Open-Vocabulary Natural-Language Explanations of LLM Activations via Soft Prompts

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce *Latent-to-Explanation Likelihood* (L2EL), a simple interface that translates internal activations of a large language model (LLM) into a short natural-language explanation without changing the underlying LLM. Given a single hidden representation (e.g., a residual-stream activation at one token and layer), a tiny mapper produces a continuous "soft prompt" that conditions a *frozen* LLM to generate an explanation. We train the mapper with weak supervision from sparse autoencoder (SAE) explanations: for each latent we sample one natural language feature description among the active SAE features and optimize the soft prompt such that the LLM emits that description when conditioned on it. At test time, L2EL supports (i) *generation* of concise free-form explanations and (ii) *probing* by scoring arbitrary hypotheses through their conditional likelihood. This reframes interpretability as conditional language modeling over explanations, enabling an open vocabulary and calibration through likelihoods. As a proof-of-concept, we train L2EL on Gemma-2-2B using GemmaScope SAEs. Our results indicate that L2EL generates reasonable explanations and can be used to probe hidden activations using natural language. L2EL preserves the strengths of language as an expressive medium while requiring only a small learned interface and no modifications to the LLM.

1 Introduction

Large language models encode rich internal representations that often contain information they do not directly state. Turning those hidden representations into legible explanations is one of the central aims of mechanistic interpretability. Recent work has made progress in extracting information from these hidden representations by using sparse autoencoders (SAEs), which decompose activations into sparse, more interpretable features [1–6]. This approach is often insightful, but it is fundamentally closed-set: SAEs have a fixed learned dictionary of features, which means that the explanation vocabulary is fixed and any nuance not captured by those features is invisible at inference time.

We take a different route: rather than forcing latents into a finite bank of features, we let the model *describe* its own state in natural language. Concretely, we learn a tiny mapper that projects a single model activation (e.g., a residual-stream activation at one token and layer) into a few continuous "soft token" embeddings. These soft tokens condition the frozen LLM, which then generates a short free-form explanation of what is represented in that latent, without any prompt text (see Figure 1). This preserves the strengths of language as an expressive, open vocabulary while keeping the LLM unchanged.

To train the mapper, we use SAE feature explanations as a source of weak, scalable supervision: for each latent, we sample one human-readable description from among the active SAE features and ask the LLM to produce that description when conditioned on the latent-derived soft prompt. This

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

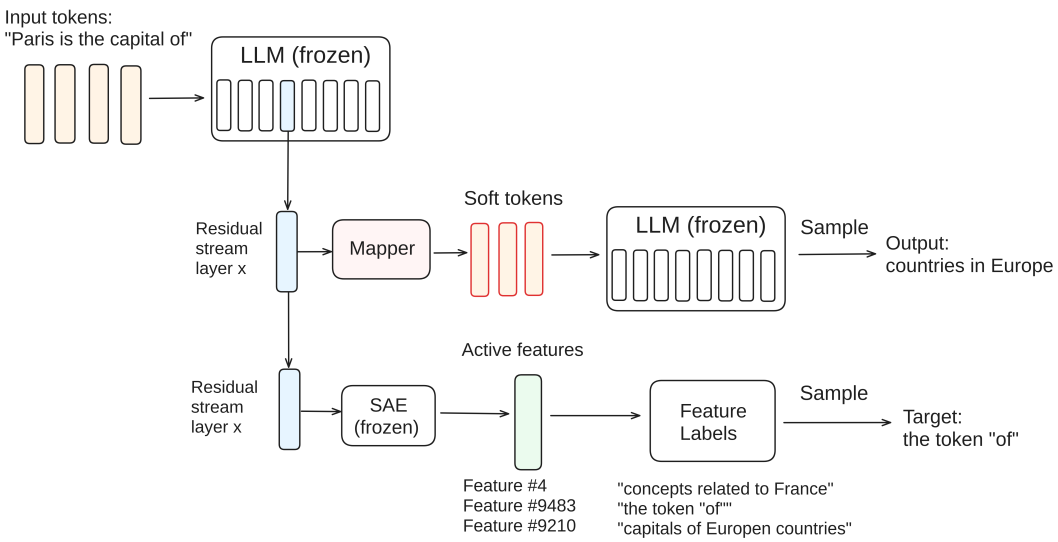

Figure 1: **L2EL overview.** A single residual-stream activation h is mapped into m soft token embeddings. The frozen LLM, conditioned solely on this soft prompt, learns to generate short textual explanations based on natural language explanations of SAE features. Only the mapper is trained.

simple objective encourages the LLM to learn a conditional distribution over all possible natural language explanations, not over a fixed index set. At test time, the same mechanism supports two complementary uses: (i) *generation*: produce a concise explanation of a latent; and (ii) *probing*: score arbitrary natural-language hypotheses against a latent by reading off their conditional likelihood under the LLM.

We formalize this as **latent-to-explanation likelihood (L2EL)**: given a latent vector h, our method learns a distribution $p(e \mid h)$ over textual explanations e. L2EL turns interpretability into standard language modeling conditioned on activations, enabling open-vocabulary explanations, probing with natural language explanations, and calibration through likelihoods rather than ad hoc thresholds.

2 Method: Latent-to-Explanation Likelihood

Let $h \in \mathbb{R}^d$ denote a residual-stream activation from the LLM at a particular token and layer. We learn a mapper $f_\phi : \mathbb{R}^d \to \mathbb{R}^{m \times d}$ that produces a continuous soft-prompt of m token embeddings $P = f_\phi(h)$ in the LLM's input-embedding space. Conditioning the frozen LLM on the soft prompt P defines an autoregressive distribution over explanations. For an explanation token sequence $e = (e_{1:T})$,

$$p(e \mid h) \;\equiv\; p\big(e \mid P = f_\phi(h)\big) \;=\; \prod_{t=1}^{T} p(e_t \,|\, e_{<t},\, P)\,,$$

which we call the *latent-to-explanation likelihood* (L2EL). Intuitively, f_ϕ translates an internal activation into a short soft prompt that steers the model to "speak" what that activation contains, while the LLM generates tokens autoregressively under this conditioning.

We train the mapper with weak supervision from automatically generated SAE feature explanations and optimize a masked next-token language-modeling objective conditioned only on the soft prompt. In particular, for each training context we extract a residual activation $h \in \mathbb{R}^d$, obtain the SAE feature activations for this vector, and sample one feature description y from the set of active features (for our sampling procedure, see Appendix A). Let $y_{1:T}$ be the tokenization of y. The mapper f_ϕ produces an m-token soft prefix $P = f_\phi(h) \in \mathbb{R}^{m \times d}$ that is prepended to the input of a *frozen* LLM with parameters θ. We then minimize the negative log-likelihood of the target tokens under teacher forcing:

$$\min_{\phi} \; \mathcal{L}(\phi) \;=\; \mathbb{E}_{(h,y)}\left[-\sum_{t=1}^{T} \log p_\theta\big(y_t \,\big|\, y_{<t},\, P = f_\phi(h)\big)\right].$$

Only the weights of f_ϕ are trained, the LLM parameters θ remain fixed. This trains the mapper to translate activations into a soft prompt that makes the base model likely to emit the sampled SAE description, thereby learning an open-vocabulary conditional distribution $p(e \mid h)$.

At test time, we take any context of interest, extract h at the chosen layer and token, form the soft prompt $P = f_\phi(h)$, and sample an explanation from $p(\cdot \mid P)$. The same mechanism enables probing: given an arbitrary hypothesis e, we compute $(e \mid h)$ and compare scores across model activations from different dataset examples to test what LLM inputs make the explanation most likely.

3 Experiments

We train L2EL on Gemma-2-2B [7], which provides both the activation h and the generation capability; only a small mapper sits between them. We focus on residual-stream activations from layer 10 and use the open-source GemmaScope SAE with a 65k-feature dictionary for weak supervision [8]. We obtain the natural language explanation for the SAE features from Neuronpedia [9], which are automatically generated with the method from [10]. We train the mapper on 200000 sequences of 256 tokens sampled from the Pile-uncopyrighted [11], totaling 51.2M training tokens.

For each token position we read the corresponding residual vector, compute SAE activations, sample one human-written feature description among the active labeled features, and treat this text as the language modeling target. For our mapper, we use a linear model that emits $m = 16$ soft prompt embeddings by applying a learned scalar gain and bias to the model activations h for each prefix slot, i.e., for $i = 1 \dots m$, $p_i = \gamma_i h + b_i$ with $\gamma_i \in \mathbb{R}$ and $b_i \in \mathbb{R}^d$.

3.1 Experiment 1: Generating Explanations

As a first experiment, we test whether the L2EL is able to generate reasonable explanations of latents. We feed the LLM the following prompt: "My favorite player is LeBron". We then extract the hidden activation and construct the soft prompt with our mapper. During generation, we prepend the resulting soft tokens and sample a diverse set of explanations (for more details see B). The prompt text is not provided, so the explanation is conditioned entirely through the soft prefix derived from h.

Table 1 shows that L2EL is able to generate diverse and coherent explanations, even when the top SAE explanations are not very informative, as there is no “LeBron James" feature in the SAE. We provide more generated explanations and details on our sampling procedure in Appendix B.

We can also use L2EL to calculate the likelihood of all of the pre-existing SAE feature explanations provided conditional on the latent. In Appendix B we show that the scoring of L2EL on these explanations sometimes seems to produce better results than the ordering by the SAE itself.

Table 1: Top activating SAE features and top generated explanations by L2EL on the following prompt: **My favorite player is LeBron**

Top explanations by SAE activation	**Top generated explanations (L2EL)**
specific tokens or markers that indicate important elements or categories within the text	references to the name "LeBron."
technical terms related to web applications and debugging processes	names of notable individuals or brands
references to notable individuals or performances in sports	names associated with the sports industry
technical terms and code-related elements in programming contexts	mentions of sports superstars and their impact on popular culture
specific technical terms and concepts related to database management and programming	names or terms related to celebrity culture or sports figures

Table 2: The dataset examples that make the following explanation most likely:
references to the wizard from Harry Potter

Snippet	Average log-prob
.. in Los Angeles, it was the battle of the bespectacled boy wizard	-1.905
.. for the first time plotting the start of Dumbledore	-1.982
.. Among these friends we call it Harry Potter	-1.988
.. that he is then speaking to Aberforth in Halfblood	-2.072
.. in Deathly Hallows once the trio arrive in Hogsme	-2.159
.. In Order of the Phoenix while the trio were in the Hog	-2.158

3.2 Experiment 2: Open-Vocabulary Probing

The second experiment tests the open-vocabulary probing capabilities of L2EL. Instead of restricting ourselves to SAE labels, we can formulate free-form explanations such as “references to the wizard from Harry Potter" and evaluate their scores across a held-out corpus. For each token position we compute $\log p(e \mid h)$ for the chosen explanation and then rank positions by their scores. Inspecting the top-ranked contexts reveals whether the explanation is genuinely capturing a coherent concept that the model recognizes in its internal states. Because this explanation is not tied to the SAE dictionary, success here indicates that L2EL can be used as a general-purpose probe over natural-language descriptions, even for concepts that an SAE did not learn explicitly. Table 2 shows that indeed all retrieved examples are related to Harry Potter.

4 Related Work

Our approach is most closely related to prompt/patch–based readouts of hidden states that elicit free-form text, such as Patchscopes [12], SelfIE [13], self-explaining SAE features [14], and LatentQA [15], as well as to “lens” methods that decode intermediate states into vocabulary distributions, including the logit lens and tuned lens [16, 17]. However, unlike these methods, we do not provide a fixed target inspection prompt at inference time (such as "What does X mean?"). Instead, we train a lightweight mapper that learns soft prompts such that the LLM itself outputs SAE feature explanations, without needing finetuning.

Furthermore, our work is related to other prompt-tuning methods learn continuous “soft prompts” that steer frozen language models for downstream tasks, including prefix-tuning [18] and prompt tuning [19]. Subsequent work improves robustness and scalability, e.g., P-Tuning v2 [20] and cross-task soft-prompt transfer (SPoT) [21].

5 Discussion and Limitations

We have presented L2EL as a proof-of-concept that translates model activations into short explanations. Although our results are promising, this work is intended as an initial demonstration of the idea rather than a mature method, and we emphasize that our experimental evidence is limited in scale and scope, with no direct comparisons to baselines, benchmarks, or ablations.

Furthermore, this approach comes with important limitations. First of all, the approach depends on automatically generated SAE feature descriptions that are noisy and inconsistently phrased, which constrains supervision quality. Secondly, it is currently unclear to what degree L2EL is generalizing to out-of-distribution hidden activations and explanations, and to what degree it is memorizing the SAE feature explanations.

Moreover, probing with natural language is sensitive to wording, with small variations in explanation giving different results. Finally, compared to SAEs, L2EL only provides natural language explanations of latents and no decomposition into feature vectors. This makes L2EL less suitable for causal interventions such as steering.

References

[1] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

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

[3] Adly Templeton. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic, 2024.

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

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

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

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

[8] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL `https://arxiv.org/abs/2408.05147`.

[9] Johnny Lin. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL `https://www.neuronpedia.org`. Software available from neuronpedia.org.

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

[11] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[12] Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024. doi: 10.48550/arXiv.2401.06102. URL `https://arxiv.org/abs/2401.06102`.

[13] Haozhe Chen, Carl Vondrick, and Chengzhi Mao. Selfie: Self-interpretation of large language model embeddings. *arXiv preprint arXiv:2403.10949*, 2024. doi: 10.48550/arXiv.2403.10949. URL `https://arxiv.org/abs/2403.10949`.

[14] Dmitrii Kharlapenko, neverix, Neel Nanda, and Arthur Conmy. Self-explaining sae features. LessWrong, August 2024. URL `https://www.lesswrong.com/posts/8ev6coxChSWcxCDy8/self-explaining-sae-features`. Accessed 22 Aug 2025.

[15] Alexander Pan, Lijie Chen, and Jacob Steinhardt. Latentqa: Teaching llms to decode activations into natural language. *arXiv preprint arXiv:2412.08686*, 2024. doi: 10.48550/arXiv.2412.08686. URL `https://arxiv.org/abs/2412.08686`.

[16] nostalgebraist. interpreting gpt: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

[17] Nora Belrose, Neel Nanda, Daniel Beren, Tom Henighan Wu, et al. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[18] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[19] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[20] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

[21] Tu Vu, Tong Wang, Brian Lester, Noah Constant, Rami Al-Rfou, Daniel Cer, and Yinfei Yang. SPoT: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2022.

A Sampling of SAE feature explanations

We want supervision to focus on features that are *informative and input-specific*. Very high–frequency SAE features behave like quasi-constants across contexts; if we trained on them, the mapper would learn to make those generic explanations likely for nearly every input, biasing $p(e \mid h)$. Likewise, raw magnitudes are not always comparable across features (different offsets/scales), so selecting by size alone favors "loud" features rather than those that are *unusually* strong for the current input. We therefore (i) filter out very frequent features and (ii) rank candidates by a robust z-score that measures atypicality relative to each feature's own variability.

Let the SAE produce post-ReLU activations $x^{\text{post}} \in \mathbb{R}^D_{\geq 0}$ and pre-activations $x^{\text{pre}} \in \mathbb{R}^D$. We estimate per-feature firing rates on a large activation stream and retain only features with empirical frequency $\leq 1\%$ (computed over 100,000 tokens). Then, for all features, we also compute the median and interquartile range (IQR) over the pre-activations and define $\sigma_j = \max(\text{IQR}_j/1.349, \varepsilon)$ with $\varepsilon = 10^{-6}$.

During training, for each input we calculate the feature activations, and select the *active* features ($x_j^{\text{post}} > 0$) that pass the frequency filter, rank them by

$$z_j = \frac{x_j^{\text{pre}} - \text{median}_j}{\sigma_j},$$

keep the top-k ($k = 5$), and uniformly sample one index. The automatically generated feature description of this index is then used as the target text.

B Generating Explanations

B.1 Sampling Generations

Given a hidden activation h, we form a soft prefix $P = f_\phi(h)$ and decode explanations autoregressively from $p_\theta(e_t \mid e_{<t}, P)$ using temperature and nucleus sampling. Concretely, at each step we divide logits by a temperature τ and sample from the smallest probability mass whose cumulative probability exceeds a threshold p (top-p). We use $\tau = 1.0$, $p = 0.9$, stop on EOS or at a maximum length of 32 tokens, and provide no textual prompt, the decoder is conditioned solely on the soft prompt P.

We draw $N = 50$ candidates, deduplicate strings, and score each with the average per-token log-likelihood $\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(e_t \mid e_{<t}, P)$ to reduce length bias. We select a diverse top-k set via greedy Maximal Marginal Relevance (MMR) on bigram Jaccard similarity: at each step we add the candidate e that maximizes

$$\beta\, \tilde{\ell}(e) \;-\; (1-\beta) \max_{e' \in S} \text{Jaccard}_{\text{bi}}(e, e'),$$

where $\tilde{\ell}$ is the min–max normalized average log-likelihood over the pool, S is the already selected set. We use $\beta = 0.2$.

B.2 Scoring Existing Feature Explanations

L2EL can turn hidden states into an open-vocabulary distribution over explanations, complementing the closed dictionary of an SAE. However, we can also use it score existing explanations for a prompt. To see how SAE activations relate to the log-probabilities under L2EL we calculate both values on the prompt: "The city that owns my heart is Kyoto"

The results are shown in Figure 2 and Table 3. Note that the explanations sorted by L2EL log-prob seem to be more relevant than the top explanations by SAE activation. For most of the top explanations by L2EL log-prob, the corresponding SAE activation is zero.

Table 3: Top SAE feature explanations scored by SAE activation and L2EL log-probability on the following prompt: **The city that owns my heart is Kyoto**

Top explanations by SAE activation	**Top explanations by L2EL log-prob**
specific tokens or markers that indicate important elements or categories within the text	locations and geographical references
technical terms and code-related elements in programming contexts	geographical names and locations
terms related to institutions and organizations	references to universities and academic institutions
proper nouns related to specific geographic locations	references to locations and places
content related to legal issues and historical contexts involving immigration or citizenship	references to academic journals

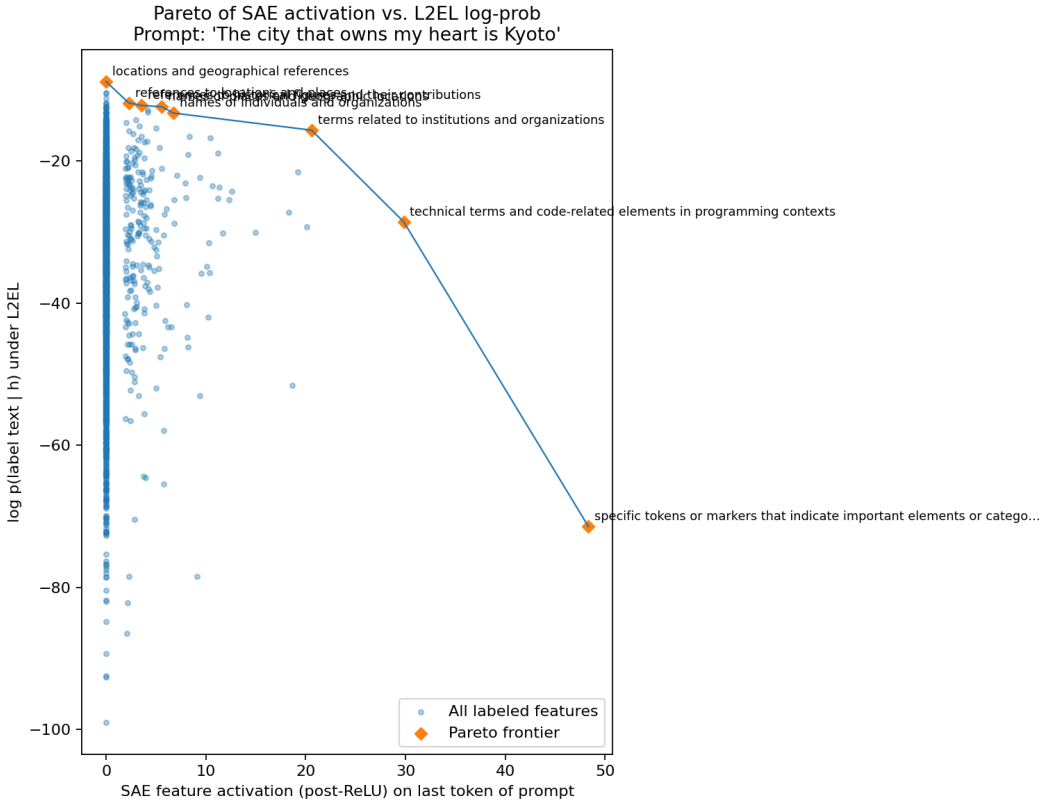

Figure 2: Pareto of SAE activation vs. L2EL likelihood for one prompt. For a single prompt, we place each labeled SAE feature as a point whose horizontal position reflects how much the feature fires on that prompt, and whose vertical position reflects the log-probability of this label under L2EL. The highlighted Pareto frontier contains features that are jointly strong on both axes.

B.3 More Examples of Generated Explanations

Table 4: Top activating SAE features and top generated explanations by L2EL on the following prompt: **This bug smells like an off-by-one error**

Top explanations by SAE activation	Top generated explanations (L2EL)
references to errors and error correction in a technical context	the word "error" in various contexts
specific tokens or markers that indicate important elements or categories within the text	terms related to errors or mistakes
technical terms related to web applications and debugging processes	references to error handling and debugging techniques
terms related to institutions and organizations	mentions of errors and their impact
references to social or cultural dynamics within communities	instances of error or errors in the code

Table 5: Top activating SAE features and top generated explanations by L2EL on the following prompt: **The most iconic villain is Darth Vader**

Top explanations by SAE activation	Top generated explanations (L2EL)
references to the "Star Wars" franchise, particularly relating to Jedi and Sith characters	references to the character "Vader"
specific technical terms and concepts related to database management and programming	instances of the word "Vader" in different contexts
technical terms related to web applications and debugging processes	phrases related to invisibility and hiding
specific tokens or markers that indicate important elements or categories within the text	mentions of the fictional character Darth Vader
technical terms and code-related elements in programming contexts	references to characters or concepts from the Star Wars franchise

Table 6: Top activating SAE features and top generated explanations by L2EL on the following prompt: **The festival I want to attend is Oktoberfest**

Top explanations by SAE activation	Top generated explanations (L2EL)
specific tokens or markers that indicate important elements or categories within the text	references to festivals or events
technical terms related to web applications and debugging processes	terms related to celebrations and events
specific technical terms and concepts related to database management and programming	mentions of "festival" and its variations
technical terms and code-related elements in programming contexts	the word "festival" indicating a large-scale event
terms related to institutions and organizations	events and festivals associated with Oktoberfest