

# Pinpointing Latent Planning in Language Models with Lightweight Mechanistic Methods

Stanford CS224N Custom Project

**Nick Rui**

Stanford University  
nickrui@stanford.edu

**Nicole Ma**

Stanford University  
manicole@stanford.edu

**Harsh Singh**

Stanford University  
singhh@stanford.edu

## Abstract

Autoregressive language models predict one token at a time, but do they plan ahead during generation? We investigate this question using lightweight and scalable mechanistic methods on Qwen3-32B and Gemma-3-27B without relying on training transcoders. In general text generation, we find little evidence of anticipatory representations, but in rhyming couplet generation language models show strong planning signal, specifically at the newline token in intermediate layers. Causal interventions confirm this signal is causally active in Gemma-3, but reveal weaker causal deployment in Qwen3, suggesting architectural differences in how planning representations are used. Our results demonstrate that lightweight probing and interventions can pinpoint *when* and *where* latent planning occurs—identifying both the conditions under which it arises (e.g., rhyming couplet generation) and its precise location within the network (specific token positions and layer ranges). This offers a scalable alternative to transcoders for analyzing latent planning.

## 0 Key Information to include

Our TA mentor is Julie Kallini. Other TAs we have corresponded with include Tristan Thrush, Luke Bailey, Ahmed Ahmed, and Qinan Yu. We have also corresponded with Emmanuel Ameisen from Anthropic over email. We are not sharing project. Using two late days (Nick has six remaining).

## 1 Introduction

Autoregressive language models generate text one token at a time, yet routinely produce outputs requiring long-range coherence. A natural question arises: do language models plan ahead during generation, and if so, where and how does this planning emerge within the model?

*Latent planning* in language models is the formation of internal representations of future outputs that causally shape token generation. Latent planning is important because it is un verbalized. Unlike chain-of-thought reasoning, where intermediate steps are directly observable, latent planning occurs entirely within a model’s hidden activations. If models are planning in ways invisible to external observers, standard behavioral evaluations may systematically miss this capacity. Understanding the mechanisms underlying generation is therefore both a scientific and safety-critical problem.

Reverse-engineering neural networks is difficult and expensive. Activations are high-dimensional, dense, and polysemantic, meaning individual neurons fire for multiple unrelated concepts, and the same computation can be distributed across many components in ways that resist simple decomposition. Prior work investigating latent planning [1, 2] uses transcoders [3] to build feature circuits [4, 5], but these require substantial compute to train, do not trivially scale to new and larger models, and have only been applied to closed-source models such as Claude Haiku 3.5 [6]. Current methods thus face both a scalability and replicability issue.

We address this by investigating latent planning using lightweight methods—linear probing, activation patching, and steering vectors—that require far less compute and resources compared to transcoders

and generalize readily to new architectures. Focusing on poem generation, we analyze open source models: Qwen3-32B [7], a general language model, and Gemma-3-27B [8], a multimodal model shown to have the greatest rhyming capabilities compared to similar open-source models [9].

**Summary of Contributions.** We show that simple mechanistic tools reveal evidence of latent planning, offering a more scalable approach for studying latent planning in language models with large parameter sizes. In addition, we provide a position and layer-wise account of where latent planning occurs in the network to compare planning mechanisms between language models.

## 2 Related Work

**Explicit Planning in Language Models.** Chain-of-thought prompting [10] demonstrates that language models can plan when allowed to externalize intermediate reasoning steps. Pfau et al. [11] show that even semantically meaningless filler tokens improve reasoning performance, suggesting useful computation occurs independent of token content. More broadly, Hao et al. [12] train models to reason in a continuous latent space, enabling search strategies inaccessible to standard chain-of-thought. These results motivate our focus on latent planning, which is implicit in nature.

**Latent Planning in Neural Networks.** A growing body of work shows that neural networks trained on next-step prediction can develop internal planning representations in structured domains. McGrath et al. [13] find that chess concepts emerge in AlphaZero’s internal layers despite no explicit supervision. Li et al. [14] show that a transformer trained to predict legal Othello moves develops an emergent board representation, later shown by Nanda et al. [15] to be linear and causally manipulable. Jenner et al. [16] provide the first mechanistic evidence of learned look-ahead in Leela Chess Zero, demonstrating via activation patching that future-move representations are causally important for move selection. Together, these results establish that planning-like algorithms can emerge in game-playing networks, motivating the question of whether language models elicit similar mechanisms.

**Methods in Mechanistic Interpretability.** Probes are standard tools for assessing what information is encoded in hidden states. Burns et al. [17] use unsupervised probes to elicit latent truth representations, and Zou et al. [18] identify linear directions associated with safety-relevant concepts and show these directions can steer model behavior. However, Hewitt and Liang [19] show that sufficiently flexible probes can achieve high accuracy by memorizing labels rather than reflecting genuine representations, motivating the use of causal interventions alongside probing. A key causal tool is activation patching: Meng et al. [20] formalize it as causal tracing to localize factual associations in GPT, and Wang et al. [21] apply it to identify the indirect object identification circuit in GPT-2 small. Steering vectors offer a complementary intervention: Turner et al. [22] compute mean-difference vectors from contrasting prompt pairs and add them during inference to shift model behavior, and Arditì et al. [23] show that refusal in aligned models is mediated by a single linear direction. At a much more expensive scale, sparse autoencoders [24] and transcoders [3] transform dense activations into interpretable features, enabling feature-level circuit analysis [4]. Lindsey et al. [1] apply these methods to study latent planning in Claude 3.5 Haiku. However, transcoders must be trained on billions of tokens from the target model’s activations, effectively requiring a second training run, making them impractical for larger open-source models without significant infrastructure. We combine probing, activation patching, and steering vectors to provide layer-wise causal granularity on this finding without any need for heavyweight compute or infrastructure.

## 3 Probing Evidence of Planning

### 3.1 Setup and Notation

Let  $\mathcal{V}$  denote the vocabulary and  $\Delta(\mathcal{V})$  the set of all probability distributions over  $\mathcal{V}$ . Consider an autoregressive transformer language model  $M$  with  $L$  layers and hidden dimension  $d$ . For a single model completion, let  $\mathbf{x} = (x_1, \dots, x_{|\mathbf{x}|})$  denote the input context token sequence, where each  $x_i \in \mathcal{V}$ . The language model generates a completion  $\mathbf{y} = (y_1, \dots, y_{|\mathbf{y}|})$  where  $y_i \sim M(\mathbf{x}, y_{<i}) \in \Delta(\mathcal{V})$ .

Throughout this paper, we adopt a relative positioning scheme where position 0 refers to the last context token  $x_{|\mathbf{x}|}$  and position  $i$  maps to token  $x_{|\mathbf{x}|+i}$  if  $i \leq 0$  and to  $y_i$  if  $i > 0$ . During a forward

pass, let  $\mathbf{h}_{\ell,i} \in \mathbb{R}^d$  denote the hidden state vector after the  $\ell$ -th transformer block at position  $i$ , i.e. the sum of all layer contributions up to and including layer  $\ell$ . Note that due to causal masking,  $\mathbf{h}_{\ell,i}$  is a function of only the tokens at positions  $\leq i$ .

A *probe* is a classifier  $f_\theta : \mathbb{R}^d \rightarrow \Delta(\mathcal{V})$  parameterized by  $\theta$  that predicts a target token directly from a single hidden state. In this work, we use linear probes of the form

$$f_{(W,b)}(\mathbf{h}) = \text{softmax}(W\mathbf{h} + \mathbf{b}),$$

where  $W \in \mathbb{R}^{V \times d}$  and  $\mathbf{b} \in \mathbb{R}^V$ . Linear probes are trained by minimizing the cross-entropy loss

$$\mathcal{L}(W, b) = -\mathbb{E}_{(\mathbf{h}, y)} [\log f_{(W,b)}(\mathbf{h})_y],$$

where  $y \in \mathcal{V}$  is the ground truth target token and  $f_{(W,b)}(\mathbf{h})_y$  is the probability assigned to token  $y$ .

Probes are trained in a two-step pipeline. First, an activation-to-token dataset is constructed by greedily sampling a model completion  $\mathbf{y}$  from a seed context  $\mathbf{x}$ , extracting hidden states  $\mathbf{h}_{\ell,i}$ , and aggregating pairs  $(\mathbf{h}_{\ell,i}, y_t)$  for  $t > i$ . We call  $t - i$  the *look-ahead distance*, measuring how far into the future the probe is tasked to predict. For each layer  $\ell$ , a probe is fit and evaluated on relevant pairs in the dataset. Probes are trained with AdamW [25] with learning rate 0.0001, weight decay 0.001, for 10 epochs. See Appendix A for model-specific details regarding tokenization and architecture.

### 3.2 Probing on General Text

To investigate general planning capabilities, we probe models on general text generation. For each layer  $\ell$  and look-ahead distance  $k$ , we train a linear probe  $f^{(\ell,k)}$  to predict  $y_{i+k}$  from  $\mathbf{h}_{\ell,i}$ . We compare probe performance to baseline unigram and bigram models trained on the same model completions. The unigram model naively outputs the highest frequency token in the model completions, and the bigram model is trained on local token co-occurrences (pairs of tokens  $k$  apart). This verifies that the probes are learning more information than the corpus of generated tokens itself provides.

We randomly sample 1,200 (1,000 train, 200 validation) token sequences of varying length between 32-64 from The Pile [26] and greedily sample model completions of length 64 tokens to build the activation-to-token dataset. We evaluate on top-1 and top-5 accuracy. We report top-5 accuracy in Figure 1 and leave top-1 accuracy results in Appendix B.

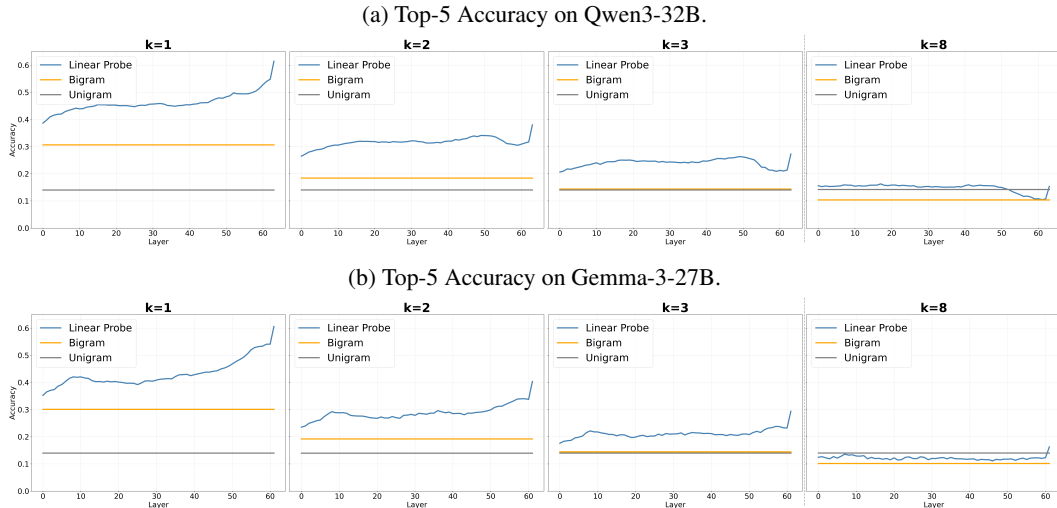


Figure 1: Accuracy of linear probes to predict  $k$  tokens ahead from hidden states on general text.

Results are similar between the two models. We find that probe performance decreases monotonically with  $k$ , suggesting that hidden states in general text generation encode little information about future tokens beyond what is explained by local statistics. For  $k = 1$ , the linear probe substantially outperforms both unigram and bigram baselines across all layers, with top-5 accuracy peaking at

around 0.60.<sup>1 2</sup> However, probe accuracy substantially drops as  $k$  increases, and by  $k = 8$  probe performance is indistinguishable and sometimes worse than both baselines across all layers.

This suggests that in general text generation, both Qwen3-32B and Gemma-3-27B do not form anticipatory representations of tokens more than a few steps ahead, consistent with predominantly local, next-token prediction behavior rather than long-range latent planning.

### 3.3 Probing on Rhyming Couplets

In contrast to general text, rhyming couplet generation is a setting where planning is advantageous. For example, freestyle rappers often plan the rhyming word at the end of a line first, then work backward to produce the words preceding it. We hypothesize language models take a similar approach. To investigate this task-specific planning, we probe on rhyming couplet generation. The input context  $\mathbf{x}$  contains the rhyme instruction and first line of a rhyming couplet. The model completion  $\mathbf{y}$  is the second line. Here,  $y_{|y|}$  (which completes the rhyme scheme) is always the target token. For each layer  $\ell$  and position  $i$ , we train a linear probe  $f^{(\ell,i)}$  to predict  $y_{|y|}$  from  $\mathbf{h}_{\ell,i}$  (Figure 2).

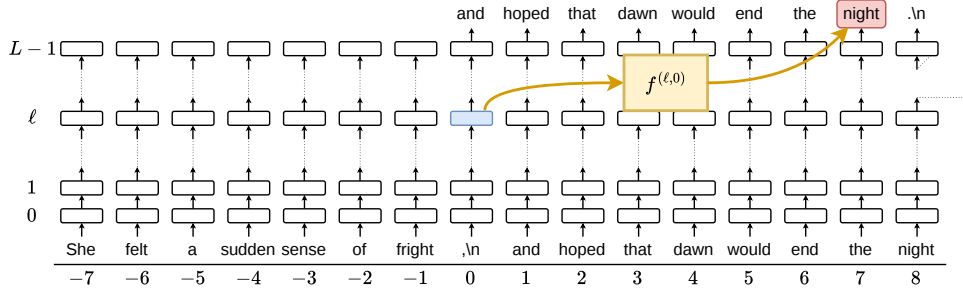


Figure 2: A linear probe  $f^{(\ell,0)}$  trained to predict the rhyming token *best* from model activations at layer  $\ell$  at the newline token position ( $i = 0$ ). Positions are indexed relative to the newline token.

We first synthetically generate 1,200 (1,000 train, 200 validation) half-couplets with Claude Sonnet 4.6 [27], prompting for diverse subjects and rhyme schemes. For example:

"A rhyming couplet:\n She felt a sudden sense of fright,\n"

To build the activation-to-token dataset, we greedily sample model completions for the second line of the couplet, continuously resampling with temperature until we have a rhyming completion. We then train and evaluate probes on four metrics: top-1, top-5, top-1 rhyme, and top-5 rhyme accuracy. We define top- $k$  rhyme accuracy as the proportion of times one of the probe’s top- $k$  predicted token rhymes with the ground truth target token, as determined by the CMU Pronouncing Dictionary [28]. Rhyme accuracy is a softer metric than exact token accuracy, since it rewards the probe for correctly identifying the rhyme family even when the exact token is missed. We include this metric since language models engaged in rhyme planning need not commit to a specific word, only to a rhyme family. We report top-5 accuracy in Figure 3 and complete results in Appendix B<sup>3</sup>.

We find that probes trained on activations at the newline token position ( $i = 0$ ) significantly outperform later positions ( $i > 0$ ) in middle layers, reversing the pattern observed in general text generation where proximity to the target improves predictability. Rhyming words were on average around 8 tokens ahead of the newline position, yet probe accuracy at  $i = 0$  far exceeds what we observed for

<sup>1</sup>Note this is over all 151,699/262,208 tokens in Qwen/Gemma’s vocabulary (see Appendix A).

<sup>2</sup>For  $k = 1$  one may expect the probe on the last layer to perform perfectly, since activations are a single linear transformation (the LM head) away from next-token logits. Though we observe a jump in accuracy at the last layer, we do not reach perfect probe performance due to limited training time, data, and the fact that both models apply an RMSNorm layer before the LM head, so logits are *not* linearly decodable from  $\mathbf{h}_{L-1}$ .

<sup>3</sup>We omit  $n$ -gram baselines here because (1) the general text experiment already serves as a negative control, (2) the small, prompt-specified rhyme family set causes  $n$ -grams to spuriously learn rhyme family frequencies, and (3) our primary goal is to compare probe accuracy *across layers and positions*.

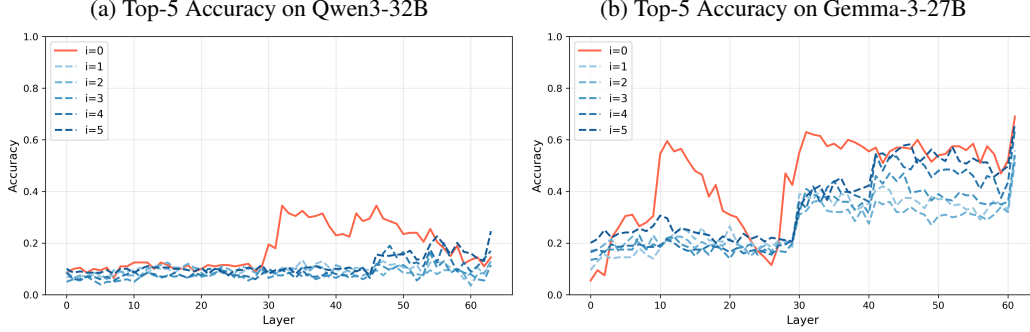


Figure 3: Accuracy of linear probes to predict the rhyming word from hidden states at position  $i = 0$  (newline token) compared to subsequent positions  $i > 0$  closer to the rhyming word.

$k = 8$  in general text. This suggests the elevated probe performance reflects a task-specific planning representation selectively constructed when structural constraints make latent planning useful.

Notably, the shape of  $i = 0$  probe performance across layers deviates strikingly from  $i > 0$  probes across both models. In Qwen3-32B, the  $i = 0$  probe peaks in layers 30–50, where it substantially outperforms all  $i > 0$  probes. In Gemma-3-27B, the  $i = 0$  probe shows two distinct peaks diverging from the other probes: the first around layers 10–20 and a stronger second around layers 30–40. In both cases, the  $i > 0$  probes only begin to approach  $i = 0$  performance in later layers.

These results lead us to hypothesize these models use the newline token as a *planning site*, a position where rhyme information is planned before generation of the second line begins. To confirm the importance of this token, we run a small ablation experiment comparing greedy rhyming success with and without the newline token in the prompt, seeing a degradation in rhyming success from 67.25% to 56.83% in Qwen3-32B and 90.33% to 61.08% in Gemma-3-27B (see Appendix C).

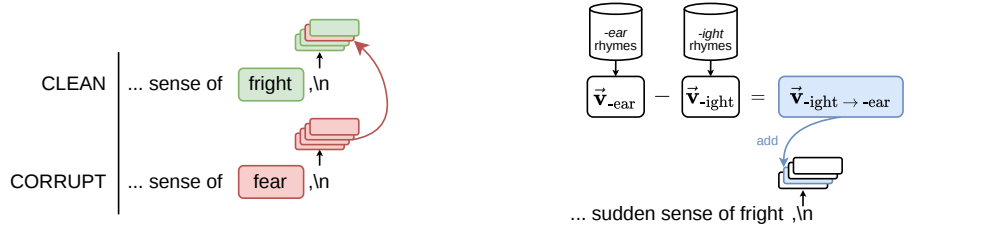
### 3.4 Discussion

These results suggest that latent planning is task-dependent rather than a general property of autoregressive generation. The contrast between probes trained on general text and rhyming couplets points toward planning representations being selectively constructed when task structure makes future tokens predictable. The localization of this signal to the newline token is consistent with [1], who reach the same hypothesis using transcoders. Our results suggest a similar role for the newline token in open source models obtained with substantially cheaper methods. In addition, differences between models in probe performance further suggest that the degree to which planning is organized around a single token position may vary with architecture or training, warranting further investigation.

However, probing is inherently observational. High probe accuracy only indicates information about future tokens is *decodable* (linearly, in this case) from hidden states, but does not establish that this information is actually used by the model during generation [2]. For example, an alternative explanation is the  $i = 0$  signal reflects passive accumulation of rhyme-relevant context via attention to the first line (say, from the tokens *rhyming* and *fright*), rather than active planning. The general text comparison weakens this objection, since analogous attention patterns exist in general text yet produce no comparable planning signal. Still, probing alone cannot rule it out.

## 4 Causal Interventions

While probing reveals that future token information is *encoded* in hidden states, it does not establish that this information is *causally used* by the model. In this section, we further investigate latent planning in rhyming couplet generation by employing intervention methods to test for causal influence directly. Given a prompt  $\mathbf{x}^{(c)}$  whose first line naturally leads to a *clean* rhyme family  $\mathcal{R}^{(c)}$ , we intervene on the model’s activations at various positions and investigate whether the generation of the second line shifts toward a different *corrupted* rhyme family  $\mathcal{R}^{(r)}$ . A successful intervention on a specific layer  $\ell$  and position  $i$  provides causal evidence that information at  $\mathbf{h}_{\ell,i}$  is causally involved in the model’s latent planning in rhyme selection downstream.



(a) Activation Patching: Hidden states from a corrupted forward pass with the rhyming word *fear* replace hidden states in a clean forward pass with the rhyming word *fright* at the newline token position. (b) Steering Vectors. Mean difference vectors from many *-ight* and *-ear* forward passes are added to their respective layers in a forward pass with the rhyming word *fright* at the newline token position.

Figure 4: Causal interventions attempting to shift the couplet rhyme scheme from *-ight* to *-ear*.

#### 4.1 Activation Patching

Let  $\mathbf{x}^{(c)}$  denote a *clean* prompt and  $\mathbf{x}^{(r)}$  a *corrupted* prompt that differs from  $\mathbf{x}^{(c)}$  only in the last word of the first line, inducing a different target rhyme family  $\mathcal{R}^{(r)}$ . The last words are synonyms with different rhyme schemes (i.e., *fear* and *fright*), so the prompts are semantically similar but different in rhyme scheme. For layer  $\ell$  and position  $i$ , let  $\mathbf{h}_{\ell,i}^{(c)}$  and  $\mathbf{h}_{\ell,i}^{(r)}$  denote the hidden states under the clean and corrupted prompts, respectively. The patching intervention replaces  $\mathbf{h}_{\ell,i}^{(c)} \leftarrow \mathbf{h}_{\ell,i}^{(r)}$  at a single layer  $\ell$  and position  $i$ , while leaving all other activations unchanged (Figure 4a). Generation then proceeds from the patched state. If  $\mathbf{h}_{\ell,0}$  causally encodes rhyme planning information, this intervention should shift completions away from  $\mathcal{R}^{(c)}$  and toward  $\mathcal{R}^{(r)}$ .

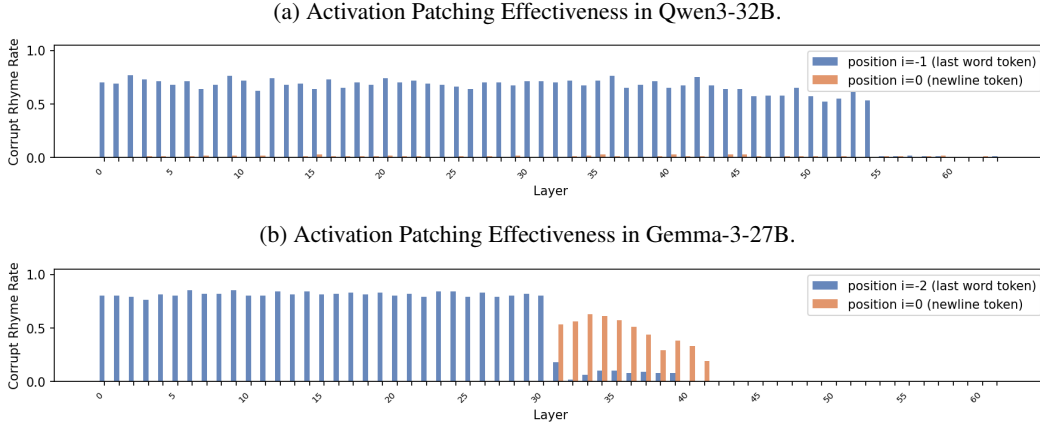


Figure 5: Corrupt rhyme rates on the clean run after patching corrupt activations at the last word and newline positions, averaged across 100 samples from 5 prompt pairs (20 per pair).

We sweep positions  $i \in \{-2, -1, 0, 1, 2, 3\}$ <sup>4</sup> across all layers and report the corrupt rhyme rate, the proportion of completions that rhyme with the corrupt last word of the first line of the couplet. We report averaged results over 5 prompt pairs (see Appendix D for the prompt pairs used) for a total of 100 samples per layer per position. Results for the two more informative positions, the last word token of the first line and the newline token, are reported in Figure 5, and full results across all six positions are shown in Figure 9 (Appendix D); notably, for Gemma the comma token at position  $i=-1$  is near zero across all layers, confirming it encodes no rhyme planning information.

<sup>4</sup>Due to tokenization differences, the last word token of the first line (e.g., *fright*) is at position  $-2$  for Gemma and  $-1$  for Qwen (see Appendix A)

For both models, patching at the last word token position is highly effective across nearly all layers, confirming that the corrupt rhyme word’s identity is causally encoded in the residual stream at that position throughout the forward pass. For Qwen3-32B, the corrupt rhyme rate at  $i = -1$  remains uniformly high ( $\sim 0.65$ - $0.77$ ) across all 64 layers with little effect at  $i = 0$ <sup>5</sup>, suggesting that rhyme information is stably maintained at the word token position and is not substantially aggregated at the newline token. For Gemma-3-27B, patching at  $i = -2$  is highly effective in early and middle layers (corrupt rhyme rate  $\sim 0.80$ , layers 0-29), but this effect drops sharply beyond layer 30. Concurrently, patching at the newline token ( $i = 0$ ) becomes effective only in these later layers, peaking around layers 33-35 (corrupt rhyme rate  $\sim 0.63$ ). This transition suggests that Gemma actively moves rhyme planning information from the word token to the newline token during the later portion of its forward pass, a form of information routing not observed in Qwen. Control baselines confirm these effects are specific to the corrupt word’s identity rather than generic activation disruption (Appendix E, Figure 10).

## 4.2 Steering Vectors

While activation patching tests causal influence on individual prompt pairs, it is sensitive to idiosyncratic differences between a specific clean and corrupted prompt. Steering vectors instead use many data points to estimate a direction in the hidden dimension space associated with a rhyme family, providing a more robust test of where rhyme planning representations are stored.

Let  $\mathcal{D}^{(c)}$  and  $\mathcal{D}^{(r)}$  be datasets of many clean and corrupted prompts, respectively. We compute the steering vector at layer  $\ell$  and position  $i$  as the mean vector difference

$$\mathbf{v}_{\ell,i}^{(c) \rightarrow (r)} = \text{mean}(\mathbf{h}_{\ell,i}^{(r)}) - \text{mean}(\mathbf{h}_{\ell,i}^{(c)}).$$

Intuitively, this vector captures a transition from the clean rhyme family  $\mathcal{R}^{(c)}$  to the corrupt rhyme family  $\mathcal{R}^{(r)}$ . For a clean prompt  $\mathbf{x}^{(c)}$ , the steering intervention adds  $\mathbf{h}_{\ell,i} \leftarrow \mathbf{h}_{\ell,i} + \alpha \mathbf{v}_{\ell,i}^{(c) \rightarrow (r)}$  at a single layer  $\ell$  and position  $i$  leaving all other activations unchanged (Figure 4b).

We investigate steering at the last word token and newline token positions across all layers, evaluating on the proportion of completions where the rhyme shifted to  $\mathcal{R}^{(r)}$ . We synthetically generate 120 examples per rhyme family—100 to compute the steering vectors and 20 for evaluation—and aggregate results across 10 clean/corrupt rhyme family pairs. Model completions are greedily sampled after intervention. Following [9], we use  $\alpha = 1.5$ <sup>6</sup>. We report results in Figure 6 with additional details and steered completion examples in Appendix G.

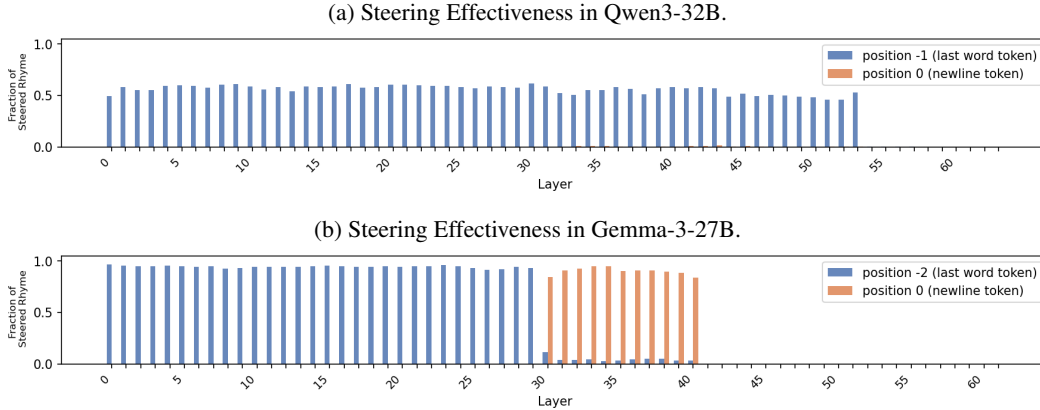


Figure 6: Effectiveness of adding steering vectors at the last word and newline position to shift the rhyme scheme toward the corrupted rhyme family.

Consistent with patching, the more informative test is steering at the newline token. In Qwen3-32B, newline steering produces little effect across all layers, consistent with patching showing minimal

<sup>5</sup>We observe extremely low but non-zero corrupt rhyme rate around layers 30-50, and further experiments patching over all layers show *some* causal evidence in Qwen3 patching at the newline (see Appendix F).

<sup>6</sup>We did not sweep  $\alpha$  due to compute limits. We acknowledge this value may not be optimal.

causal influence at  $i = 0$ : both interventions converge on the same conclusion that the newline token does not causally mediate rhyme planning in Qwen3. In Gemma-3-27B, the two positions are effective in complementary layer ranges, as last word steering dominates in early layers (0–30) but collapses after layer 30, while newline steering becomes effective precisely in middle layers (30–40). This is consistent with both the second probing peak in Figure 3 and the patching transition observed at the same depth. Taken together, the steering results corroborate the patching evidence for a representational handoff in Gemma where rhyme information is carried at the last word token in early layers, then transferred to and causally read from the newline token in middle layers. Again, this is absent in Qwen3.

### 4.3 Discussion

The causal results clarify the probing findings and reveal diversity in planning mechanisms between models. In Gemma-3-27B, both patching and steering confirm the newline token as a causally active planning site in middle layers (30–40), coinciding precisely with the second probing peak—suggesting these layers are where the planning representation is both constructed and read out. The first probing peak (layers 10–20) shows no corresponding causal effect, pointing to a possible earlier coarse encoding of rhyme context that is not directly used during generation. In Qwen3-32B, near-zero causal effect at position  $i = 0$  across both interventions suggests the newline probing signal is a representational byproduct of attending to the first line rather than active planning. Rhyme information instead remains causally localized to the last word token throughout all 64 layers.

Our full-layer sweeps also reveal that causal influence is not uniform across depth but concentrated in specific regions. This provides a more granular account than prior work, as [9] only reports steering at a single best layer and notes a small patching case study on Gemma-9B. Our results on larger models aggregated across many prompt pairs provide more systematic evidence. We also recover consistent evidence for the newline as a planning site, aligning with [1] but using far more lightweight methods on open-source models. However, we recognize that our approach cannot attribute planning to specific attention heads or MLP components, as interventions operate only over the residual stream.

## 5 Conclusion

Using lightweight mechanistic methods, we find that latent planning in language models is task-dependent, emerging selectively in structure-constrained generation but not in general text. Causal interventions reveal that Gemma-3-27B actively uses newline token representations for rhyme planning, while Qwen3-32B appears to rely primarily on the last word token. That two models of similar scale and capability organize the same planning computation around different token positions suggests that latent planning mechanisms are not universal but shaped by architectural choices or training. This suggests behavioral similarity between models need not imply mechanistic similarity.

Future work could further investigate other open-source models as well as larger models on the order of 100B parameters using these lightweight, scalable methods. The differences observed between Qwen and Gemma suggest that a broader model survey could reveal whether information routing to a dedicated planning site is a common strategy or a consequence of specific architectures or training regimes. Additionally, analysis beyond rhyming to other structure-constrained tasks, such as code generation, would help establish how broadly latent planning generalizes. On the causal side, our layer-wise sweeps identify where planning representations are read out, but cannot attribute this to specific model components such as individual attention heads or MLP layers. Applying more targeted interventions like path patching could decompose the planning circuit more precisely and clarify how information is routed between positions. Finally, as transcoder training becomes more feasible for larger open-source models, using lightweight methods first can help guide where in the model to investigate rather than running full circuit analyses across all layers and positions, substantially reducing compute and resource requirements for investigating latent planning.

### Team contributions

Nick: probing experiments, steering experiments, diagrams, writing.

Nicole: patching experiments, diagrams, writing, infinite compute provider.

Harsh: results analysis, related works, writing, poster.

## References

- [1] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. <https://transformer-circuits.pub/2025/attribution-graphs/biology.html>, 2025. Transformer Circuits Thread.
- [2] Michael Hanna and Emmanuel Ameisen. Latent planning emerges with scale. In *The Fourteenth International Conference on Learning Representations*, 2026.
- [3] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [4] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. <https://transformer-circuits.pub/2025/attribution-graphs/methods.html>, 2025. Transformer Circuits Thread.
- [5] 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. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [6] Anthropic. Claude 3.5 Haiku. <https://www.anthropic.com/news/claude-haiku-3-5>, October 2024. Large language model. Released October 2024.
- [7] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.
- [8] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Pettrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu,

Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pöder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025.

- [9] Jim Maar, Denis Paperno, Callum Stuart McDougall, and Neel Nanda. What’s the plan? metrics for implicit planning in llms and their application to rhyme generation and question answering, 2026.
- [10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022.
- [11] Jacob Pfau, William Merrill, and Samuel R. Bowman. Let’s think dot by dot: Hidden computation in transformer language models, 2024.
- [12] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024.
- [13] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47), November 2022.
- [14] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task, 2024.
- [15] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models, 2023.
- [16] Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart Russell. Evidence of learned look-ahead in a chess-playing neural network, 2024.
- [17] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2024.
- [18] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025.
- [19] John Hewitt and Percy Liang. Designing and interpreting probes with control tasks, 2019.
- [20] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

- [21] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small, 2022.
- [22] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization, 2023.
- [23] Andy Ardit, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024.
- [24] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.
- [25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [26] 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.
- [27] Anthropic. Claude Sonnet 4.6. <https://www.anthropic.com/news/claude-sonnet-4-6>, February 2026. Large language model. Model string: claude-sonnet-4-6. Accessed: 2026-03-05.
- [28] Richard L. Weide. The CMU Pronouncing Dictionary. <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>, 1993.

## A Model Configurations

We provide relevant details for the language models we probe and intervene on in Table 1.

Table 1: Model Architecture Details

|                                  | Qwen3-32B [7] | Gemma-3-27B [8] |
|----------------------------------|---------------|-----------------|
| vocabulary size, $ \mathcal{V} $ | 151,936       | 262,208         |
| hidden dimension, $d$            | 5,120         | 5,376           |
| number of layers, $L$            | 64            | 62              |

Notably, the models differ in tokenization. For example, Qwen3 treats , \n as a single token while Gemma-3 treats , and \n as separate tokens, hence the difference in positions of the last word token in rhyming couplets (position  $-1$  in Qwen, position  $-2$  in Gemma).

|           |     |        |        |      |     |       |      |     |
|-----------|-----|--------|--------|------|-----|-------|------|-----|
| Position: | ... | $-2$   | $-1$   | $0$  | $1$ | $2$   | $3$  | ... |
| Qwen3:    | ... | of     | fright | , \n | and | hoped | that | ... |
| Gemma-3:  | ... | fright | ,      | \n   | and | hoped | that | ... |

## B Additional Probing Results

We report additional top-1 accuracy results for probing on general text (Figure 7) and complete results for rhyming couplets (Figure 8).

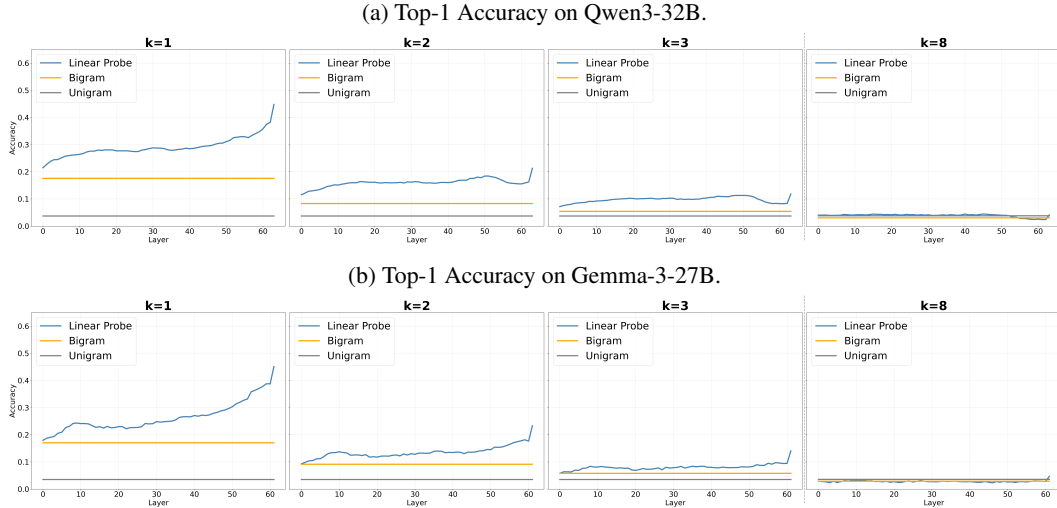


Figure 7: Accuracy of linear probes to predict  $k$  tokens ahead from hidden states on general text.

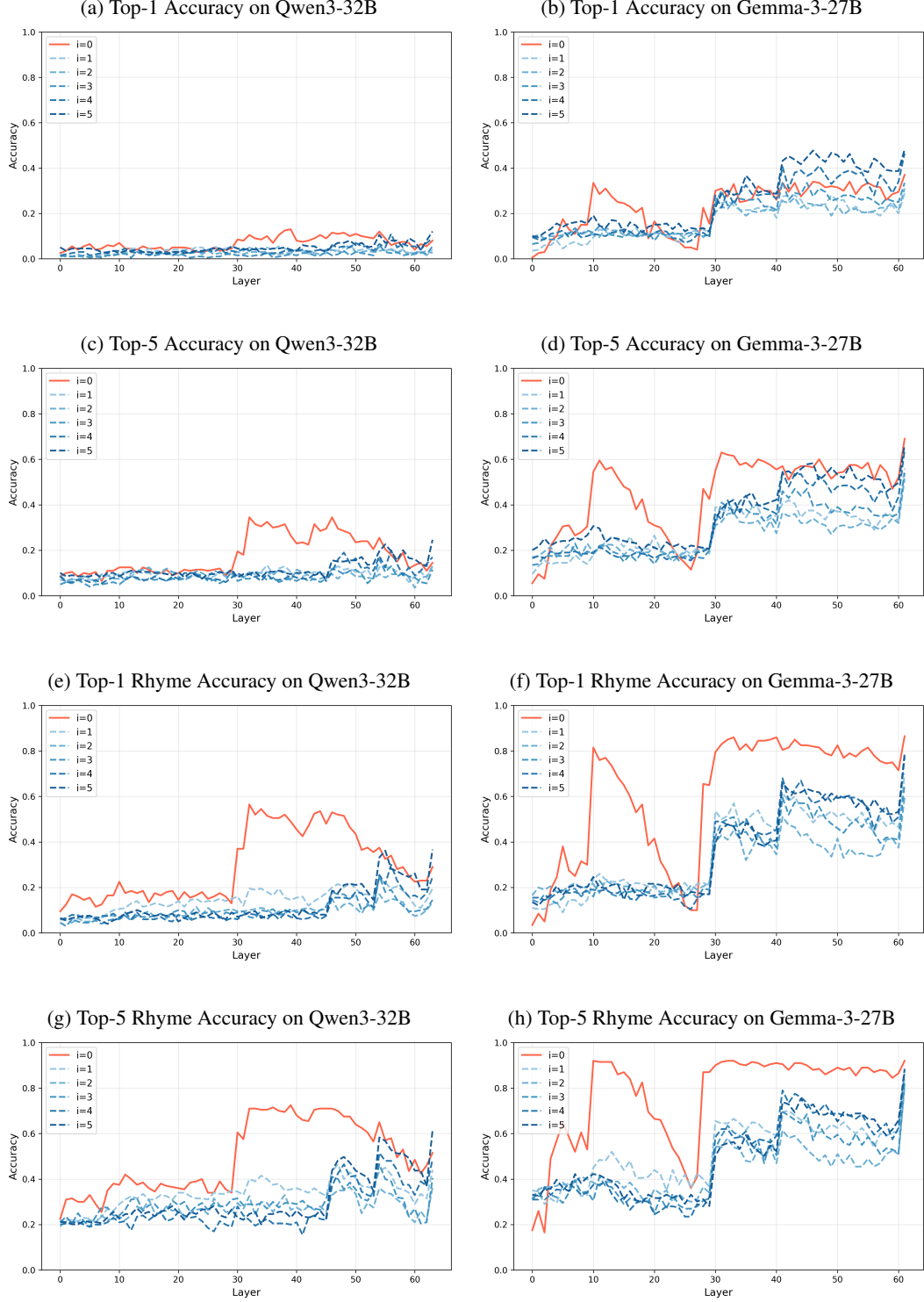


Figure 8: Accuracy of linear probes to predict the rhyming word from hidden states at position  $i = 0$  (newline token) compared to subsequent positions  $i > 0$  closer to the rhyming word.

## C Rhyming Capabilities Omitting Newline Tokens

To investigate the importance of the newline token on rhyming capabilities, we compare the baseline rhyming capabilities of the models’ greedy completions compared to prompts with the newline token omitted. For evaluation, we use the same set of 1,200 synthetically generated prompts from the poem probing experiments.

| Experiment      | Qwen3-32B | Gemma-3-27B |
|-----------------|-----------|-------------|
| With newline    | 67.25%    | 90.33%      |
| Without newline | 56.83%    | 61.08%      |

Table 2: Rhyming Capabilities with and without newline.

We find that, despite the instruction for rhyming still being present in the context, the models have much less success rhyming. This suggests the importance of the newline token in rhyme scheme planning.

We note an interesting observation specifically for completions sampled from Gemma-3-27B. In all greedy completions with the newline token, Gemma-3 follows the poem up by starting a haiku. For example:

Prompt: A rhyming couplet:\n She hoped to one day obtain,\n  
Completion: A life free from sorrow and pain.\n\n A haiku:\n Green leaves

Without the newline token, however, Gemma-3 always ends the completion with a special `<end_of_turn>` token, though it seems to still attempt the rhyme. For example,

Prompt: A rhyming couplet: She hoped to one day obtain,  
Completion: A life free from sorrow and pain.<end\_of\_turn>

This hints that Gemma-3 may have been specifically fine-tuned for poem generation, hence its overreliance on the newline token and its memorization of the training corpus, which likely consisted of many other poems, like haikus.

## D Additional Activation Patching Details

For each pair, we show the clean prompt, the corrupt prompt, and an example patched completion where activation patching was successful in steering the rhyme scheme.

### Pair 1: doom / dread, patched at $\ell = 10, i = -1$ :

Clean: A rhyming couplet:\n The empty house was filled with silent  
doom,\n when suddenly they  
Corrupt: A rhyming couplet:\n The empty house was filled with silent  
dread,\n when suddenly they  
Patched: A rhyming couplet:\n The empty house was filled with silent  
doom,\n when suddenly they heard a creaking bed.

### Pair 2: bliss / joy, patched at $\ell = 1, i = -2$ :

Clean: A rhyming couplet:\n The children laughed in bliss,\n until they  
all  
Corrupt: A rhyming couplet:\n The children laughed in joy,\n until they all  
Patched: A rhyming couplet:\n The children laughed in bliss,\n until they  
all became a toy.

### Pair 3: dark / night, patched at $\ell = 0, i = -2$ :

Clean: A rhyming couplet:\n She wandered home alone into the dark,\n and then she  
Corrupt: A rhyming couplet:\n She wandered home alone into the night,\n and then she  
Patched: A rhyming couplet:\n She wandered home alone into the dark,\n and then she saw a strange and eerie light.

**Pair 4: grief / pain, patched at  $\ell = 2, i = -1$ :**

Clean: A rhyming couplet:\n I never knew the depth of such grief,\n as though the  
Corrupt: A rhyming couplet:\n I never knew the depth of such pain,\n as though the  
Patched: A rhyming couplet:\n I never knew the depth of such grief,\n as though the sky had lost its rain.

**Pair 5: fright / fear, patched at  $\ell = 1, i = -1$ :**

Clean: A rhyming couplet:\n She felt a sudden sense of fright,\n and hoped that  
Corrupt: A rhyming couplet:\n She felt a sudden sense of fear,\n and hoped that  
Patched: A rhyming couplet:\n She felt a sudden sense of fright,\n and hoped that someone would appear.

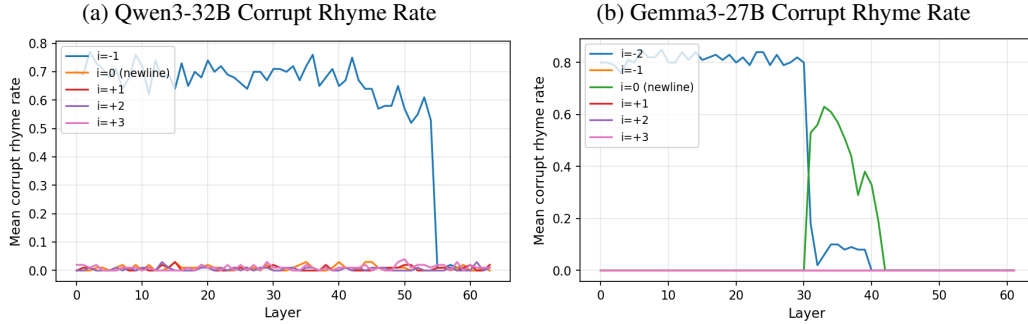


Figure 9: Mean corrupt rhyme rate across layers and token positions under activation patching, averaged over 5 prompt pairs ( $N=100$ ). Each curve shows the rate at which the clean run produces a word rhyming with the corrupt cue when activations at that position and layer are replaced.

## E Activation Patching Baselines

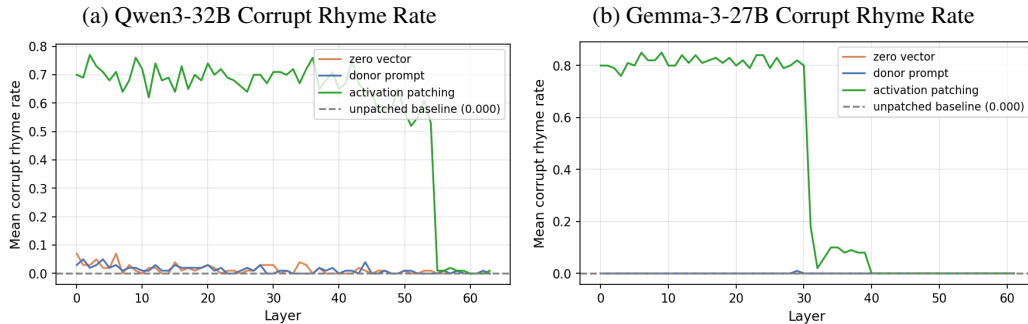


Figure 10: Mean corrupt rhyme rate across layers, compared against multiple baselines

To verify that the high corrupt rhyme rates observed under activation patching reflect the specific encoding of the corrupt rhyme word rather than a generic effect of perturbing the residual stream, we run two additional baselines. The *zero-vector* baseline replaces the hidden state at the patch position with an all-zeros vector, testing whether simply zeroing out that position is sufficient to induce corrupt rhyming. The *donor-prompt* baseline replaces it with a single hidden state vector cached from the same token position in a semantically unrelated sentence (“The weather outside is warm and sunny today, and the birds are singing.”), testing whether any arbitrary non-zero activation can produce the effect. Figure 10 shows results for both models at their respective key positions ( $i=-2$  for Gemma-3-27B,  $i=-1$  for Qwen3-32B), averaged across all five prompt pairs. In both cases, the two control conditions produce corrupt rhyme rates near zero across all layers, while true activation patching reaches  $\sim 0.80$  for Gemma and  $\sim 0.70$  for Qwen3. This confirms that the effect is causally specific to the identity of the corrupt word encoded in the residual stream at that position, and not an artifact of activation disruption per se.

## F Activation Patching All Layers

Since activation patching on a single layer can be a small intervention with negligible effect, we also experiment with patching all layers at once at a single position. We report results below in Table 3.

| Patch Position | Corrupted Rhyme % |             |
|----------------|-------------------|-------------|
|                | Qwen3-32B         | Gemma-3-27B |
| None           | 0                 | 0           |
| Last Word      | 96                | 100         |
| 0              | 9                 | 73          |
| 1              | 0                 | 0           |
| 2              | 0                 | 0           |
| 3              | 0                 | 0           |

Table 3: Effect of activation patching *all layers* at different positions on the proportion of generated rhymes matching the corrupted rhyme family.

We observe that activation patching on the newline token (position 0) has some observable effect in Qwen3, and the already-observed large effect in Gemma. Notably, patching at positions  $> 0$  have no effect, meaning we find little evidence of planning representations being stored in the first few tokens of the second line.

## G Additional Steering Vector Details

We first chose 10 simple rhyme families: *-at*, *-it*, *-ay*, *-ight*, *-ing*, *-ell*, *-all*, *-ake*, *-ock*, *-own*. Then, for each family we prompt Claude Sonnet 4.6 to generate 120 half-couplets using that rhyme family. If there are  $n < 120$  English words in a certain rhyme family, then we prompt to ensure no more than  $\lceil n/120 \rceil$  appear in the corpus to ensure diversity. We run code (with the CMU Pronouncing dictionary API [28]) to verify this diversity constraint. Data is then randomly shuffled and separated into the steering vector computation set and evaluation set.

We run forward passes, storing activations and computing steering vectors for all 90 possible rhyme family steers. We only run experiments on 10 of these pairs due to compute constraints. Specifically, we only attempt to steer from  $\{-at, -it\}$  into  $\{-ell, -all, -ake, -ock, -own\}$ .

We provide a successful steered example. The following generation was from adding a computed steering vector for  $-at \rightarrow -ell$  at the newline token position ( $i = 0$ ) at layer 33 in Gemma-3-27B.

Prompt: A rhyming couplet:\nShe shook her head at the spoiled little brat,\n\nUnsteered: Who threw his toys and sat right where he sat.\n\nSteered: And wished she'd sent him straight to hell.