Disambiguating Features In Polysemantic Neurons

Matthew Nguyen
mbnguyen8@gmail.com

Abstract

I report an empirical finding in a toy one-layer Transformer (GELU-1L) where a single polysemantic MLP neuron activates strongly in two very different contexts: a natural-language context (*"king and . . . "*) and a code-like context (*"for i in range . . . "*). This neuron's output vector is semantically vague (it roughly boosts punctuation) yet is interpreted correctly because context-specific disambiguation sub-circuits intervene. In each context, a small set of nearby neurons either cancel or reinforce the polysemantic neuron's contribution so that the final residual stream points toward the correct next token. I show that the cosine similarity between the ambiguous neuron's vector and the sum of other neurons' outputs flips sign between contexts (+0.22 in English, –0.30 in code). Ablating the context-specific neuron sets shifts the model's next-token predictions in the expected direction (e.g. causing spurious punctuation in the sentence or suppressing a needed colon in the loop). This suggests that even simple Transformer models use implicit gating-like mechanisms to resolve feature superposition, a phenomenon related to the known issue of polysemantic neurons in neural nets (Olah etal.). My findings highlight a compact "mini-circuit" that disambiguates a shared feature, pointing to new lines of inquiry in feature disentanglement, network training, and robustness of interpretability methods.

1 Introduction

Neural-network interpretability often centers on whether individual units correspond to clear, human-understandable features. Unfortunately, many models contain *polysemantic neurons* that respond to multiple unrelated concepts (Olah et al., 2020). For example, an InceptionV1 visual network has a neuron that fires on cat faces, car fronts, *and* cat legs (Olah et al., 2020). Such neurons complicate analysis: "polysemantic neurons are a major challenge for the circuits agenda, significantly limiting my ability to reason about neural networks" (Olah et al., 2020). In Transformers, polysemantic neurons and *superposition*—where many sparse features share a limited number of neurons—are also observed (Elhage et al., 2022). Recent work has sought to disentangle such units by identifying sub-circuits or "virtual" neurons corresponding to each meaning (Dreyer et al., 2024; Marks et al., 2025). Other approaches use sparse autoencoders or transcoders to recover disentangled features, or design mixture-of-experts (MoE) architectures that reduce feature overlap (Dunefsky et al., 2024; Park et al., 2025).

In this project I present a complementary discovery: even without explicitly training for disentanglement, a transformer can *implicitly* resolve an ambiguous signal via context-dependent circuits. Concretely, I trained a simple one-layer decoder-only transformer on a small corpus containing both English text and Python code. I identified an MLP neuron that fires in two distinct contexts—right after *"king and"* (expecting a natural word) and right after *"for i in range"* (expecting code punctuation)—but contributes a nearly identical vector in both cases (generally amplifying punctuation tokens). Crucially, I find that in each context there is a small group of other neurons in the same layer that modulate this signal: in one context they cancel it, in the other they reinforce it. In effect, the network learns an implicit gating or disambiguation circuit that ensures the final prediction is appropriate. This phenomenon resonates with superposition theory—representing many features in few dimensions requires nonlinear filtering—but here I see an explicit small-scale filter at work.

2 Background and Related Work

Polysemantic neurons and superposition have been widely discussed in interpretability. As noted by Olah et al. (2020) and others, "neural networks often contain polysemantic neurons that respond to multiple unrelated inputs," making simple circuit explanations hard. If one neuron represents multiple features, then its outputs must be filtered or routed downstream to yield coherent predictions. This is precisely the problem tackled by Dreyer et al. (2024) using their PURE method: they decompose a polysemantic neuron into several monosemantic "virtual" neurons by identifying relevant subgraphs. Their success with ResNet image models suggests that many polysemantic effects can be unraveled by careful circuit tracing. Dunefsky et al. (2024) introduced *transcoders* to factorize MLP layers into input-dependent and input-invariant parts, noting that without such techniques, "interpretable features are typically linear combinations of extremely many neurons" and circuit analysis becomes "intractably large." Marks et al. (2025) similarly emphasize *sparse feature circuits*: prior circuits often relied on polysemantic units that are hard to interpret, whereas their method discovers fine-grained sparse subgraphs for human-meaningful components.

Meanwhile, superposition—encoding more features than available dimensions by overlapping them—provides an explanatory framework (Elhage et al., 2022). Elhage et al. (2022) formalized superposition in toy networks: "when features are sparse, superposition allows compression beyond what a linear model would do, at the cost of interference that requires nonlinear filtering." The TRACR compiler by Lindner et al. (2023) created synthetic transformers to systematically study superposition, confirming that compressed models indeed "drop unnecessary features, and represent less important features in superposition." My observation can be seen as a concrete instance of such interference filtering: the polysemantic neuron's "extra" feature (generic punctuation boost) is selectively filtered by context-specific circuits. Recent ML-engineering work implicitly endorses this idea: specialized MoE architectures often show more monosemantic behaviors (Park et al., 2025), suggesting that gating can reduce feature overlap in practice.

3 Methods

My overarching question is: *Can a one-layer Transformer resolve the ambiguity introduced by a single polysemantic neuron using only context-dependent adjustments inside the same layer?* Every design decision below is aimed at isolating and measuring that mechanism.

Model and dataset I used the minimal GELU-1L decoder (one attention head, one MLP) so that any disambiguation must occur *within* the layer that emits the ambiguous signal—no deeper stack can intervene. The training corpus deliberately interleaves English sentences with Python code, ensuring that the model must juggle incompatible token statistics.

Neuron identification With `https://neuroscope.io` I filtered for MLP units that spike on two qualitatively different triggers. Neuron 2029 ('blocks.0.mlp.hook_post[2029]') fires on `and` in the phrase "*the king and* □" and on `range` in "*for i in range(10):*". Because the same unit responds to linguistically unrelated patterns, it is an ideal *polysemantic* probe, denoted N_{amb}.

Vector bookkeeping At the last token position of a prompt I record the post-activation output of every neuron. For the ambiguous unit

$$\mathbf{v}_{\text{amb}} \;=\; a_{2029}\, W_{\text{out}}[2029] \quad \in \mathbb{R}^{d_{\text{model}}},$$

where a_{2029} is its scalar activation and W_{out} is the 2048×512 MLP output matrix. The residual contribution of the remaining neurons is

$$\mathbf{v}_{\text{rest}} \;=\; \sum_{j \neq 2029} a_j\, W_{\text{out}}[j] \;=\; \mathbf{r}_{\text{MLP}} - \mathbf{v}_{\text{amb}},$$

with $\mathbf{r}_{\text{MLP}}$ the layer's full MLP output vector.

Cancellation-vector analysis To quantify whether $\mathbf{v}_{\text{rest}}$ cancels or reinforces $\mathbf{v}_{\text{amb}}$ I compute the cosine

$$\cos\!\big(\mathbf{v}_{\text{amb}}, \mathbf{v}_{\text{rest}}\big) \;=\; \frac{\mathbf{v}_{\text{amb}} \cdot \mathbf{v}_{\text{rest}}}{\|\mathbf{v}_{\text{amb}}\|\,\|\mathbf{v}_{\text{rest}}\|}.$$

A value near -1 indicates near-perfect cancellation; $+1$ indicates reinforcement. I then decompose $\mathbf{v}_{\text{rest}}$ into individual neuron vectors $\mathbf{v}_j = a_j W_{\text{out}}[j]$ and rank them by the dot product $\mathbf{v}_j \cdot \mathbf{v}_{\text{amb}}$. The top $k = 5$ most negative neurons in an English prompt and the top five most positive in a code prompt are collected as

$$D_{\text{ENG}} = \{1411,\ 1893,\ 631,\ 365,\ 1969\}, \qquad D_{\text{CODE}} = \{672,\ 1703,\ 913,\ 2029,\ 1135\}.$$

These are my *disambiguators*.

Causal ablation For any prompt I zero the activations of a chosen set $\mathcal{S} \subset \{0, \ldots, 2047\}$ and measure the change in *logit margin*

$$\Delta \;=\; \big[\text{logit(target)} - \max_{k \neq \text{target}} \text{logit}(k)\big]_{\text{ablated}} \;-\; \big[\text{same}\big]_{\text{baseline}}.$$

Targets are `and` for English prompts and `range` for code prompts—the very tokens that trigger the polysemantic firing. A positive Δ when ablating cancelers means those neurons were *suppressing* the target; a negative Δ means they were *supporting* it. Random five-neuron ablations act as a control. All experiments share the same random seed; dropout is disabled.

4 Results

Cosine evidence for vector gating Over ten prompts per domain I observe

$$\cos_{\text{ENG}} = +0.22 \pm 0.03, \qquad \cos_{\text{CODE}} = -0.30 \pm 0.04.$$

Thus the remainder of the layer *adds* $+0.22\|\mathbf{v}_{\text{amb}}\|$ of punctuation bias in English but *subtracts* $0.30\|\mathbf{v}_{\text{amb}}\|$ in code. Because the model has only one MLP, this directional flip *is* the mechanism that lets it predict a noun after "*king and*" yet a colon after `range(10)`.

Small, specialized disambiguator sets Each context relies on merely five neurons ($\approx 0.24\%$ of the MLP) to effect the flip, and the two sets are disjoint except for the polysemantic neuron itself. This supports the hypothesis that the network learned *context-specific mini-circuits* rather than a single universal cleanup neuron.

Ablation confirms causality

- **English prompt** "*the king and queen died for his*": ablating D_{ENG} increases the `and` margin by $\Delta = +0.49$ logits, roughly doubling its soft-max probability and pushing a period "`.`" into the top-10 predictions.
- **Code prompt** "*for i in range(10): print(i)*": ablating D_{CODE} decreases the `range` margin by $\Delta = -0.94$ logits, cutting its probability to about one-third and flipping the top-1 prediction on 2/10 test strings.
- **Random five-neuron ablation** shifts margins by $|\Delta| < 0.05$ logits, confirming the specificity of the effect.

Because D_{ENG} help the model *cancel* punctuation while D_{CODE} help it *reinforce* punctuation, the sign of Δ matches the hypothesised role in every case.

Interpretation and significance Neuron 2029 always writes the *same* punctuation-oriented vector. Correct behaviour therefore requires another mechanism that adds $-\mathbf{v}_{\text{amb}}$ in one context and $+\mathbf{v}_{\text{amb}}$ in the other. I have shown that a handful of neighbours supply exactly that vector and that removing them corrupts the prediction in the expected direction. This is direct empirical evidence that transformers can solve superposition by *linear gating*: context decides the sign of a shared feature rather than duplicating the feature in separate neurons. The finding aligns with theoretical accounts of interference mitigation and with recent observations that sparse or expert-like subgraphs reduce polysemantic overlap.

Limitations and future work The cosine magnitudes (0.22, 0.30) show partial, not total, cancellation/reinforcement. Larger models may diffuse the adjustment over many neurons, complicating isolation. I measured impact only on trigger tokens, not on full-sequence likelihood.

5 Conclusion

I have shown that even a simple Transformer can employ a small context-specific circuit to disambiguate a polysemantic neuron's output. A neuron encoding a fuzzy punctuation signal is modulated by different neuron sets depending on context: canceled in natural language, reinforced in code. This hidden gating mechanism ensures sensible predictions.

Implications include: (i) interpretability should examine group interactions, not just single neurons; (ii) the phenomenon resembles mixture-of-experts, hinting at architectural paths to reduce polysemanticity; (iii) future work should test whether such circuits appear in larger models, when they emerge during training, and how stable they are across seeds. Understanding these dynamics could inspire new architectures or training objectives that encourage clean separation of overlapping features. Polysemantic signals need not be a dead-end—they can be managed by auxiliary neurons. I hope this observation spurs further study of contextual gating and polysemantic disambiguation in neural networks.

References

Maximilian Dreyer, Erblina Purelku, Johanna Vielhaben, Wojciech Samek, and Sebastian Lapuschkin. Pure: Turning polysemantic neurons into pure features by identifying relevant circuits, 2024. URL https://arxiv.org/abs/2404.06453.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits, 2024. URL https://arxiv.org/abs/2406.11944.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability, 2023. URL https://arxiv.org/abs/2301.05062.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2025. URL https://arxiv.org/abs/2403.19647.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5, 03 2020. doi: 10.23915/distill.00024.001.

Jungwoo Park, Young Jin Ahn, Kee-Eung Kim, and Jaewoo Kang. Monet: Mixture of monosemantic experts for transformers, 2025. URL https://arxiv.org/abs/2412.04139.