InverseScope: Scalable Activation Inversion for Interpreting Large Language Models

Anonymous ACL submission

Abstract

Understanding the internal representations of large language models (LLMs) is a central challenge in interpretability research. Existing feature interpretability methods often rely on strong structural assumptions—such as linearity or sparsity—that may not hold in practice. In this work, we introduce InverseScope, an assumption-light and scalable framework for interpreting neural activations via input inversion. Given a target activation, InverseScope characterizes its encoded information by generating natural-language inputs that produce nearby activations, grounding abstract internal states in concrete language. To overcome the prohibitive cost of sampling in high-dimensional activation spaces, we propose a novel control-layer conditioning architecture that substantially improves sample efficiency compared to prior token-prepending approaches. We demonstrate that InverseScope can be used to study various problems, including sentence-level linear analogies, encoded information in activation neighborhoods, and local representation directions. The framework scales to state-of-the-art open-source models of up to 14B parameters.

1 Introduction

Recent advances in mechanistic interpretability aim to reverse-engineer neural networks' computations into human-understandable processes (Bereska and Gavves, 2024; Sharkey et al., 2025). A central task in this field is feature interpretability, which seeks to understand what information is encoded in a network's activations and how it is represented. This understanding is crucial for analyzing how information propagates and is processed across layers. Numerous methods have been proposed for feature interpretability, including linear probing (Alain and Bengio, 2016; Park et al., 2023), sparse auto-encoders (Cunningham et al., 2023; Gao et al., 2024), and other approaches (Bau et al., 2017; Xu et al., 2024).

Despite their successes, these methods share a fundamental limitation: they rely on strong assumptions about the structure of neural representations. Specifically, linear probing assumes a linear relationship between activations and specific concepts in the inputs, and sparse auto-encoders presuppose that activations can be decomposed into a sparse sum of linear directions. The validity of these assumptions remains an open and actively debated question, particularly in the context of LLMs (Levy and Geva, 2024; Engels et al.; Smith, 2024). Designing experiments that rigorously test these hypotheses is itself challenging, making it difficult to assess the reliability of interpretations derived from such approaches.

To address these limitations, a promising alternative is to interpret activations by inverting them back to the input space, where representations are naturally grounded in human-understandable language. This inversion-based paradigm has a long history in computer vision (Erhan et al., 2009; Mahendran and Vedaldi, 2014) and has recently gained traction in the NLP community. For instance, Rep2Text (Zhao et al., 2024) demonstrates that the representation at a single last-token position encodes sufficient information to autoregressively reconstruct the full input text via a trained adapter, while InversionView (Huang et al., 2024) shows that the information encoded in a hidden state can be characterized by collecting a distribution of inputs that yield similar activations. Together, these works highlight the potential of inversion-based methods to provide an assumption-light and intuitive framework for feature interpretability, enabling a direct mapping between abstract internal states and concrete natural language.

However, existing inversion methods each face their own limitations. Rep2Text (Zhao et al., 2024) targets exact reconstruction of the original input and consequently lacks the neighborhood perspective central to interpretability. InversionView

(Huang et al., 2024) frames inversion as neighborhood characterization, but its applicability has largely been demonstrated in task-specific templates and smaller-scale settings. Architecturally, both methods condition by projecting the target activation into embedding space and injecting it as a soft token in the autoregressive stack. Our ablation experiments show that this input-level conditioning is suboptimal: injecting conditioning through dedicated intermediate control layers yields substantially better retrieval quality and sample efficiency (Section 4.2).

Our main contributions are as follows:

- **Control-layer conditioning architecture.** We introduce a conditional generation architecture in which control layers inject the target activation z into the backbone at every Transformer layer via cross-attention. This design substantially outperforms the token-prepending approach used in prior work, enabling reliable inversion at scale.

- **Scaling to general distributions and large models.** We scale inversion-based interpretability to general-domain text (Wikipedia (Wikimedia Foundation) and Project Gutenberg (Project Gutenberg Literary Archive Foundation)) and to LLMs of up to 14B parameters, removing the dependence on task-specific templates that has restricted prior methods. We further investigate how inversion quality varies with target model size and scope size.

- **Empirical analysis of representation geometry.** We demonstrate that InverseScope can be used to analyze representation geometry. We use it to recover sentence-level linear analogy structure in LLM activations, to study information loss in activations, and to discover feature directions in activation space.

Concretely, our evaluation is organized into five parts: sentence-to-vector case studies, retrieval-efficiency comparison, scaling-law analysis, most-different-neighbor search, and backpropagation-based concept-direction discovery. Together, these results position InverseScope as a practical inversion framework for geometric analysis of LLM activations.

2 Related works

Activation interpretation methods. A variety of methods have been developed to interpret neural network features. Classical approaches such as linear probing train simple classifiers on activations to identify linearly encoded features (Alain and Bengio, 2016; Park et al., 2023). More recent work includes sparse dictionary learning, which decomposes activations into sparse and interpretable components to disentangle feature representations (Cunningham et al., 2023; Gao et al., 2024). While effective, these approaches typically rely on strong structural assumptions about the underlying representations, for example that features are linearly separable, sparse, or decomposable into a fixed dictionary. Such assumptions may not hold in general, potentially limiting their ability to capture more complex or distributed representations in LLMs.

Natural language interpretability. Several recent works have explored assigning human-interpretable labels—such as natural language descriptions—to the internal activations of LLMs. Early approaches such as the logit lens interpret intermediate activations by projecting them directly into the vocabulary space and reading out the most likely tokens, effectively providing single-word descriptions of representations (Geva et al., 2022). Building on this idea, training-free methods like SelfIE (Chen et al., 2024) and PatchScope (Ghandeharioun et al., 2024) use a pretrained LLM to read out the information encoded in residual stream activations. Complementing these approaches, LatentQA (Pan et al., 2024) trains a decoder model to answer natural language questions about activations. More recent work further generalizes this paradigm by treating activations as a new input modality: Activation Oracles are trained to answer arbitrary queries about activations, enabling more flexible and general-purpose natural language interpretation (Karvonen et al., 2025a). Despite their flexibility, these methods face a fundamental challenge: the natural language explanations they produce are difficult to verify. In particular, they may suffer from hallucination, where the generated descriptions are fluent and plausible but do not faithfully reflect the underlying representations, making it challenging to assess their reliability.

Inversion-based interpretability. A complementary line of research focuses on interpreting model activations by identifying the inputs that give

rise to them, a strategy with a long history in computer vision via activation maximization and representation inversion (Erhan et al., 2009; Nguyen et al., 2016; Mahendran and Vedaldi, 2014). Recent work has extended this paradigm to language models along two distinct directions. Rep2Text (Zhao et al., 2024) trains a learned adapter to autoregressively decode the full input text from the single last-token representation of an LLM, demonstrating that substantial lexical and semantic information is recoverable from this compressed summary vector. InversionView (Huang et al., 2024) takes a different approach: rather than reconstructing the exact original input, it samples a distribution of inputs whose activations are close to the target, thereby characterizing what information is encoded in a given hidden state. InverseScope differs from existing work in both interpretive goal and architecture. Rep2Text targets exact reconstruction of the original input, whereas InversionView characterizes activation neighborhoods but is confined to task-specific settings. Architecturally, InversionView, Rep2Text, and Activation Oracles all follow the same practice: the target activation is projected into the token embedding space and placed as a soft token alongside the model's input embeddings. InverseScope instead injects the conditioning signal at every Transformer layer via dedicated control layers. In application scope, InverseScope supports general-domain training data and scales to LLMs of up to 14B parameters.

Conditional and controllable generation. Conditional generation, which generates outputs based on a constrained intermediate state, is a foundational technique in deep learning. Initially established with Conditional VAEs (Sohn et al., 2015) and Conditional GANs (Mirza and Osindero, 2014), this paradigm has evolved into methods where the conditioning signal specifies desired structure in the output. Notable examples include modern text-to-image synthesis systems (Ramesh et al., 2022) and structured guidance methods such as ControlNet (Zhang et al., 2023), which condition large generative models on auxiliary inputs. Similarly, controllable text generation (Keskar et al., 2019) steers language model outputs using explicit attributes. These developments highlight the generality of treating intermediate representations as controllable interfaces for generation.

3 InverseScope

This section presents InverseScope in three parts. We first formalize the geometric intuition behind inversion-based interpretability and explain why local activation neighborhoods can reveal encoded features. We then describe the Control-Layer architecture that enables efficient conditional generation from target activations. Finally, we detail the training objective and data construction pipeline used to scale inversion from controlled templates to training on general-domain data and large target models.

3.1 Intuition

Inversion-based interpretability is grounded in a simple geometric intuition: nearby activations tend to preserve similar semantic content (Bengio et al., 2013). Throughout the paper, we use *activation* to denote the hidden representation at a specific layer and token position, $z \in \mathbb{R}^d$. If two inputs induce nearly identical activations, then their differences are unlikely to be strongly encoded in that representation. Conversely, if a local activation neighborhood consistently maps to inputs sharing a linguistic pattern, that pattern is likely encoded in that region of latent space. This intuition is compatible with the continuity of neural networks, where nearby states induce similar downstream computations.

Building on this intuition, our method investigates the information encoded in a target activation z by inverting the activation geometry to identify inputs whose activations are close to z. The resulting neighborhood in input space supports explicit, testable hypotheses about which features are preserved (or discarded) by the representation.

For example, let (x, z) be an input–activation pair where x is "Language models are unsupervised multitask learners." and z is extracted from the layer-16 residual stream at the final period token. InverseScope generates nearby prompts such as "Transformer models are lifelong unsupervised learners.", "Transformer models are self-supervised learning machines.", and "Neural networks are unsupervised multi-task learners." In contrast, an in-domain but semantically different sentence such as "Attention is all you need." remains farther from z. This neighborhood suggests that z encodes information about model-family semantics and unsupervised-learning attributes, which can then be tested quantitatively.

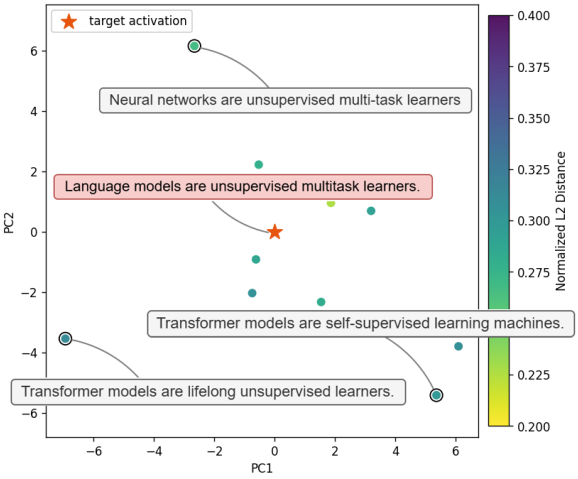

Figure 1: An example neighborhood generated by InverseScope. The target input is "Language models are unsupervised multitask learners.", and the target activation is extracted at the final inference position.

The key challenge in implementing this approach lies in efficiently obtaining inputs whose activations land near the target z. In the high-dimensional activation spaces of modern LLMs, where dimensions can number in the thousands, the probability that a randomly sampled input produces an activation close to z decays exponentially with dimensionality. This makes naive rejection sampling prohibitively inefficient. To address this, we train a conditional generator as a learned prior, which directly produces candidate inputs likely to yield activations near z, making the sampling process tractable.

3.2 Architecture

In this section we describe the network architecture of **InverseScope**. Our objective is to efficiently sample from the conditional distribution $P(x; z)$ defined in the previous section. To achieve this, InverseScope is designed as a conditional generator trained to approximate this target distribution.

We adopt the decoder-only Transformer paradigm, which has proven highly effective for modeling natural language distributions via next-token prediction objectives (Brown et al., 2020). InverseScope extends this standard language modeling framework by conditioning the token prediction not only on the preceding sequence context but critically, also on the target activation z. This conditioning is essential to align the generated sequence with the semantic information encoded in z, allowing the generator to approximate the target distribution $P(x; z)$.

As illustrated in Figure 2, our conditional generator extends a decoder-only Transformer with *Control Layers* that inject the conditioning activation z into each layer's hidden state. Each Control Layer is a cross-attention layer from token states to the conditioning vector. Because conditioning is performed with a single vector, we introduce a learned sink key for each control head to avoid degenerate one-key behavior and improve optimization stability. These sink keys are initialized to zero and updated during training. A mathematical formulation of Control Layers is provided in Section A.2.

This design differs from prior inversion-style architectures, where the target activation is first projected to embedding space and then injected as a soft token (Huang et al., 2024; Karvonen et al., 2025a; Zhao et al., 2024). In that family, extraction of condition information and ordinary token–token interaction are both delegated to the same self-attention stack, so the model must multiplex these competing roles within shared attention heads. Moreover, because the condition is represented as just another token, its signal is repeatedly transformed by subsequent layers and can become progressively diluted at higher depths, reducing how reliably downstream layers can recover and use the original condition information. InverseScope instead applies direct per-layer conditioning, yielding a persistent control signal throughout generation. We show in Section 4.2 that this leads to substantially better retrieval quality and sample efficiency.

3.3 Training

Loss function. We maximize the conditional log-likelihood of each training sequence given its paired activation. The objective reduces to standard next-token prediction with position-dependent weights:

$$\mathcal{L}(\theta; x_i, z_i) = \sum_{t=1}^{n_i-1} w_{(n_i-t)} \log P_\theta(x_{i,t+1}; x_{i,1:t}, z_i),$$

where θ denotes all trainable parameters. We set $w_i = 1$ for $1 \leqslant i \leqslant 32$, $w_i = 0.5$ for $33 \leqslant i \leqslant 64$, $w_i = 0.25$ for $65 \leqslant i \leqslant 128$, and $w_i = 0$ otherwise. This reweighting emphasizes tokens that are temporally closer to the extraction site, which typically carry stronger supervision for the target activation. The design is motivated by the same locality prior behind sliding-window attention: for

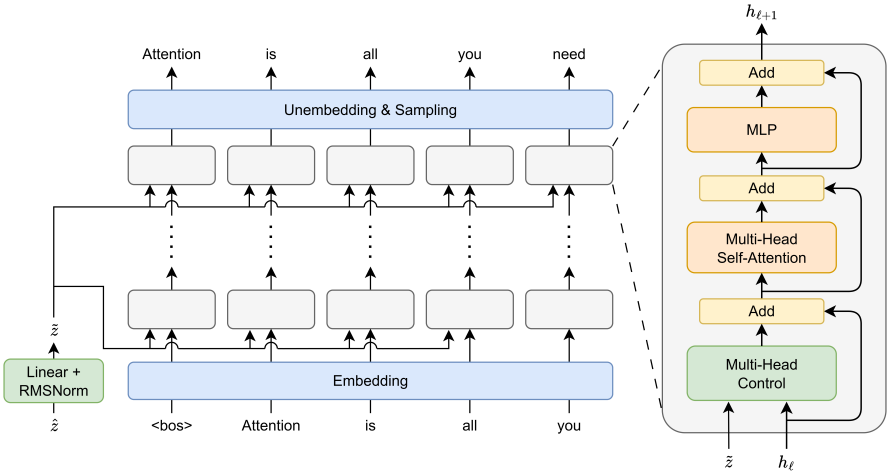

Figure 2: Network architecture of InverseScope. The decoder-only Transformer backbone is shown with its parameters colored in orange and blue. The additional layers introduced for conditioning are colored in green.

a representation at a given position, nearby context contributes most directly, while the contribution of distant tokens decays rapidly after multiple transformations. Our piecewise schedule is a simple approximation of this distance-decay effect.

Long-context regime. This objective enables training on activations extracted from arbitrary positions in natural long-form text. Because tokens beyond a 128-token window receive zero weight, InverseScope is not optimized to reconstruct very distant context, and at inference time for long prompts we evaluate reconstruction over the most recent 128 tokens around the extraction site. Nevertheless, this setting provides a key practical advantage: to the best of our knowledge, InverseScope is the first inversion method that can directly condition on activations extracted from long-context inference positions.

Dataset. We train on 100M input–activation pairs (x, z) from general-domain text and arbitrary positions for activation extraction, with sequence lengths from 4 to 256 tokens, sourced from Wikipedia and Project Gutenberg (Wikimedia Foundation; Project Gutenberg Literary Archive Foundation). Full dataset construction details are provided in Appendix A.

Initialization and optimization. We initialize InverseScope from pretrained backbones. For newly introduced control layers, we initialize the output projection to zero and copy query/key/value projections from the corresponding self-attention layer. This zero-output initialization yields a stable cold start: at step 0, the model matches the unconditional pretrained LM, and conditioning strength increases smoothly during training.

During training, we optimize all parameters using AdamW (Loshchilov and Hutter, 2019) with a warmup–stable–decay learning-rate schedule. Full hyperparameter settings are provided in Appendix A.3.

4 Experiments

In this section, we evaluate InverseScope across several dimensions. We first show, through case studies, that InverseScope successfully recovers feature geometry in LLM activations. We then demonstrate that our architectural design substantially improves sample efficiency over previous baselines. Third, we examine how performance scales with both target-LLM size and InverseScope-backbone size. Finally, we present two additional uses of InverseScope: studying information loss by searching for the most different neighbors, and obtaining input-specific feature directions via back-propagation.

Unless otherwise stated, all experiments use the `post_resid` activation at layer 16 of Qwen3-4B (Qwen Team, 2025) as the inversion target, with a 5B InverseScope model initialized from Qwen3-4B-Base.

4.1 Case studies: sentence-to-vector

A well-known property of embedding models is the word-to-vector property: letting $v(\cdot)$ denote word embeddings, we have $v(\text{woman}) - v(\text{man}) +$

L2 Dist.	Prompts
0.1397	The queen stood on the dais.*
0.1475	The princess stood on the dais.
0.1568	The king stood on the dais.
0.1666	The queen stood upon the dais.
0.1793	The tall queen stood on the dais.
0.1887	The woman stood on the dais.
0.2002	The princess stood at the dais.
0.2065	The young queen stood on the dais.

Table 1: Prompts generated by InverseScope conditioned on the target activation $z(\text{woman}) - z(\text{man}) + z(\text{king})$, ranked by L2 distance to the target. The expected sentence (marked with *) ranks at the top.

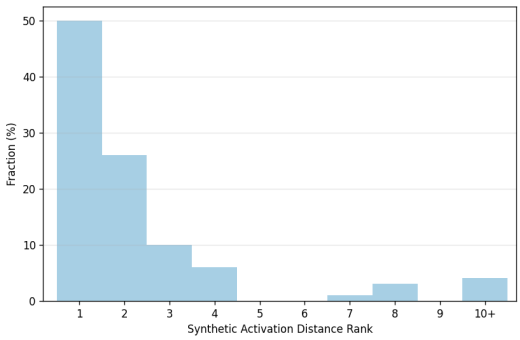

Figure 3: Histogram of rank positions of synthetic activations relative to target activations across sentence-level analogy tests. In 50% of cases, the synthetic activation is ranked first (closest).

$v(\text{king}) \approx v(\text{queen})$, where "$\approx$" means that $v(\text{queen})$ is the closest embedding to the target vector among all words in the vocabulary.

We ask whether an analogous property holds at the sentence level for LLM activations. For example, consider the four sentences: "The {woman/man/king/queen} stood on the dais." Treating the activation $z(x)$ at the last token position of a given layer as a sentence embedding, we investigate whether the synthetic activation $z(\text{"The woman ..."}) - z(\text{"The man ..."}) + z(\text{"The king ..."})$ is the closest to $z(\text{"The queen ..."})$ among all possible sentences. Unlike the word vocabulary, which has a fixed, enumerable size, the space of sentences grows exponentially with length, rendering exhaustive search for the closest sentence embedding computationally intractable.

InverseScope addresses this challenge by conditioning the generator on the target vector and sampling candidate sentences from the resulting conditional distribution $P(x; z)$. Table 1 lists the top prompts generated by InverseScope. Indeed, "The queen stood at the dais." achieves the smallest L2 distance to the target, confirming that LLM activations preserve this linear analogy structure at the sentence level.

Beyond the top-1 match, the neighborhood in Table 1 provides a finer-grained view of what the target activation encodes. Two patterns are clear. First, the predicate template "stood on the dais" is preserved almost perfectly across neighbors (with minor surface variants such as "upon/at the dais"), indicating that this event structure is strongly encoded in the local representation. Second, variation is concentrated in the subject token (e.g., "queen," "princess," "woman," and occasionally "king"), suggesting that the activation also captures a structured semantic axis over role/gender attributes rather than a single memorized lexical item.

We run the same protocol on 100 additional analogy pairs and sentence templates (Appendix C.1), and report the rank of the target activation among sampled neighbors. As shown in Figure 3, the synthetic activation is ranked first in 50% of the evaluated cases. Additional successes and failure cases are included in Appendix C.1.

The sentence-level analogy effect extends beyond lexical substitution in fixed templates. We also observe analogous behavior for higher-level transformations such as tense changes and active/passive voice. Additional examples are provided in Appendix C.1.

4.2 Retrieval Efficiency

A core requirement for inversion-based interpretability is that the generator reliably produces inputs whose activations lie close to the target z. We evaluate this via the *normalized L2 distance* between each generated sample's activation and the conditioning target (Appendix B). We compare three strategies: (1) **InverseScope**, which samples from the learned conditional distribution $P(x; z)$; (2) **Baseline**, a soft-token conditioning architecture that prepends a projected activation token, following prior inversion methods; and (3) **Random**, exhaustive nearest-neighbor retrieval from a 10M-sentence corpus.

Figure 4 shows the distribution of retrieval distances for all three methods. Random corpus retrieval performs poorly: it almost never finds samples with normalized L2 distance below 0.6. The soft-token baseline improves over random retrieval, but still almost never reaches very close neighbors

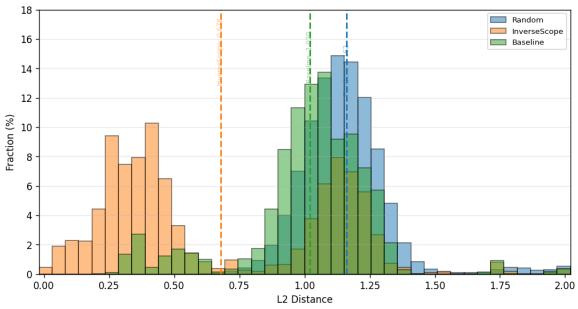

Figure 4: Distribution of L2 distances between sampled inputs and the conditioning activation z, for InverseScope (orange), the soft-token baseline (green), and random corpus retrieval (blue). Dashed lines indicate the mean of each distribution.

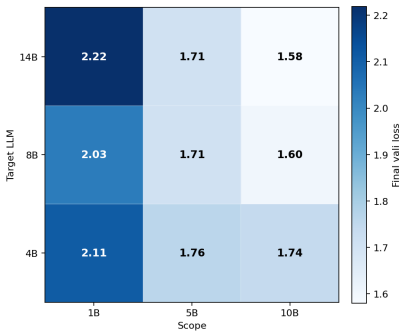

Figure 5: Scaling behavior of InverseScope across target-model size and scope-model size, measured by validation loss.

below 0.25.

InverseScope substantially outperforms the soft-token baseline across both mean quality and tail efficiency. Its mean retrieval distance is **0.678**, compared to 1.02 for the baseline. More importantly, InverseScope samples ultra-close neighbors around 0.25 with probability greater than 10%, while the baseline is nearly unable to do so. This indicates a clear efficiency breakthrough: per-layer conditioning not only improves average retrieval quality, but also dramatically increases the chance of obtaining high-fidelity neighbors.

4.3 Scaling law and different sites

This section studies how inversion quality changes with model scale and with the activation site selected from the target LLM. For scaling, we collect activations from Qwen3-4B, 8B, and 14B, and train 1B/5B/10B InverseScope variants initialized from Qwen3-0.6B/4B/8B-Base. Parameter counts are reported in Table 3 in Appendix A.2. These scaling experiments use residual-stream activations from the middle layer (50% depth) of each target model.

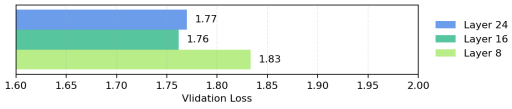

Figure 6: Validation loss when reconstructing from residual-stream activations at different target-model layers.

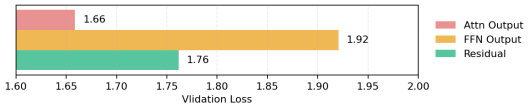

Figure 7: Validation loss when reconstructing from different module outputs at layer 16.

Figure 5 reports validation loss for all target/scope size combinations. Overall, lower losses concentrate toward the upper-right region (larger target models and larger scope models), although the trend is not strictly monotonic for every adjacent pair. This pattern is consistent with two effects: larger target models can encode richer information in residual activations, and larger scope models have greater capacity to decode that information during inversion.

For the 4B target model, however, increasing scope size from 5B to 10B yields only modest gains relative to larger-target settings. This suggests a diminishing-returns regime in which a scope model at roughly target scale is already sufficient to recover most decodable information from the activation.

We next analyze activation-site effects. For these experiments, we choose Qwen3-4B as the target model and train 5B InverseScope models. In Figure 6, we compare reconstruction quality across residual-stream layers of the target model. Deeper layers (16 and 24, i.e., 50% and 75% depth in Qwen3-4B) yield similar losses, while layer 8 (25% depth) is substantially worse. This suggests that at early layers, sentence/paragraph-level information is not yet fully aggregated into the residual stream, whereas by around the middle layers the representation is already sufficiently informative for inversion.

Figure 7 compares different module outputs at the same layer (attention output, FFN output, and residual stream). We observe clear separations: attention outputs give the lowest validation loss, while FFN outputs are the hardest to invert. A plausible explanation is that attention outputs pre-

L2 Dist.	Prompt
Original	In 1998 the western link to Munich Airport was opened as a two-track line which branches off north of Neufahr
0.1553	In 1998 the western link to Munich Airport was completed with the extension of the Autobahn A 99 to the south of Neufahr

Table 2: A high-edit pair within one activation neighborhood. Despite the near-neighbor relation in activation space ($d = 0.1553$), the token edit distance remains large ($= 16$). The target sentence is unfinished, so the corresponding prediction is also unfinished.

serve contextual interactions in a form that is more directly decodable, whereas FFN transformations discard or compress input-specific variation that is less relevant to next-token prediction.

Overall, these site-dependent inversion losses provide an empirical proxy for how much input information remains recoverable at different parts of the network. This complements the scaling analysis by showing not only *how much* information is decodable, but also *where* in the model that information is most accessible.

4.4 Probing Information Loss via Most-Different Neighbors

As discussed in the intuition section (Section 3.1), when inputs with distinct features are compressed into the same activation neighborhood, the differing features are likely not encoded in the activation. Here, we directly probe this phenomenon by searching for the most different inputs inside a small activation neighborhood.

Given a target activation from the validation set, we use InverseScope to sample neighboring activations and decode their corresponding prompts. We measure textual difference with token edit distance, and search for the most different prompt under the activation-distance constraint $d(z, \tilde{z}) < 0.2$.

Table 2 provides a representative outlier pair. Although the second prompt is still close in activation space ($d = 0.1553$), it changes multiple semantic factors simultaneously (event type, infrastructure entity, and direction). This suggests that those factors are weakly represented—or partially discarded—in this local region of the activation manifold, while shared backbone content remains dominant. The full neighborhood ranking for this case is provided in Table 8 in Appendix C.2.

4.5 Local Concept Directions via Backpropagation

Many interpretability interventions (e.g., linear steering) rely on the assumption that concept information can be manipulated along approximately linear directions in activation space. We use InverseScope to extract such *local* directions in a case-dependent manner, without requiring a globally linear concept representation.

Given a target activation z and a text-level objective, we define a differentiable loss on InverseScope's reconstruction and back-propagate that loss to the conditioning activation. For example, for the prompt "The capital of France is Paris.", we can maximize the log-probability of the token "Paris" and compute

$$g_x = \nabla_z \mathcal{L}_{\text{concept}}(x, z).$$

The gradient g provides a local direction that, according to InverseScope, most strongly increases the target concept under the chosen objective.

Compared with probing methods, this approach does not require carefully constructed paired datasets to fit a separate linear classifier (Alain and Bengio, 2016). Compared with SAE-based global features (Cunningham et al., 2023; Gao et al., 2024), the extracted direction is local and input-dependent, which is better suited to concepts that are globally nonlinear but locally steerable.

In practice, we do not apply raw gradients directly. Instead, for each concept we use the normalized gradient direction scaled by the RMS magnitude of the current activation, i.e.,

$$\tilde{g}_c(z) = \mathrm{RMS}(z) \cdot \frac{g_c}{\|g_c\|_2},$$

and perform edits of the form

$$z' = z - \alpha \tilde{g}_{\text{France}} + \beta \tilde{g}_{\text{Germany}}.$$

We evaluate this intervention on samples from the RAVEL dataset with $\alpha = \beta = 0.5$.

Results. On RAVEL, this direction-editing procedure achieves a causal score of 0.82. This result is higher than the reported SAE causal score on the same benchmark (Karvonen et al., 2025b), indicating that local concept directions extracted through InverseScope are a competitive and promising alternative for causal representation editing.

5 Limitations

While our results demonstrate the effectiveness of InverseScope, several limitations remain. First, long-context handling is only partially solved. InverseScope can condition on activations extracted from long-context inference positions, but reconstruction is still optimized and evaluated within a local token window around the extraction site rather than over the full original sequence. Extending inversion quality to full-sequence long-context reconstruction remains open. Second, several analyses in this paper are still case-study driven. We do not yet provide a fully standardized benchmark suite for inversion-based interpretability. Third, many experiments are qualitative and limited in scope and should be expanded across more tasks and model settings.

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *ArXiv*, abs/1610.01644.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Leonard Bereska and Efstratios Gavves. 2024. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. 2024. Selfie: Self-interpretation of large language model embeddings. *arXiv preprint arXiv:2403.10949*.

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

Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1.

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

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. *ArXiv*, abs/2401.06102.

Xinting Huang, Madhur Panwar, Navin Goyal, and Michael Hahn. 2024. Inversionview: A general-purpose method for reading information from neural activations. *arXiv preprint arXiv:2405.17653*.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, and Samuel Marks. 2025a. Activation oracles: Training and evaluating llms as general-purpose activation explainers. *ArXiv*, abs/2512.15674.

Adam Karvonen, Can Rager, Jiahai Wang, Ryan Greenblatt, Robert Huben, Adam Jermyn, Geoffrey French, Adly Templeton, Trenton Bricken, and Samuel Marks. 2025b. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 29223–29264.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Amit Arnold Levy and Mor Geva. 2024. Language models encode numbers using digit representations in base 10. *arXiv preprint arXiv:2410.11781*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Aravindh Mahendran and Andrea Vedaldi. 2014. Understanding deep image representations by inverting them. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784.

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

Anh Totti Nguyen, Jason Yosinski, and Jeff Clune. 2016. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *ArXiv*, abs/1602.03616.

Alexander Pan, Lijie Chen, and Jacob Steinhardt. 2024. Latentqa: Teaching llms to decode activations into natural language. *arXiv preprint arXiv:2412.08686*.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2023. The linear representation hypothesis and the geometry of large language models. *ArXiv*, abs/2311.03658.

Project Gutenberg Literary Archive Foundation. Project gutenberg. https://www.gutenberg.org/. Public-domain ebook archive; accessed May 9, 2026.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, and 1 others. 2025. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*.

Lewis Smith. 2024. The 'strong' feature hypothesis could be wrong. Accessed: 2025-05-15.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Neural Information Processing Systems*.

Wikimedia Foundation. Wikipedia, The Free Encyclopedia. https://www.wikipedia.org/. Text corpus from Wikimedia database dumps; accessed May 9, 2026.

Zhihao Xu, Ruixuan Huang, Changyu Chen, and Xiting Wang. 2024. Uncovering safety risks of large language models through concept activation vector. *Advances in Neural Information Processing Systems*, 37:116743–116782.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824.

Haiyan Zhao, Zirui He, Yiming Tang, Fan Yang, Ali Payani, Dianbo Liu, and Mengnan Du. 2024. Rep2text: Decoding full text from a single LLM token representation. *arXiv preprint arXiv:2511.06571*.

A Training details

A.1 Dataset

Unlike InversionView (Huang et al., 2024), which trains separate task-specific generators, and Rep2Text (Zhao et al., 2024), which targets exact input reconstruction on curated datasets, we train a single general-purpose generator on broad, task-agnostic text.

Source data. We extract full paragraphs from Wikipedia and Project Gutenberg, retaining 100M high-quality examples in total (50M from each source).

Activation extraction and pair construction. We use TransformerLens (Nanda and Bloom, 2022) to extract activations from Qwen3-4B (post-residual stream, layer 16). For each tokenized paragraph with token sequence length L, we sample one split index uniformly from

$$i \sim \mathrm{Uniform}\{4, \ldots, \min(L, 256)\}.$$

We then truncate the paragraph at index i: the prefix up to the split point is used as input text x, and the activation at the same position is used as conditioning target z, yielding one training pair (x, z) per paragraph. This split-index construction is equivalent to the random extraction-position sampling described in the main text. It leaves the activation-neighborhood evaluation protocol unchanged and only formalizes the procedure for constructing position-conditioned training pairs.

Training input format. During InverseScope training, we prepend `<|im_start|>` at the beginning of the text sequence and use `<|image_pag|>` as the boundary marker at the split point.

Activation Normalization. We apply a simple per-channel normalization to remove statistical biases: for each activation channel, we subtract its mean and divide by its standard deviation. This step is critical for Qwen3 models, which exhibit massive activations in certain channels that would otherwise destabilize InverseScope training.

Backbone Name	Backbone Size	Scope Size
`Qwen3-0.6B-Base`	0.75B*	1.00B
`Qwen3-4B-Base`	4.41B*	5.59B
`Qwen3-8B-Base`	8.19B	10.08B

Table 3: Parameter sizes of InverseScope variants. *Our backbone parameter count is larger than the original technical report because we untie the embedding and LM head for consistency across training settings.

A.2 Architecture and Model Size

The mathematical formulation of multi-head control layers is as follows:

$$q_{\ell,m}^{(i)} = Q_{\ell,m} h_\ell^{(i)}, \quad k_{\ell,m} = K_{\ell,m} z, \quad v_{\ell,m} = V_{\ell,m} z, \quad m \in \{1, \ldots, H\},$$

$$\omega_{\ell,m}^{(i)} = \frac{\exp(\langle q_{\ell,m}^{(i)}, k_{\ell,m} \rangle)}{\exp(\langle q_{\ell,m}^{(i)}, k_{\ell,m} \rangle) + \exp(\langle q_{\ell,m}^{(i)}, k_{\text{sink},m} \rangle)},$$

$$\text{Control}_\ell(h_\ell^{(i)}, z) = \sum_{m=1}^{H} \omega_{\ell,m}^{(i)} v_{\ell,m},$$

where i is the inference position index, m is the head index, and H is the number of control heads. The parameter matrices $Q_{\ell,m}, K_{\ell,m}, V_{\ell,m}$ are distinct from the backbone self-attention parameters. Each head has its own sink key $k_{\text{sink},m}$, and head outputs are summed to form the final control signal. This control signal is then added to the hidden state $h_\ell^{(i)}$ after the standard self-attention operation.

A.3 Hyperparameter Settings

We optimize InverseScope with AdamW. Peak learning rates depend on the generator backbone size: we use 10^{-4} for Qwen3-0.6B-Base, 5×10^{-5} for Qwen3-4B-Base, and 3.5×10^{-5} for Qwen3-8B-Base. The learning-rate schedule is warmup–stable–decay (WSD): a linear warmup over the first 10% of training steps to the peak rate; a stable phase at that peak; and a final phase spanning the last 20% of steps in which the learning rate decays linearly from the peak to 20% of the peak value.

Optimization details for reproducibility. All models are trained on the same 100M-example dataset with global batch size 4096 samples, corresponding to approximately 24,415 optimization steps. Micro-batch sizes are 32 (1B), 16 (5B), and 8 (10B), with gradient accumulation used to match the global batch size. We use AdamW with LitGPT default β coefficients and weight decay 0.1. With an average sequence length of about 30 tokens per example, the total token budget is approximately 3T tokens.

Decoding and sampling details for reproducibility. Unless otherwise noted, all generation-based evaluations use a fixed decoding policy shared across methods: top-k sampling with $k = 64$ and maximum generation length 128 tokens. For each inversion query, we sample 64 candidates.

Randomness and statistical reporting. Inference uses a fixed random seed (42). Per-target sampling counts are reported with each experiment; when aggregate statistics are reported, we include mean and spread over the corresponding sample set.

A.4 Compute Resources

Activation extraction takes 24/24/36 hours for the 1B/5B/10B settings on two 8×H800 nodes to produce activations for the 100M prompt–activation pairs. We use TransformerLens for extraction, which is not optimized for large-scale inference. We therefore expect extraction runtime to decrease substantially when using a high-throughput inference framework such as vLLM.

InverseScope training takes 20/54/95 hours for the 1B/5B/10B models on one 8×H800 node. All experiments use mixed-precision training with the LitGPT framework and data parallelism with gradient accumulation.

A.5 Artifact Licenses and Terms of Use

Our experiments use external artifacts (datasets and base models) that are distributed under their respective original licenses and terms of use. Specifically, training text is sourced from Wikipedia and Project Gutenberg; use and redistribution of these sources should follow the corresponding upstream policies and license conditions. Base model checkpoints (e.g., Qwen3 variants) are also used under their original model terms.

Any release of code, processed data, or model checkpoints derived from this work should preserve attribution requirements and comply with the licenses and usage restrictions of all upstream artifacts. Users are responsible for verifying downstream compliance in their target jurisdiction and deployment setting.

B Choice of Retrieval Metric

To measure how close a generated input x is to the target activation z, we use the *normalized L2*

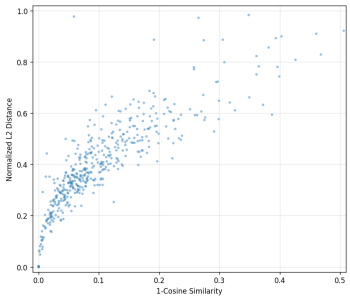

Figure 8: Scatter plot of $1 - \cos(\cdot, \cdot)$ versus normalized L2 distance for samples generated by InverseScope.

distance:

$$d(x, z) = \frac{\|z(x) - z\|_2}{\sqrt{D}},$$

where $z(x)$ denotes the activation produced by input x at the target site, and D is the activation dimension. The $\sqrt{D}$ normalization makes the metric comparable across models of different sizes.

An alternative candidate is cosine dissimilarity, $1 - \cos(z(x), z)$, which is scale-invariant and widely used in representation learning. In our computation, we center activations by subtracting a global mean vector before cosine evaluation, so cosine scores are more stable across samples.

Figure 8 shows a strong monotonic relationship between normalized L2 distance and cosine dissimilarity on samples generated by InverseScope. This indicates that both metrics induce similar rankings in practice when thresholds are calibrated consistently. If points are sampled uniformly at random in activation space, the cosine scores collapse near zero and become much less informative.

C More experiments

C.1 More case studies

Compared with the main-text example (which mainly tests a simple lexical substitution pattern), the cases in this appendix probe more general sentence-level transformations. Each case follows the same analogy-style construction: given a pair of source sentences (neg, pos) and an attribute-transfer sentence add, we form a target activation from $z(\text{add}) + z(\text{pos}) - z(\text{neg})$ and test whether InverseScope recovers the intended target sentence.

The four settings below cover two broader pattern families. First, Tables 4 and 5 test tense and slot-level compositionality: e.g., "The clerk mailed the letter to the manager." $\rightarrow$ "The clerk mails the letter to the manager.", combined with "The clerk mailed the note to the manager." to yield the target "The clerk mails the note to the manager."; and "The lab ran the test on Monday." $\rightarrow$ "The lab runs the test on Monday.", combined with "The lab ran the trial on Monday." to yield "The lab runs the trial on Monday.". Second, Tables 6 and 7 test active–passive voice transfer with simultaneous argument substitution: e.g., "The aide filed the form with the agency." $\rightarrow$ "The form was filed with the agency by the aide.", combined with "The aide filed the report with the agency." to yield "The report was filed with the agency by the aide.", and an analogous panel/grant/prize example.

Across all four cases, the expected target sentence ranks first, indicating that the linear structure observed in the main text is not limited to simple word replacement. Instead, it extends to richer compositional patterns involving tense, lexical slot substitution, and voice alternation, suggesting a relatively strong degree of sentence-level linearity in local activation geometry.

C.2 Neighborhoods

This subsection reports the complete neighborhood for the Munich Airport example discussed in Table 2. Table 8 lists the full prompt ranking within the local activation neighborhood, sorted by L2 distance to the target activation.

D Potential Risks of RAVEL-Style Direction Editing

Our RAVEL result shows that local concept directions extracted through InverseScope can causally modify model behavior. This capability is useful for interpretability research, but it also introduces potential misuse risks. In particular, direction-based interventions may be used to deliberately steer internal representations toward unsafe or policy-violating outputs, or to bypass safety mechanisms tied to specific latent features.

We emphasize that our experiments are conducted in a controlled evaluation setting and are intended for analysis rather than deployment-time manipulation. We do not claim robustness of these interventions across unrestricted real-world settings. As a practical mitigation, we recommend restricting release of high-risk steering artifacts (e.g., reusable direction vectors tied to sensitive behaviors), documenting evaluation scope and failure modes, and auditing intervention effects under safety-focused test suites before any downstream

L2 Dist.	Prompts
0.1262	The clerk mails the note to the manager. *
0.1794	The clerk mails the letter to the supervisor.
0.2412	The clerk sends the letter to the manager.
0.2430	The clerk transmits the letter to the manager.
0.2484	The clerk sends the letter to the hotel manager.
0.2491	The clerk sends the letter to the supervisor.
0.2526	The clerk delivers the letter to the manager.
0.2527	The clerk sends the letter to the mayor.

Table 4: Additional sentence-level analogy example (tense transformation). Prompts are generated by InverseScope and ranked by normalized L2 distance to the target activation. The expected sentence (marked with *) ranks first.

L2 Dist.	Prompts
0.1928	The lab runs the trial on Monday. *
0.2038	The lab runs the test on Monday.
0.2238	The lab runs the test on Tuesday.
0.2262	The lab runs the experiment on Monday.
0.2264	The lab runs the test on Saturday.
0.2373	The lab runs the experiment on Tuesday.
0.2378	The lab runs the test on Wednesday.
0.2379	The lab runs the experiment on Friday.

Table 5: Additional sentence-level analogy example (temporal slot substitution). Prompts are generated by InverseScope and ranked by normalized L2 distance to the target activation. The expected sentence (marked with *) ranks first.

use.

E Use of AI Assistants

We used AI assistant tools to support drafting and editing of manuscript text (e.g., language polishing, formatting, and minor structural revisions). All technical content—including method design, experimental setup, implementation, quantitative results, and final claims—was verified and finalized by the human authors. The authors take full responsibility for the correctness of the reported methods, analyses, and conclusions.

L2 Dist.	Prompts
0.4149	The funding package was approved by the city council. *
0.4215	The grant from the state legislature was approved by the committee.
0.4223	The state legislature appropriation was approved by the committee.
0.4239	The state legislative appropriation was approved by the committee.
0.4328	The governor's grant of funds was approved by the committee.
0.4336	The government appropriations bill was approved by the committee.
0.4340	The state legislature grant of funds was approved by the committee.
0.4353	The state appropriation bill was approved by the committee.

Table 6: Additional sentence-level analogy example (active-to-passive variant I). Prompts are generated by InverseScope and ranked by normalized L2 distance to the target activation. The expected sentence (marked with *) ranks first.

L2 Dist.	Prompts
0.2545	The prize was awarded to the team by the panel. *
0.2563	The award was given to the team by the panel.
0.2599	The award was bestowed on the team by the panel.
0.2627	The award was conferred on the team by the panel.
0.2700	The prize was bestowed on the team by the panel.
0.2748	The award was bestowed on the group by the panel.
0.2826	The award was presented to the team by the panel.
0.3270	The award was bestowed by the panel by the judges.

Table 7: Additional sentence-level analogy example (active-to-passive variant II). Prompts are generated by InverseScope and ranked by normalized L2 distance to the target activation. The expected sentence (marked with *) ranks first.

L2 Dist.	Prompts
0.0000	In 1998 the western link to Munich Airport was opened as a two-track line which branches off north of Neufahr.*
0.1309	In 1998 the western link to Munich Airport was completed with the construction of a new bridge to the north of Neufahr.
0.1445	In 1998 the western link to Munich Airport was opened to the north of Neufahr.
0.1455	In 1998 the western link to Munich Airport (Munich International Airport) was opened to the north of Neufahr.
0.1523	In 1998 the western link to Munich Airport was completed with the extension of the Autobahn A9 to the north of Neufahr.
0.1553	In 1998 the western link to Munich Airport was completed with the extension of the Autobahn A 99 to the south of Neufahr.
0.1689	In 1998 the western link to Munich Airport was opened with the extension of the S-Bahn line S8 to the north of Neufahr.
0.1709	In 1998 the western link to Munich Airport was opened with the extension of the S-Bahn line S8 to the south of Neufahr.
0.1738	In 1998 the western link to Munich Airport was established with the extension of the S-Bahn line S8 to the north of Neufahr.

Table 8: Prompts generated by InverseScope conditioned on the target activation at the Munich Airport western-link sentence, ranked by L2 distance to the target. The ground-truth sentence (marked with *) is the reference text; the row at L2$= 0.1553$ is the largest token-edit outlier in the neighborhood (edit distance $= 16$).