features we take ones with width 32k. We denote `MA` for `MaxAct`, `VP` for `VocabProj`, `TC` for `TokenChange`, and `EnsembleR` and `EnsembleC` for the raw and concatenation based ensembles.

E Human Evaluations

To lend credence to our use of an LLM-judge and assess how well LLM-generated feature descriptions align with human judgment, we conducted two human evaluations.

E.1 Justifying Using an LLM as a Judge

To justify our use of an LLM-as-a-judge in the output-based evaluation, we apply the alternative annotator test proposed by Calderon et al. (2025). Following their procedure, we use three human annotators (graduate students) and a set of 100 randomly selected feature examples, evenly split between Llama-3.1 8B and Gemma-2 2B. For each feature, human annotators were given feature descriptions generated by `VocabProj`, and the three text sets $\mathcal{T}_{\mathbf{v}_f}$, $\mathcal{T}_{\mathbf{v}'_f}$, and $\mathcal{T}_{\mathbf{v}''_f}$. Each annotator then indicated which of the three sets matches the given description, as per the output-based metric. Consistent with Calderon et al. (2025), we set $\epsilon = 0.15$ to reflect our use of graduate student annotators. The analysis yielded a winning rate $\omega = 1$ with p-value 0.03, supporting our use of an LLM-as-a-judge.

E.2 Evaluating LLM Generated Descriptions

To evaluate how well LLM-generated feature descriptions align with human judgment, we tasked human annotators (6 graduate students) with scoring their faithfulness with respect to (a) input-faithfulness: what activates the feature and (b) output-faithfulness: how the feature affects the model's outputs. The instructions provided to the annotators are shown in Figure 6. We collected annotations for feature descriptions generated by `MaxAct`, `VocabProj`, and `Ensemble Raw (All)` for 100 randomly selected SAE features from Gemma-2 2B.

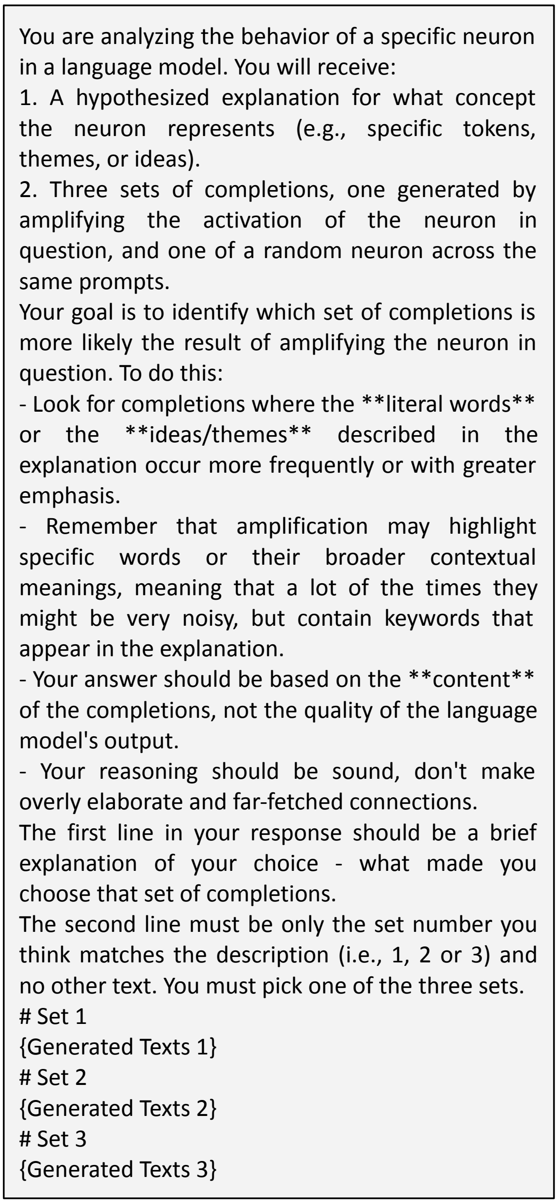
Figure 5 shows the results, where the overall trends align well with our proposed input- and output-based evaluations, discussed in §5.2. `MaxAct` performs better on the input evaluation, `VocabProj` on the output evaluation, and `Ensemble Raw (All)` performs best on both. However, `VocabProj` performed slightly worse than expected on the output evaluation. This discrepancy may stem from the difficulty humans face in evaluating a feature's effect on text generation, as it requires detecting subtle changes across multiple texts. Indeed, in the annotator test conducted in §E, the judge LLM outperformed human annotators, supporting this claim. Furthermore, `MaxAct`'s success in the input evaluation could be influenced by the descriptions being derived from the same data used for comparison, potentially biasing results in its favor. Nonetheless, these findings reinforce the claims in §5.2, that input-centric methods perform better on input-based evaluations, output-centric methods on output-based ones, and ensembles perform best on both.

F Additional Details and Examples for Dead Feature Analysis

F.1 Generating Candidate Prompts

To generate the candidate prompts, we first generate 150 potentially activating sentences in the same way as when doing so for the output metric, based on `VocabProj` and `MaxAct`. We then compile a list of tokens using both `VocabProj` and `TokenChange`, and create candidate prompts that begin with `<BOS>` followed by either of the following:

- A single token (1 candidate per token).
- Two random tokens (250 candidates).



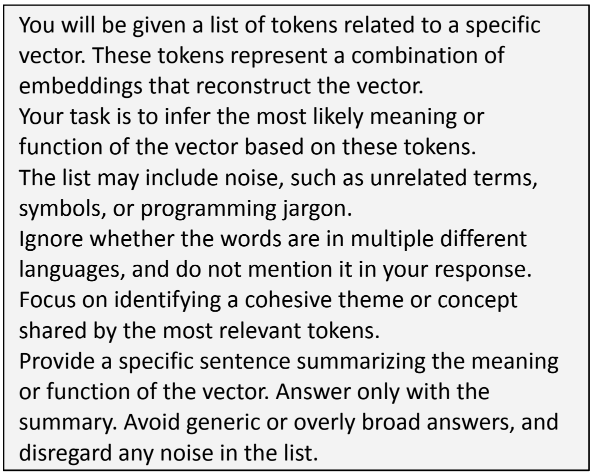
You are analyzing the behavior of a specific neuron in a language model. You will receive:
1. A hypothesized explanation for what concept the neuron represents (e.g., specific tokens, themes, or ideas).
2. Three sets of completions, one generated by amplifying the activation of the neuron in question, and one of a random neuron across the same prompts.
Your goal is to identify which set of completions is more likely the result of amplifying the neuron in question. To do this:
- Look for completions where the **literal words** or the **ideas/themes** described in the explanation occur more frequently or with greater emphasis.
- Remember that amplification may highlight specific words or their broader contextual meanings, meaning that a lot of the times they might be very noisy, but contain keywords that appear in the explanation.
- Your answer should be based on the **content** of the completions, not the quality of the language model's output.
- Your reasoning should be sound, don't make overly elaborate and far-fetched connections.
The first line in your response should be a brief explanation of your choice - what made you choose that set of completions.
The second line must be only the set number you think matches the description (i.e., 1, 2 or 3) and no other text. You must pick one of the three sets.
Set 1
{Generated Texts 1}
Set 2
{Generated Texts 2}
Set 3
{Generated Texts 3}

Figure 7: Prompt given to the judge LLM for the output-based evaluation.

- Three random tokens (250 candidates).
- Five random tokens (200 candidates).
- Twelve random tokens (200 candidates).
- Twenty-five random tokens (100 candidates).
- Thirty-two random tokens (50 candidates).

F.2 Dead Feature Revival Example

As an example of a feature deemed to be dead that we managed to revive, and that also has a clear and faithful description, we take residual stream SAE feature 64628 in layer 23 of Gemma-2 2B. Using `VocabProj` we can get an explanation for the feature: “`gaming, focusing on players, gameplay, and game mechanics`”. Indeed when examining the top tokens when projecting the feature to vocabulary space, they are all related to games, and players. The candidate prompt that managed to trigger this feature is “`**Player Agency**: Choices and consequences, branching narratives.`”. We can then see in Figure 9 that this description is faithful when amplifying the feature and examining text generated from open ended prompts, like in the output evaluation.



You will be given a list of tokens related to a specific vector. These tokens represent a combination of embeddings that reconstruct the vector.
Your task is to infer the most likely meaning or function of the vector based on these tokens.
The list may include noise, such as unrelated terms, symbols, or programming jargon.
Ignore whether the words are in multiple different languages, and do not mention it in your response.
Focus on identifying a cohesive theme or concept shared by the most relevant tokens.
Provide a specific sentence summarizing the meaning or function of the vector. Answer only with the summary. Avoid generic or overly broad answers, and disregard any noise in the list.

Figure 8: Prompt given to the explainer model for the `VocabProj` method.

G Additional Examples for Qualitative Analysis

Table 8 shows descriptions generated by `MaxAct`, `VocabProj` and `TokenChange`.

H Resources and Packages

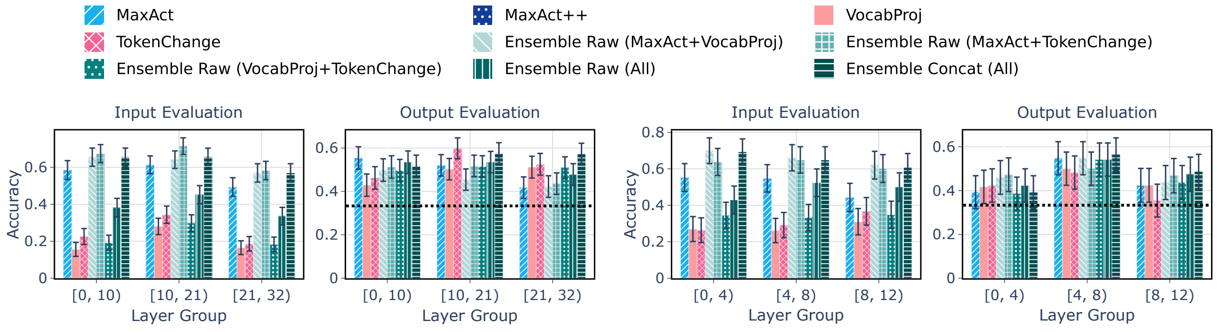
In our experiments, we used models, data and code from the following packages: transformers (Wolf, 2019), datasets (Lhoest et al., 2021), TransformerLens (Nanda and Bloom, 2022) and SAELens (Joseph Bloom and Chanin, 2024). The authors also made use of AI models, specifically ChatGPT, for implementing specific helper functions. All of the experiments were conducted using a single A100 80GB or H100 80GB GPU.

Example feature layer-type/id	**Description by `MaxAct`**	**Description by `VocabProj`**	**Description by `TokenChange`**
`3-MLP-16K/ 4878`	Terms and themes related to various genres of storytelling, particularly in horror, drama, and fantasy.	A blend of themes and genres commonly found in storytelling or media, with a specific focus on dramatic, horror, and suspenseful narratives.	Categorization or analysis of music and entertainment genres, possibly including content recommendations or thematic associations.
`19-MLP-16K/ 5635`	References to political events and milestones.	Concepts related to time measurement such as days, weeks, weekends, and months, indicating it likely pertains to scheduling or planning events.	Time periods, particularly weeks and weekends, along with some programming or markup elements for building or managing templates or components.
`21-RES-16K/ 10714`	Information related to bird species and wildlife activities.	Concepts related to birdwatching and ornithology, focusing on activities such as observing, spotting, and recording bird species in their natural habitats.	Enhancing or analyzing bird watching or ornithological data and experiences, possibly improving the tracking of bird sightings and interactions.
`19-MLP-16K/ 1450`	Mentions of notable locations, organizations, or events, particularly in various contexts.	Concepts related to self-reflection, purpose, and generalization in various contexts, focusing on the exploration of identity and overarching themes in literature or philosophy.	Recognize and generate variations of the term "general" and its context, along with concepts associated with insight and observation.

Table 8: Example descriptions by `MaxAct`, `VocabProj` and `TokenChange` for 4 SAE features from GemmaScope.

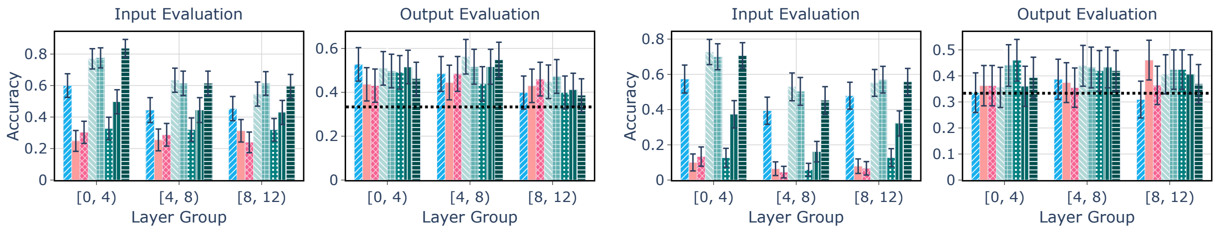
```
<+0.25>'I think': ' it's a really good idea to introduce the game in a way that is not just a tutorial'
<+0.25>'The explanation is simple:': ' the game has been updated to the new version of the game.'
<+0.25>'We': ' are a group of friends who are trying to get together and have a fun night of bowling. We'
<+0.5>'I think': ' the main reason is that the game is not really balanced. The game is not balanced at all.'
<+0.5>'The explanation is simple:': ' it is not possible to play <em><strong>FIFA 22</strong></em> with the new console without'
<+0.5>'We': ' are a group of players who are looking for new friends to play with!'
```

Figure 9: Text generated when amplifying a feature pronounced to be dead, which we managed to activate using the explanation generated by `VocabProj`, which was “gaming, focusing on players, gameplay, and game mechanics”.



(a) MLP 32k SAE features from Llama-3.1.

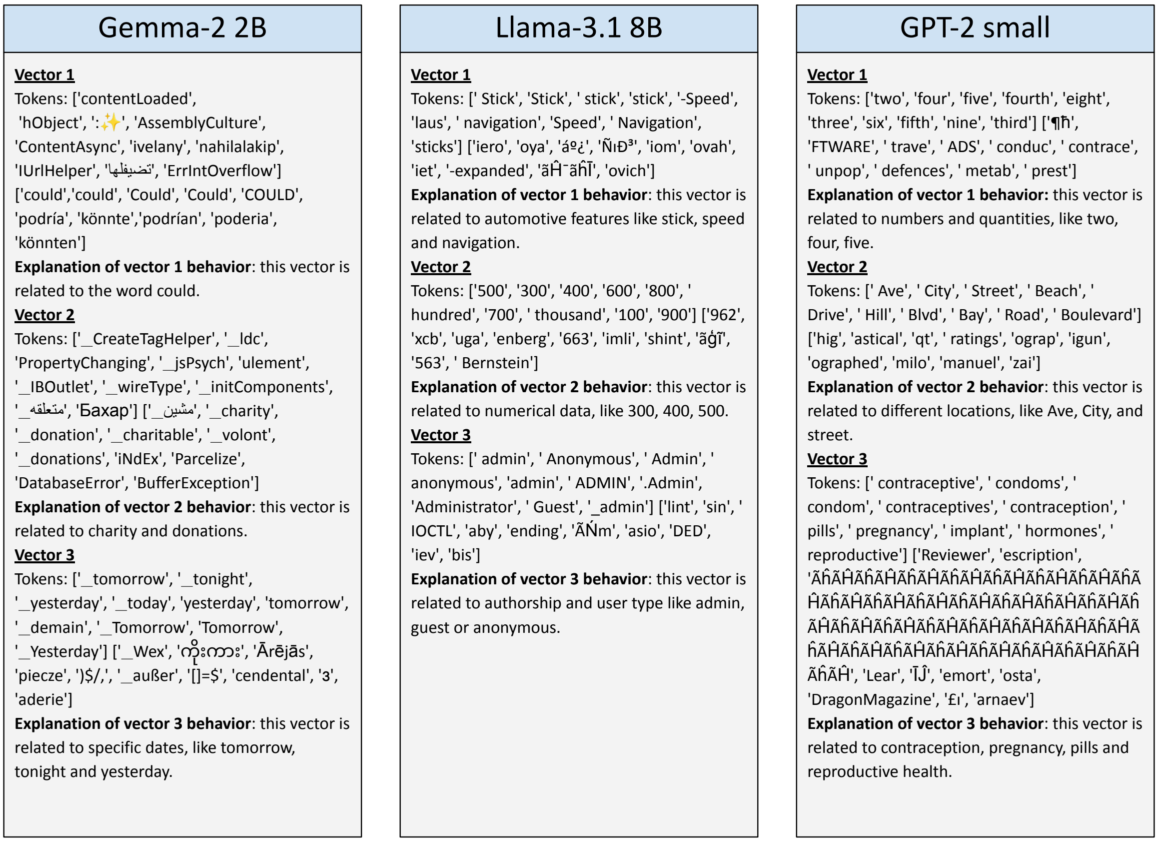
(b) Mid residual stream 32k SAE features from GPT-2 small.



(c) Residual stream 32k SAE features from GPT-2 small.

(d) MLP 32k SAE features from GPT-2 small.

Figure 10: Performance of the various methods on the proposed metrics, for Llama-3.1 8B (upper left) and GPT-2 small (upper right and lower row). For the output metric, the baseline (dashed black line) is $1/3$ since the judge LLM picks between three sets of texts.



Gemma-2 2B

Vector 1

Tokens: ['contentLoaded', 'hObject', ':✨', 'AssemblyCulture', 'ContentAsync', 'ivelany', 'nahilalakip', 'IUrlHelper', 'تضيفلها', 'ErrIntOverflow'] ['could','could', 'Could', 'Could', 'COULD', 'podría', 'könnte','podrían', 'poderia', 'könnten']

Explanation of vector 1 behavior: this vector is related to the word could.

Vector 2

Tokens: ['_CreateTagHelper', '_ldc', 'PropertyChanging', '_jsPsych', 'ulement', '_IBOutlet', '_wireType', '_initComponents', '_متعلقه', 'Бахар'] ['_مشين', '_charity', '_donation', '_charitable', '_volont', '_donations', 'iNdEx', 'Parcelize', 'DatabaseError', 'BufferException']

Explanation of vector 2 behavior: this vector is related to charity and donations.

Vector 3

Tokens: ['_tomorrow', '_tonight', '_yesterday', '_today', 'yesterday', 'tomorrow', '_demain', '_Tomorrow', 'Tomorrow', '_Yesterday'] ['_Wex', 'ကိုးကား', 'Ārējās', 'piecze', ')$/,', '_außer', '[]=$', 'cendental', '3', 'aderie']

Explanation of vector 3 behavior: this vector is related to specific dates, like tomorrow, tonight and yesterday.

Llama-3.1 8B

Vector 1

Tokens: [' Stick', 'Stick', ' stick', 'stick', '-Speed', 'laus', ' navigation', 'Speed', ' Navigation', 'sticks'] ['iero', 'oya', 'áº¿', 'ÑıÐ³', 'iom', 'ovah', 'iet', '-expanded', 'ãĤ¯ãĥĪ', 'ovich']

Explanation of vector 1 behavior: this vector is related to automotive features like stick, speed and navigation.

Vector 2

Tokens: ['500', '300', '400', '600', '800', ' hundred', '700', ' thousand', '100', '900'] ['962', 'xcb', 'uga', 'enberg', '663', 'imli', 'shint', 'ãģĪ', '563', ' Bernstein']

Explanation of vector 2 behavior: this vector is related to numerical data, like 300, 400, 500.

Vector 3

Tokens: [' admin', ' Anonymous', ' Admin', ' anonymous', 'admin', ' ADMIN', '.Admin', 'Administrator', ' Guest', '_admin'] ['lint', 'sin', ' IOCTL', 'aby', 'ending', 'ÃŃm', 'asio', 'DED', 'iev', 'bis']

Explanation of vector 3 behavior: this vector is related to authorship and user type like admin, guest or anonymous.

GPT-2 small

Vector 1

Tokens: ['two', 'four', 'five', 'fourth', 'eight', 'three', 'six', 'fifth', 'nine', 'third'] ['¶ħ', 'FTWARE', ' trave', ' ADS', ' conduc', ' contrace', ' unpop', ' defences', ' metab', ' prest']

Explanation of vector 1 behavior: this vector is related to numbers and quantities, like two, four, five.

Vector 2

Tokens: [' Ave', ' City', ' Street', ' Beach', ' Drive', ' Hill', ' Blvd', ' Bay', ' Road', ' Boulevard'] ['hig', 'astical', 'qt', ' ratings', 'ograp', 'igun', 'ographed', 'milo', 'manuel', 'zai']

Explanation of vector 2 behavior: this vector is related to different locations, like Ave, City, and street.

Vector 3

Tokens: [' contraceptive', ' condoms', ' condom', ' contraceptives', ' contraception', ' pills', ' pregnancy', ' implant', ' hormones', ' reproductive'] ['Reviewer', 'escription', 'ÃĥÃĤ', 'Lear', 'ĪĴ', 'emort', 'osta', 'DragonMagazine', '£ı', 'arnaev']

Explanation of vector 3 behavior: this vector is related to contraception, pregnancy, pills and reproductive health.

Figure 11: Three demonstrations of tokens and their descriptions for each model, added to the base prompt forming a fine-tuned prompt.

We're studying neurons in a neural network. Each neuron has certain inputs that activate it and outputs that it leads to. You will receive two pieces of information about a neuron: the activations it has for certain inputs, the words its output is most associated with. These will be separated into two sections [INPUT] and [OUTPUT].
The [INPUT] activation format is token<tab>activation. Activation values range from 0 to 10. A neuron finding what it's looking for is represented by a non-zero activation value. The higher the activation value, the stronger the match.
The [OUTPUT] format is a list of words related to that specific neuron. These tokens represent a combination of embeddings that reconstruct the vector. You can infer the most likely output or function of the neuron based on these tokens. The list may include noise, such as unrelated terms, symbols, or programming jargon. Ignore whether the words are in multiple different languages, and do not mention it in your response. Focus on identifying a cohesive theme or concept shared by the most relevant tokens.
Your response should be a concise (1-2 sentence) explanation of the neuron, encompassing what triggers it (input) and what it does once triggered (output). If the two sides relate to one another you may include that in your explanation, otherwise simply state the input and output.

Neuron 1

[INPUT]

```
   Activations:
<start>
esc	0
aping	10
 the	4
 studio	0
 ,	0
 pic	0
col	0
i	0
 is	0
 warm	0
ly	0
 affecting	3
 and	0
 so	0
 is	0
 this	0
 ad	0
roit	0
ly	0
 minimalist	0
 movie	0
 .	0
<end>
```

```
Same activations, but with all zeros filtered out:
<start>
'	1
 disappearing	6
earing	10
<end>
<start>
aping	10
 the	4
 affecting	3
<end>
```

[OUTPUT]

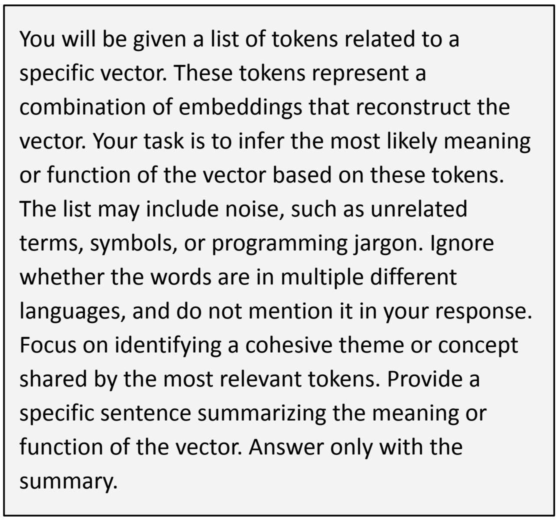
['to', 'To', 'TO', 'Towards', 'towards', 'TOWARDS', 'toward', 'Toward', 'TOWARD', 'toward', 'Toward', 'TOWARD', 'life', 'do', 'fdsa', 'aaaaaa', 'aaaaa', 'aaaa', 'aaa', 'aa', 'a', 'A']

Explanation of neuron 1 behavior: the main thing this neuron does is find present tense verbs ending in 'ing', and then outputs words related to directionality or movement to or towards something.

Neuron 2

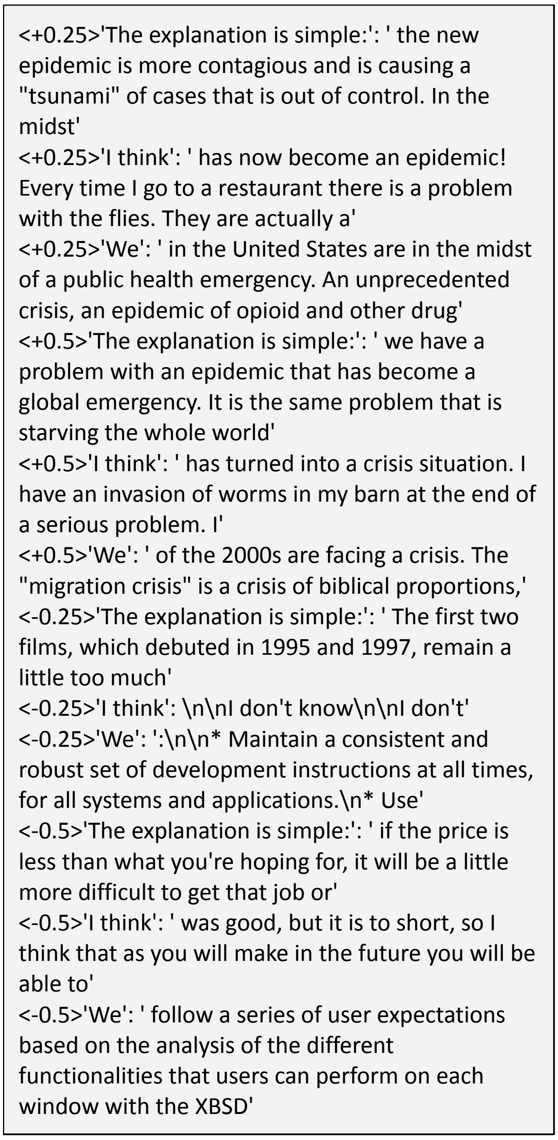
{Activation Info}
{Tokens}

Figure 12: Prompt given to the explainer model for the `Ensemble Raw` method.



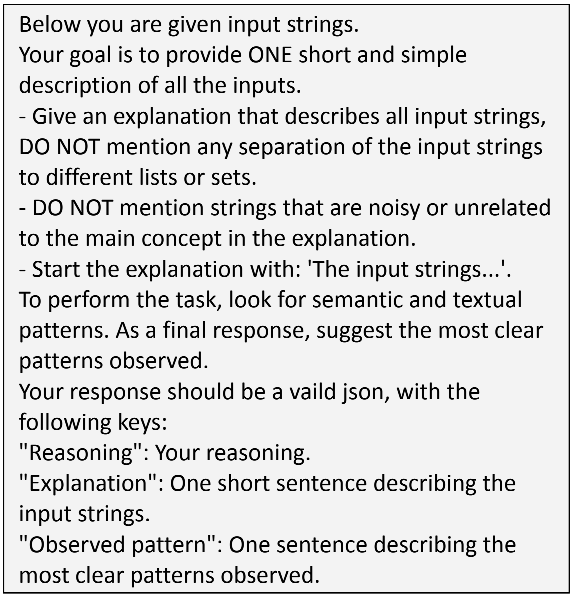
You will be given a list of tokens related to a specific vector. These tokens represent a combination of embeddings that reconstruct the vector. Your task is to infer the most likely meaning or function of the vector based on these tokens. The list may include noise, such as unrelated terms, symbols, or programming jargon. Ignore whether the words are in multiple different languages, and do not mention it in your response. Focus on identifying a cohesive theme or concept shared by the most relevant tokens. Provide a specific sentence summarizing the meaning or function of the vector. Answer only with the summary.

Figure 13: The basic fine-tuned prompt `VocabProj` method.



<+0.25>'The explanation is simple:': ' the new epidemic is more contagious and is causing a "tsunami" of cases that is out of control. In the midst'
<+0.25>'I think': ' has now become an epidemic! Every time I go to a restaurant there is a problem with the flies. They are actually a'
<+0.25>'We': ' in the United States are in the midst of a public health emergency. An unprecedented crisis, an epidemic of opioid and other drug'
<+0.5>'The explanation is simple:': ' we have a problem with an epidemic that has become a global emergency. It is the same problem that is starving the whole world'
<+0.5>'I think': ' has turned into a crisis situation. I have an invasion of worms in my barn at the end of a serious problem. I'
<+0.5>'We': ' of the 2000s are facing a crisis. The "migration crisis" is a crisis of biblical proportions,'
<-0.25>'The explanation is simple:': ' The first two films, which debuted in 1995 and 1997, remain a little too much'
<-0.25>'I think': \n\nI don't know\n\nI don't'
<-0.25>'We': ':\n\n* Maintain a consistent and robust set of development instructions at all times, for all systems and applications.\n* Use'
<-0.5>'The explanation is simple:': ' if the price is less than what you're hoping for, it will be a little more difficult to get that job or'
<-0.5>'I think': ' was good, but it is to short, so I think that as you will make in the future you will be able to'
<-0.5>'We': ' follow a series of user expectations based on the analysis of the different functionalities that users can perform on each window with the XBSD'

Figure 14: An example of a steered text set for the output-based metric.



Below you are given input strings.
Your goal is to provide ONE short and simple description of all the inputs.
- Give an explanation that describes all input strings, DO NOT mention any separation of the input strings to different lists or sets.
- DO NOT mention strings that are noisy or unrelated to the main concept in the explanation.
- Start the explanation with: 'The input strings...'.
To perform the task, look for semantic and textual patterns. As a final response, suggest the most clear patterns observed.
Your response should be a vaild json, with the following keys:
"Reasoning": Your reasoning.
"Explanation": One short sentence describing the input strings.
"Observed pattern": One sentence describing the most clear patterns observed.

Figure 15: A first variant of a generic prompt for the `VocabProj` method.

```
<+0.5>'The explanation is simple:': ' to buy or
purchase a property, you buy the legal ownership of
the home or buy the ownership of the land that you
would'
<+0.5>'I think': '2023 has the best sex buy sex buy buy
buy buy buy buy buy buy buy buy buy buy buy buy
buy'
<+0.5>'We':
'<b>-Buy-U-Buy-Buy-Buy-Buy-Buy-Buy-Buy-Buy-Buy-B
uy'
<+1>'The explanation is simple:': '  the purchase of the
firm-based      purchase-purchase-purchase-purchase
purchase would be a cash buy-purchase-purchase'
<+1>'I think': ' comprar comprar comprar comprar buy
buy buy purchase purchase purchase purchase
purchase purchase purchase purchase purchase
purchase purchase purchase rent rent rent rent rent
rent'
<+1>'We':  '<strong>-purchase  purchase  purchase
purchase  purchase  purchase  purchase  purchase
purchase  purchase  purchase  purchase  purchase
purchase      purchase      purchase      purchase
purchasepurchase   purchase   purchase   purchase
purchase'
<-0.5>'The explanation is simple:': ' Because some
forms of incontinence are more difficult to make
without the uronysys. A good integrated and
well-designed post'
<-0.5>'I think': ' 385 and 18 that are "11 and 181" are
all 100'
<-0.5>'We': ' are a team of 1000 and we are 100%
committed to creating the best-ever call'
<-1>'The explanation is simple:': ' "I' 118arroll 15 111
10 111 11'
<-1>'I think': '00011100000000000000000800'
<-1>'We': ' include some small and relatively easy-on-
"";\r
therein]));\r]));\r]));\r]));\r]));\r]));\r]));\r]));\r]));\
r]));\r]));\r]));\r]));\r'
```

Figure 16: Higher clamping value when steering feature with description “purchasing activities, including buying, viewing, and downloading products”, leading to degenerate text.

```
You are a meticulous AI researcher conducting an
important investigation into patterns found in
language.
Your task is to analyze text and provide an
explanation that thoroughly encapsulates possible
patterns found in it.
Guidelines:
You will be given a list of string tokens.
- Try to produce a concise final description. Simply
describe the text features that are common in the
tokens, and what patterns you found.
- If the tokens are uninformative, you don't need to
mention them. Try to summarize the patterns found
in the tokens.
- Do not make lists of possible explanations. Keep
your explanations short and concise.
- Give an explanation that describes all input strings,
DO NOT mention any separation of the input strings
to different lists or sets.
- DO NOT mention strings that are noisy or unrelated
to the main concept in the explanation.
Your response should be a vaild json, with the
following keys:
"Reasoning": Your reasoning.
"Explanation": One short sentence describing the
input strings.
"Observed pattern": One sentence describing the
most clear patterns observed.
```

Figure 17: A second variant of a generic prompt for the `VocabProj` method.