Interpreting Sentiment Analysis in Llama 3 8B via Sparse Autoencoder Feature Extraction

Yash Chaudhary
UCID: ygc2
ygc2@njit.edu

Kaviya Sree Ravikumar Meenakshi
UCID: kr549
kr549@njit.edu

Prof. Mengnan Du
mengnan.du@njit.edu

New Jersey Institute of Technology
Course: DS-680 (Spring 2025)

April 28, 2025

Abstract

Large Language Models (LLMs) like Llama 3 exhibit impressive capabilities across various NLP tasks, including sentiment analysis, yet their internal reasoning remains largely opaque. Understanding how these models represent concepts is crucial for trust, safety, and targeted improvement. This paper investigates the internal representations of the Llama 3 8B Instruct model by applying Sparse Autoencoders (SAEs) to activations extracted during an IMDb sentiment analysis task. We successfully trained a custom SAE on activations from the Layer 24 MLP block, achieving high sparsity (average L0 norm < 1 for the selected checkpoint) while maintaining low reconstruction error. We present a preliminary qualitative analysis of features learned by the SAE by examining the text segments that maximally activate them. Our findings suggest that while many features encode low-level syntactic or structural information, some exhibit potential correlations with negation or evaluation, hinting at how sentiment-related concepts might be represented distributively at this layer. This work demonstrates the feasibility of using SAEs to probe specific layers of state-of-the-art open models like Llama 3 for task-specific interpretations.

Keywords: Large Language Models, Interpretability, Sparse Autoencoders, Llama 3, Sentiment Analysis, Feature Extraction.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance on a wide array of natural language processing tasks [1]. Models like Meta's Llama 3 [1] represent the cutting edge of open-source LLM development. However, despite their capabilities, the internal mechanisms by which these complex neural networks arrive at their outputs remain largely a "black box". This lack of transparency poses challenges for debugging, ensuring safety and alignment, and building trust in AI systems.

Interpretability research aims to bridge this gap by developing methods to understand the internal computations and representations within LLMs. Sentiment analysis, a fundamental NLP task involving classifying text based on expressed emotion or opinion, provides a concrete testbed for these methods. Understanding *how* an LLM determines if a movie review is positive or negative can shed light on its representation of valence, negation, specific topics (like acting or plot), and potentially reveal biases or failure modes.

Recent work has shown promise in using Sparse Autoencoders (SAEs) [2, 3] to decompose the high-dimensional activation vectors within LLMs into a much larger number of sparsely activating, potentially interpretable features. These features aim to represent specific concepts or functions relevant to the model's processing.

In this paper, we apply this technique to investigate the Llama 3 8B Instruct model. Specifically, our contributions are:

1. We detail the methodology for extracting internal activations from a specific layer (Layer 24 MLP output) of Llama 3 8B Instruct while performing sentiment analysis on the standard IMDb dataset [4].
2. We describe the process of training a custom SAE on these activations using NJIT's Wulver HPC cluster, achieving high sparsity while maintaining good reconstruction fidelity.
3. We present a preliminary qualitative interpretation of a subset of features learned by the SAE (using the Epoch 10 checkpoint), by analyzing the text contexts that maximally activate them.
4. We discuss the nature of the learned features, finding a prevalence of low-level linguistic patterns alongside hints of semantic relevance, contributing to the understanding of representations within this specific model layer and task context.

This work demonstrates a practical pipeline for applying SAE-based interpretability to a modern open LLM and provides initial insights into the representations learned by Llama 3 for sentiment analysis.

2 Related Work

Understanding the inner workings of LLMs is a rapidly growing field. Early approaches often involved analyzing attention patterns [5] or training linear probes on internal activation vectors to test if specific concepts are linearly decodable [6]. While insightful, these methods often provide limited, correlational understanding.

Mechanistic interpretability [7] seeks a more causal understanding by reverse-engineering specific circuits or algorithms learned by the model. Techniques include causal tracing [8] and activation patching [9]. These methods can provide deep insights but are often labor-intensive and focus on specific, narrow model behaviors.

Sparse Autoencoders (SAEs) have emerged as a promising technique for discovering interpretable features at scale [2, 3]. By training an overcomplete autoencoder with a sparsity penalty (typically L1) on model activations, SAEs aim to find a basis where each dimension corresponds to a specific, meaningful concept or circuit component. This approach builds on earlier work in sparse coding [10]. Research has demonstrated SAEs identifying features related to specific entities, syntactic structures, and abstract concepts in various models [3, 11].

Interpreting sentiment analysis specifically has been explored using various methods, including attention visualization, probing [12], and identifying influential neurons [13]. However, applying SAEs to dissect the specific features used by a modern model like Llama 3 for sentiment analysis provides a novel perspective.

The Llama family of models [1, 14, 15] are widely used open-source LLMs. While some interpretability work exists for earlier versions, applying state-of-the-art techniques like SAEs to the latest iterations, such as Llama 3 8B Instruct, particularly within a specific task context like sentiment analysis, remains an active area of investigation. Our work contributes to this by training and analyzing a custom SAE specifically for this model, layer, and task.

3 Methodology

This section details the process used to generate activations, train the SAE, and perform initial feature interpretation. All large-scale computations were performed on the NJIT Wulver HPC cluster.

3.1 Base Model and Dataset

We utilized the **Llama 3 8B Instruct** model (`meta-llama/Llama-3.1-8B-Instruct`) [1] loaded via the Hugging Face `transformers` library [16]. To manage memory constraints, the model was loaded using 4-bit quantization via the `bitsandbytes` library [17] with `bfloat16` compute precision.

The task chosen was sentiment analysis on the standard **IMDb Large Movie Review Dataset** [4] available from Hugging Face Datasets (`stanfordnlp/imdb`). This dataset contains 25,000 training reviews and 25,000 testing reviews labeled as positive or negative.

3.2 Activation Extraction

We focused on interpreting the activations from the output of the Multi-Layer Perceptron (MLP) block within a specific decoder layer of Llama 3. Based on common practices in interpretability research suggesting mid-to-late layers capture more abstract representations, we targeted **Layer 24**.

- **Target Layer Path:** The specific PyTorch module path hooked was `model.layers[24].mlp`.
- **Activation Dimension:** The output dimension of this MLP block in Llama 3 8B is $d_{act} = 4096$.
- **Hooking Mechanism:** We registered a PyTorch forward hook [18] on the target module. The hook function captured the module's output tensor, detached it from the computation graph, moved it to the CPU, and cast it to `float32` for storage.
- **Data Generation:** We processed the entire IMDb **training split** (25,000 examples) and **test split** (25,000 examples) through the Llama 3 model. Reviews were formatted using the Llama 3 Instruct chat template (see Appendix **??** for an example). Inference was performed in batches of size 4.
- **Storage:** Activations for each batch were saved as individual `.pt` files containing tensors of shape '[batch_size, seq_len, d_act]'. This resulted in 6250 files for the training split and 6250 files for the test split, stored in separate directories ('llama_activations_train/', 'llama_activations_test/') on the Wulver cluster filesystem.

3.3 Sparse Autoencoder (SAE) Architecture and Training

We implemented a simple SAE architecture consisting of a single hidden layer.

- **Architecture:** An encoder (`nn.Linear`) maps the input activations ($d_{in} = d_{act} = 4096$) to a higher-dimensional feature space (d_{SAE}). A ReLU activation function is applied to induce sparsity. A decoder (`nn.Linear`) maps the sparse features back to the original activation dimension ($d_{out} = d_{in}$). We used an **expansion factor of 4x**, resulting in $d_{SAE} = 4 \times 4096 = 16384$ features. Both linear layers included bias terms, and weights were initialized using Kaiming uniform initialization [19].
- **Loss Function:** The SAE was trained to minimize a combined loss function:

$$\mathcal{L} = \underbrace{\|x - \hat{x}\|^2}_{\text{MSE Loss}} + \lambda \quad \underbrace{\|f\|_1}_{\text{L1 Sparsity Loss}}$$

where x is the original activation vector, $\hat{x}$ is the reconstructed activation vector from the SAE decoder, f are the feature activations (output of the encoder followed by ReLU), and λ is the L1 coefficient controlling the sparsity penalty.

- **Training Data:** The SAE was trained solely on the activations extracted from the **IMDb training split**.
- **Training Details:**
  - **L1 Coefficient (λ):** Tuned experimentally. The run producing the checkpoint chosen for interpretation (Epoch 10) used $\lambda = \mathbf{1 \times 10^{-3}}$.
  - **Optimizer:** AdamW [20] with a learning rate of 1×10^{-4}.
  - **Batching:** Activation files were loaded using a custom PyTorch DataLoader collation function that concatenated the flattened token activations from multiple files. A batch size of 16 files was used.
  - **Epochs:** The model was trained for a total of 50 epochs.
  - **Environment:** Training was performed on a single NVIDIA A100 GPU on the Wulver cluster. Checkpoints were saved every 5 epochs.

3.4 Interpretation Protocol

To interpret the learned SAE features, we focused on the checkpoint saved after **Epoch 10**, which exhibited a desirable balance between low reconstruction error and high (but non-zero) sparsity based on training logs.

- **Max Activating Examples Search:** We developed a script (`find_max_activations.py`) to identify the text segments that cause the highest activation for specific feature indices. This script iterated through all activation files in the `llama_activations_train/` directory, passed the activations through the loaded Epoch 10 SAE, and used a min-heap to track the Top N (initially N=100) highest activation values for a given target feature across all tokens.
- **Context Retrieval:** The script then mapped the global token indices corresponding to these top activations back to their original example index in the IMDB training set and the token's position within that example. It extracted the activating token and a surrounding context window (15 tokens before/after) by re-tokenizing the original text.
- **Qualitative Analysis:** We manually reviewed the generated text contexts for a selected subset of 20 features to identify recurring linguistic patterns or semantic themes associated with high feature activation, forming hypotheses about the feature's function.

4 Results

4.1 SAE Performance (Epoch 10 Checkpoint)

The SAE training process converged successfully. The checkpoint selected for interpretation, Epoch 10 (trained with $\lambda = 1 \times 10^{-3}$), demonstrated strong performance metrics:

- **Reconstruction Loss (MSE):** Achieved an average MSE loss per token of approximately **0.0074**. This low value indicates that the SAE, despite its sparsity, can reconstruct the original Llama 3 activations from Layer 24 with high fidelity.
- **Sparsity (L0 Norm):** Achieved an average L0 norm of approximately **0.94 features** per token across the training data distribution. This signifies that, on average, less than one feature (out of 16384) was active above the threshold (1e-6) for any given input token activation, confirming the

high degree of sparsity induced by the L1 penalty at this stage. Individual examples analyzed later showed slightly higher but still very low operational L0 norms (typically 5-25).

Training curves illustrating the convergence of MSE loss and L0 norm over the 50 epochs are planned for inclusion (see Figure 1). The model successfully learned a sparse, reconstructive dictionary of the target activation space.

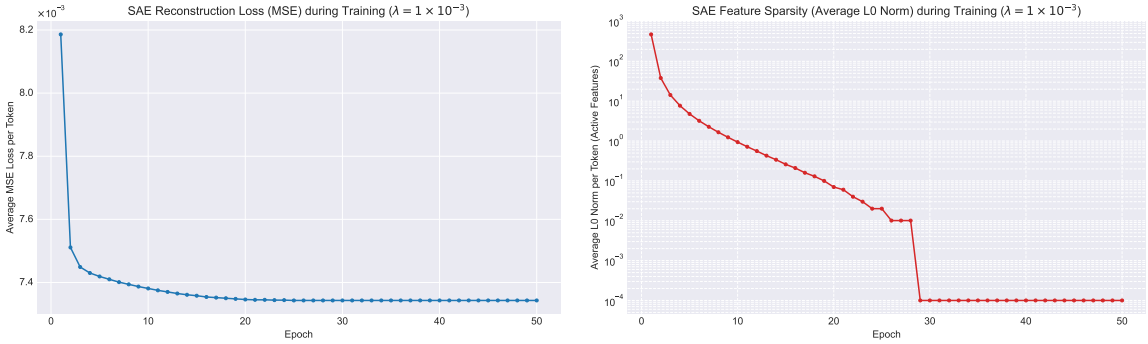

Figure 1: SAE Training Curves over 50 epochs. Left: Average MSE reconstruction loss per token. Right: Average L0 norm (active features) per token. Note the rapid decrease in L0 norm with $\lambda = 1 \times 10^{-3}$

4.2 Qualitative Feature Interpretation

We performed qualitative analysis on the Top 100 maximally activating examples for 20 features learned by the Epoch 10 SAE checkpoint. Our preliminary findings suggest a significant presence of features encoding lower-level linguistic properties, with some potentially polysemantic features showing hints of semantic relevance. We highlight interpretations for a representative subset below. (Note: Context snippets are illustrative examples from the Top 100.)

- **Feature 9988 (Low-Level / Structural):**
  - *Hypothesis:* Appears to detect frequent, low-level structural or token-based patterns, potentially related to common sequences, punctuation, capitalization, or positional information. Does not seem tied to a specific high-level semantic concept.
  - *Example Context:* `...summer reruns, so I »> ordered«< it from local library. Well, any episode...`
- **Feature 6225 (Punctuation / Common Tokens):**
  - *Hypothesis:* Primarily responsive to punctuation (especially commas, periods) and common function words or sub-word units, likely involved in basic structural parsing.
  - *Example Context:* `...aura »> that«< he's become known for in the past decade. As much...`
- **Feature 8435 (Punctuation / Evaluation Context):**
  - *Hypothesis:* Activates strongly on quote punctuation (") and shows sensitivity to contexts involving evaluative language (e.g., `incredible`, `bad`, `hateful`) or judgment, potentially regardless of specific valence. Mixed with function word triggers.
  - *Example Context:* `...<|begin_of_text|>An »> incredible«< little English film for so many reasons. First...`
- **Feature 9164 (Syntax / Negation / Topic Hint):**

- *Hypothesis:* Predominantly detects common function words (`and`, `is`, `the`) and punctuation, but shows a consistent secondary activation pattern on the negation token `not` and occasionally on topic words like `comedy`. Suggests a primary syntactic role with potential weak sensitivity to negation/topic.
- *Example Context:* `...boring because Jane is »> not«< passionate enough, or because there are not...`

- **Feature 10410 (Mixed / Potential Negative Valence):**
  - *Hypothesis:* Polysemantic, activating on diverse token types. Shows a noticeable tendency to activate on words within negative contexts (`flat`, `fails`, `vulgar`, `evil`, `problem`), suggesting a weak correlation with negative concepts, though heavily mixed with other triggers.
  - *Example Context:* `...line and goes »> flat«< . The acting for one is appalling! Here we...`

- **Feature 8178 (Mixed / Negation / Nouns):**
  - *Hypothesis:* Polysemantic. Shows sensitivity to negation (`nobody`) and potentially descriptions involving agents or concepts (`characters`, `directing`, `ways`), alongside common syntactic triggers.
  - *Example Context:* `...glad »> nobody«< tried to put them all in a single dish....`

- **Feature 1986 (Common / Proper Nouns):**
  - *Hypothesis:* Primarily linked to common function words and structure, but exhibits occasional strong activation on proper nouns (`Arthur`, `Castro`, `Albert`) or common content words (`people`), suggesting a potential role in entity recognition or common discourse patterns.
  - *Example Context:* `...first experience of »> Arthur«< Askey, I have to admit I was very impressed...`

Interpretation Summary: This initial analysis of features from the highly sparse Epoch 10 SAE indicates that many learned features in Llama 3 Layer 24 MLP capture fine-grained linguistic details related to syntax, punctuation, common tokens, and sub-word structure. While some features display weak correlations with semantic elements like negation or evaluation, cleanly interpretable, monosemantic features directly representing sentiment were not immediately prevalent in the analyzed subset. This suggests sentiment may be represented distributively or rely on combinations of these lower-level features at this stage of the model.

5 Discussion

The successful training of a SAE on Llama 3 8B activations demonstrates the technical feasibility of applying this method to modern open LLMs. Achieving high sparsity (average $L_0 < 1$ for the selected checkpoint) while maintaining low reconstruction error confirms that the high-dimensional activation space can be effectively decomposed into a sparse dictionary basis.

Our preliminary qualitative interpretation of features learned at Layer 24 suggests a predominance of features capturing low-level linguistic patterns, syntax, and common token structures. This observation aligns with hypotheses in interpretability research proposing that intermediate layers primarily focus on building structural representations, while higher layers synthesize more abstract semantic meanings. The relative absence of clearly interpretable "sentiment features" in our initial sample does not imply that

sentiment is not represented; rather, it may be encoded in a distributed fashion across multiple features, reliant on combinations of lower-level linguistic cues. Alternatively, such features may exist but were not included in the subset we analyzed, or they may require different hyperparameters to emerge more distinctly.

The polysemy observed in several features (e.g., 10410, 8178) highlights a known challenge in SAE interpretation [3]. Although sparsity promotes disentanglement, individual dictionary elements can still capture multiple, sometimes unrelated, concepts. Future work might leverage techniques such as analyzing feature correlations or employing alternative SAE architectures to further disentangle polysemantic features.

Future Work: Several promising directions exist for future research. Analyzing features across different layers (both earlier and later) could reveal how representations evolve throughout the model. Training SAEs with varied hyperparameters (e.g., adjusting the L1 coefficient to target slightly higher L0 norms like 10–50, or exploring different expansion factors) may yield more semantically distinct features. Applying quantitative methods—such as correlating feature activations with sentiment scores across the full dataset or performing ablation studies to causally intervene on individual features—would strengthen our understanding of feature functionality. Investigating feature interactions, developing automated interpretation or clustering pipelines, and extending this methodology to different tasks and models are also important directions for future work.

6 Conclusion

We presented a successful application of Sparse Autoencoders (SAEs) to interpret internal representations of the Llama 3 8B Instruct model performing sentiment analysis on the IMDb dataset. By training a custom SAE on activations from the Layer 24 MLP output, we obtained a sparse dictionary basis (Avg L0 < 1 for the analyzed Epoch 10 checkpoint) that accurately reconstructed the original activations. Preliminary qualitative analysis of the text contexts maximally activating a subset of 20 learned features suggests a prevalence of features encoding low-level syntactic, punctuation, and common token patterns, with some exhibiting mixed signals or weak semantic correlations (e.g., related to negation or evaluation). While clear high-level sentiment features were not immediately identified in this initial set, our work demonstrates a viable pipeline for probing Llama 3 using SAEs and provides initial insights into the nature of representations at this layer, suggesting sentiment might be encoded distributively or built upon these finer-grained linguistic features. Further analysis across more features and layers is warranted to build a more complete picture of Llama 3's internal workings.

Acknowledgments

We thank Prof. Mengnan Du for their guidance.

References

[1] AI at Meta. *Llama 3 Model Card*. `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`. 2024.

[2] Anthropic. *Towards Monosemanticity: Decomposing Language Models With Dictionaries*. [URL or ArXiv ID]. 2023.

[3] Anthropic. *Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet*. [URL or ArXiv ID]. 2024.

[4] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[5] Sarthak Jain and Byron C. Wallace. *Attention is Not Explanation*. 2019. arXiv: 1902.10186 [cs.CL].

[6] John Hewitt and Percy Liang. “Designing and Interpreting Probes with Control Tasks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 2733–2743.

[7] Chris Olah et al. “Zoom In: An Introduction to Circuits”. In: *Distill* (2020). DOI: 10.23915/distill.00024.001. URL: https://distill.pub/2020/circuits/zoom-in.

[8] Kevin Meng et al. “Locating and Editing Factual Associations in GPT”. In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. 2022.

[9] Kevin Wang et al. “Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 small”. In: *arXiv preprint arXiv:2211.00593* (2022).

[10] Bruno A Olshausen and David J Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. In: *Nature* 381.6583 (1996), pp. 607–609.

[11] Neel Nanda. “Actually, Othello is Two Circuits!” In: (2023).

[12] Timothy Niven and Hung-Yu Kao. “Probing Neural Network Comprehension of Natural Language Arguments”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 4658–4664.

[13] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. “Learning to Generate Reviews and Discovering Sentiment”. In: *arXiv preprint arXiv:1704.01444* (2017).

[14] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[15] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[16] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[17] Tim Dettmers et al. *LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale*. 2022. arXiv: 2208.07339 [cs.LG].

[18] PyTorch Team. *PyTorch Hooks Documentation*. https://pytorch.org/docs/stable/generated/torch.nn.modules.module.register_forward_hook.html. Accessed 2025.

[19] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[20] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *arXiv preprint arXiv:1711.05101* (2017). URL: https://arxiv.org/abs/1711.05101.