VASAE: Naming SAE Dictionary Directions with Vocabulary-Aligned Anchoring

Kairui Zhang [1] **Ziwen Yu** [2] **Zahraa S. Abdallah** [1] **Martha Lewis** [3]

Abstract

Sparse autoencoders (SAEs) provide useful decompositions of Transformer residual streams, but their learned features are usually named post hoc rather than directly connected to the Transformer's token vocabulary. We introduce Vocabulary-Aligned Sparse Autoencoder (VASAE), a method that trains SAE features under vocabulary-aligned anchoring and assigns each feature an intrinsic token name: the token string whose embedding is nearest to that feature. Without reducing reconstruction quality compared with a standard SAE, VASAE produces dictionaries with vocabulary-aligned features. Using a 0.8 cutoff on the nearest-token alignment score, dictionaries trained on GPT-2-small post-residual streams align about 90% of features in layers 0–10. In Llama-3.1-8B, representative shallow and middle-layer dictionaries contain strongly aligned features, including 92.8% in the shallow layer, while the representative final-layer dictionary shows limited alignment. After subtracting the sentence-level mean sparse code, case studies show that many remaining intrinsic token names are relevant to nearby input tokens. These results suggest that vocabulary-aligned anchoring can connect learned features to intrinsic token names during training, complementing post hoc interpretation of learned dictionaries.

1. Introduction

Transformer architectures are a central model class for generative AI (Vaswani et al., 2017; Brown et al., 2020), and language-model performance has improved with model and data scale (Hoffmann et al., 2022). Their behavior, however, is produced by high-dimensional internal states that are difficult to inspect directly (Belinkov & Glass, 2019; Ghandeharioun et al., 2024). In decoder-only Transformers, one important state is the *residual stream*: the vector channel that each Transformer block reads from and writes to (Elhage et al., 2021). Understanding this stream matters because it carries the intermediate information from which later layers compute next-token predictions (Geva et al., 2021; Ghandeharioun et al., 2024).

Sparse Autoencoders (SAEs) are increasingly used to decompose residual streams into sparse codes over learned dictionaries (Huben et al., 2024; Gao et al., 2025). In this paper, the dictionary is the decoder weight matrix, and a *feature* is one column, or direction, of that matrix. A *sparse code* is a vector of feature coefficients. Each entry corresponds to one feature. For a given residual-stream vector, most entries are zero, and a larger nonzero value means that the corresponding feature contributes more strongly to the reconstruction. Recent work uses SAE features for circuit discovery as interpretability tools (Marks et al., 2025; Karvonen et al., 2025). Standard SAEs, however, learn features for reconstruction and sparsity rather than direct naming (Huben et al., 2024; Gao et al., 2025). Feature labels are usually assigned after training by inspecting contexts where the corresponding element of the sparse code is large, or by running a separate automated interpretation procedure (Bills et al., 2023; Paulo et al., 2025), making interpretation a separate post-hoc step. Vocabulary-Aligned Sparse Autoencoder (VASAE) assigns a training-time nearest-token label to each learned decoder direction.

The technical bottleneck is that the dictionary geometry and the vocabulary geometry are usually disconnected. Prior work shows that language-model representation spaces have nontrivial geometric structure, including anisotropy and dominant directions (Ethayarajh, 2019; Mu & Viswanath, 2018). Separately, weight-tying work shows that input and output embeddings are a central vocabulary interface in language models (Press & Wolf, 2017; Inan et al., 2017). A learned feature can reconstruct activations while lying far from any token embedding, so it has no intrinsic vocabulary-level name. The opposite design would force the decoder to equal the token embedding matrix; in this paper we test

[1]Intelligent Systems Laboratory, University of Bristol, Bristol, UK [2]University of Bristol, Bristol, UK [3]University of Amsterdam, Amsterdam, Netherlands. Correspondence to: Kairui Zhang <pu22650@bristol.ac.uk>.

Accepted to the Mechanistic Interpretability Workshop at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

whether this hard-tied decoder is sufficient. The missing object is therefore a learnable dictionary that remains flexible for reconstruction while staying close enough to vocabulary directions to provide candidate token-based names. A token name is a geometry-derived label for a direction.

Our key observation is that the language model already contains a model-internal, vocabulary-indexed set of reference directions when the token embeddings share dimensionality with the SAE decoder space: the fixed input token embeddings (Vaswani et al., 2017). We use these embeddings as anchors, not as frozen features. Based on this observation, we propose VASAE, a SAE whose learned features are trained with the usual sparse reconstruction objective plus a vocabulary-aligned anchor objective. Each feature is encouraged to stay close to its nearest token embedding, and after training receives an *intrinsic token name*: the token string whose embedding is nearest to that feature. Here intrinsic means that the name is induced by the trained feature's location relative to the model's own vocabulary embeddings.

Empirically, VASAE achieves reconstruction metrics comparable to a standard SAE under the tested settings while adding a geometric vocabulary-alignment signal. Across layers, it has comparable variance explained (VE) on GPT-2-small (0.965) and Llama-3.1-8B (0.931), while hard-tying the decoder to token embeddings has lower reconstruction metrics. On GPT-2-small, 89–94% of features in layers $L0$–$L10$ exceed the diagnostic alignment cutoff. With a larger tested anchor coefficient ($\lambda_{\text{anchor}} = 5 \times 10^{-3}$), Llama-3.1-8B reaches 92.8% at $L0$ but remains unstable in the final representative layer. Case studies show intrinsic token names assigned to active features in context.

Our contributions are:

- We introduce VASAE: a soft vocabulary-anchoring objective that assigns learned SAE decoder directions nearest-token names.
- We evaluate reconstruction under vocabulary anchoring and test whether hard-tying the decoder to token embeddings is sufficient.
- We analyze embedding-space feature alignment and vocabulary coverage, and use case studies to show intrinsic token names as geometry-derived identifiers.

2. Background

Residual-stream SAEs, token embeddings, and vocabulary readouts all depend on the geometry of Transformer residual states. VASAE uses this geometry to define vocabulary-indexed anchors for learned SAE decoder directions.

2.1. Transformer Residual Stream

Transformer models maintain a persistent residual stream, which carries information across layers (Vaswani et al., 2017; Elhage et al., 2021). Panel A of Figure 1 shows the standard Transformer structure used in this setup. Let $\mathcal{V}$ be the vocabulary space and let $\mathbf{W}_E \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{model}}}$ be the token embedding matrix, where d_{model} is the model dimension. Throughout the paper, we use a row-vector convention to stay consistent with implementations. For a token $v \in \mathcal{V}$, we denote its embedding by $\mathbf{w}_v \in \mathbb{R}^{d_{\text{model}}}$ and write the initial post-residual stream as $\mathbf{h}^{(0)} = \mathbf{w}_v$.

At each layer ℓ, the residual stream is updated by adding the outputs of the attention and multi-layer perceptron (MLP) modules:

$$\mathbf{h}^{(\ell)} = \mathbf{h}^{(\ell-1)} + \mathrm{Attn}_\ell(\mathbf{h}^{(\ell-1)}) + \mathrm{MLP}_\ell(\mathbf{h}^{(\ell-1)}). \quad (1)$$

This schematic update omits the sequential sublayer structure and LayerNorm operations of the underlying Pre-LN Transformer blocks.

We call $\mathbf{h}^{(\ell)} \in \mathbb{R}^{d_{\text{model}}}$ the layer-ℓ post-residual stream: the residual stream after layer ℓ has added its attention and MLP outputs. In this work, all SAE variants are trained and evaluated on the post-residual stream, not on the intermediate attention or MLP outputs before they are added back into the residual stream.

For autoregressive language models, the final prediction logits are obtained by projecting the last-layer post-residual stream through the unembedding matrix $\mathbf{W}_U \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{model}}}$:

$$\mathbf{u} = \mathbf{h}^{(L)} \mathbf{W}_U^\top. \quad (2)$$

Embedding and unembedding weight tying is a standard language-model design choice (Press & Wolf, 2017; Inan et al., 2017). In the Generative Pre-trained Transformer 2 (GPT-2)-small checkpoint used here (Radford et al., 2019), the embedding and unembedding matrices are tied ($\mathbf{W}_U = \mathbf{W}_E$). In the Llama-3.1-8B checkpoint used here (Grattafiori et al., 2024), they are not. For models with untied input embeddings and output unembeddings, the choice of anchor matrix is not unique. Input embeddings provide vocabulary-indexed residual directions near the model input, whereas unembedding directions may be more appropriate for later layers. In this work we focus on input-embedding anchors and treat unembedding-based anchors as a possible extension.

2.2. Vocabulary-Aligned Probing

A common diagnostic for Transformer residual streams is to project intermediate post-residual streams into vocabulary-logit space. In our auxiliary LogitLens-style readout, matching the implementation used for the reported readout metric,

we directly project the post-residual stream from intermediate layers through the unembedding matrix:

$$\mathbf{u}^{(\ell)} = \mathbf{h}^{(\ell)}\mathbf{W}_U^\top. \tag{3}$$

This measures which token logits are linearly decodable from an intermediate post-residual stream under this direct readout. It is separate from the cross-entropy (CE) recovery metric, where the patched residual stream is passed through the remaining model computation before final logits are read. This background motivates the reference frame used by VASAE: vocabulary-indexed directions can be used to relate residual-stream geometry to token space. Unlike a logit-lens readout, VASAE does not decode activations into token predictions; it softly anchors learned SAE decoder directions to vocabulary-indexed reference directions.

2.3. SAEs for Residual-Stream Analysis

SAEs map a post-residual stream to a sparse code and then decode the sparse code back into residual-stream space. Panel B of Figure 1 shows this encoder–decoder structure. SAE variants differ in how they parameterize the encoder and enforce sparsity. This paper uses a top-k SAE (Makhzani & Frey, 2014; Gao et al., 2025), where sparsity is achieved by keeping only the largest k encoder coordinates after ReLU.

The encoder uses $\mathbf{W}_\mathcal{E} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{sparse}}}$ and $\mathbf{b}_\mathcal{E} \in \mathbb{R}^{d_{\text{sparse}}}$, applying a learned affine map followed by ReLU and top-k selection:

$$\mathbf{z} = \mathcal{E}(\mathbf{h}) = \mathrm{TopK}_k(\mathrm{ReLU}(\mathbf{h}\mathbf{W}_\mathcal{E} + \mathbf{b}_\mathcal{E})). \tag{4}$$

Here $\mathcal{E}$ denotes the full sparse encoder, including the ReLU nonlinearity and top-k selection. The vector $\mathbf{z}$ is the sparse code. Each scalar $\mathbf{z}_i$ is one element in the sparse code vector and is the coefficient for feature i. The operator TopK_k keeps the largest k coordinates and sets all others to zero, so each sparse code satisfies $\|\mathbf{z}\|_0 \le k$, where $\|\cdot\|_0$ denotes the number of nonzero coordinates.

The decoder uses $\mathbf{W}_\mathcal{D} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{sparse}}}$ and $\mathbf{b}_\mathcal{D} \in \mathbb{R}^{d_{\text{model}}}$ to map the sparse code back to the residual-stream space:

$$\tilde{\mathbf{h}} = \mathbf{z}\mathbf{W}_\mathcal{D}^\top + \mathbf{b}_\mathcal{D}. \tag{5}$$

The online experiments use the extracted post-residual streams directly; we do not subtract a separate stored activation mean before the encoder. Constant offsets can be represented by the learned affine encoder and decoder bias terms.

For a training set of N post-residual streams $\{\mathbf{h}_i\}_{i=1}^N$, where N counts evaluated token positions, the SAE training loss is the empirical reconstruction error under this sparse encoder:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{h}_i - \tilde{\mathbf{h}}_i\right\|_2^2. \tag{6}$$

Sparsity means only a few elements of the sparse code are nonzero for a given post-residual stream, yielding a decomposition with a small active feature set. In Section 3, we keep the same reconstruction problem with sparse codes and add vocabulary-aligned anchoring to the dictionary.

3. Method

We decompose the post-residual stream with a learned dictionary whose features are anchored to the fixed token-embedding directions by a vocabulary-aligned objective. Sparsity enters through the sparse code: for a given post-residual stream, only a few dictionary features have nonzero corresponding elements in the sparse code. Our proposed VASAE method keeps the SAE dictionary learnable and uses token embeddings only as naming anchors. The method is summarized in Figure 1.

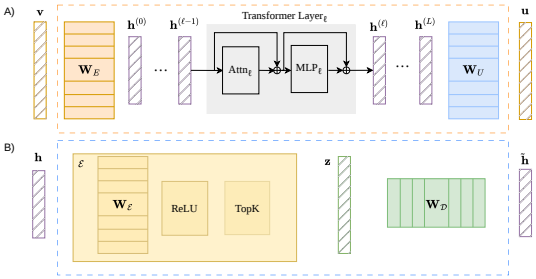

Figure 1. Transformer residual-stream decomposition with VASAE. The model learns the dictionary $\mathbf{W}_\mathcal{D}$ under a reconstruction objective plus the vocabulary-anchor loss $\mathcal{L}_{\text{anchor}}$.

3.1. VASAE Architecture

For each model layer, we train a separate VASAE model on the post-residual stream from that layer. The input is $\mathbf{h} \in \mathbb{R}^{d_{\text{model}}}$, and the encoder is the same linear ReLU top-k map as in Equation 4. This produces a nonnegative sparse code $\mathbf{z} \in \mathbb{R}^{d_{\text{sparse}}}$ with at most k nonzero elements. The reconstruction is produced by the learned decoder $\mathbf{W}_\mathcal{D}$ as in Equation 5.

We write the dictionary as $\mathbf{W}_\mathcal{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_{d_{\text{sparse}}})$, where $\mathbf{d}_i \in \mathbb{R}^{d_{\text{model}}}$ is feature i. The scalar $\mathbf{z}_i$ is one element in the sparse code vector and is the coefficient for feature i.

The sparse code dimension d_{sparse} is a configurable VASAE hyperparameter. In the experiments, we instantiate the vocabulary-sized case $d_{\text{sparse}} = |\mathcal{V}|$, so the learned dictionary has one feature slot per vocabulary item. Other

dictionary sizes are compatible with the method. The decoder columns remain learnable, and token embeddings are fixed reference directions during VASAE training. We use vocabulary-sized dictionaries in the main experiments to make the learned feature set and vocabulary anchor set comparable in scale.

3.2. Vocabulary-Aligned Anchoring

VASAE assumes that SAE decoder directions and the chosen vocabulary anchor directions live in the same dimensional vector space. For residual-stream SAEs, this assumption holds when the model's token embedding vectors have the same width as the residual stream. Architectures with factorized embeddings or mismatched dimensions would require an explicit learnable projection matrix that maps token embeddings to the residual-stream dimension, or an anchor set already defined in the residual-stream space.

For each learned feature $\mathbf{d}_i$, VASAE compares the feature with every fixed token embedding and records both the best cosine similarity and the vocabulary item that attains it:

$$s_i = \max_{v \in \mathcal{V}} \cos(\mathbf{d}_i, \mathbf{w}_v), \tag{7}$$

$$v_i^\star = \arg\max_{v \in \mathcal{V}} \cos(\mathbf{d}_i, \mathbf{w}_v). \tag{8}$$

Thus, s_i is the nearest-token alignment score: it is large only when feature i lies close to some vocabulary direction. The vocabulary item $v_i^\star$ determines the intrinsic token name for the feature. The feature vector $\mathbf{d}_i$ itself remains free to move under the reconstruction objective. Cosine similarities are computed after L_2-normalizing decoder columns and token embedding vectors. For each decoder column, we compute cosine similarity to all token embeddings. The nearest token embedding defines the current anchor. The loss below rewards high similarity to this anchor and is optimized together with reconstruction under the top-k sparse-code constraint.

To increase nearest-token alignment, we define the anchor loss so that minimizing it increases the cosine similarity between each feature and its nearest token embedding:

$$\mathcal{L}_{\text{anchor}} = -\frac{1}{d_{\text{sparse}}} \sum_{i=1}^{d_{\text{sparse}}} s_i. \tag{9}$$

Combined with reconstruction loss in Equation 6, the total loss of VASAE is

$$\mathcal{L}_{\text{vasae}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{anchor}} \mathcal{L}_{\text{anchor}}, \tag{10}$$

During optimization, $v_i^\star$ is recomputed from the current feature vector $\mathbf{d}_i$, so the nearest-token anchor can change as the dictionary changes. The token embeddings serve only as fixed reference vectors.

The anchor coefficient sets the weight of the vocabulary-alignment term relative to reconstruction. The top-k constraint fixes the number of active features separately. Larger λ_{anchor} values pull decoder directions more strongly toward token embeddings, so we tune λ_{anchor} as a regularization hyperparameter.

For large vocabularies, we compute the nearest-token scores and lookup with a memory-bounded chunked similarity implementation. The chunked computation avoids materializing the full feature-by-vocabulary similarity matrix. Details are given in Appendix F.

3.3. Feature Naming

After training, each feature $\mathbf{d}_i$ receives an intrinsic token name. The name is the token string corresponding to the nearest vocabulary item $v_i^\star$ in Equation 8. We use the term intrinsic token name to mean the nearest vocabulary token under cosine similarity in the chosen embedding space. This is a geometric identifier for the feature. We call this name intrinsic because it is determined by the geometry between the learned feature vector and the fixed token-embedding matrix, rather than by manually inspecting contexts where elements of the sparse code are large or by applying an additional post-hoc labeling procedure.

We report s_i together with the intrinsic token name as the feature's alignment score. A larger s_i means that the feature vector is closer to its named token embedding. A smaller s_i means that the nearest token still gives the closest vocabulary direction, but with lower geometric support.

4. Experiments

We evaluate whether vocabulary-aligned anchoring largely preserves SAE reconstruction, learns dictionaries with token-aligned features, and assigns token names to features with large sparse-code coefficients.

4.1. Experimental Setup

Dataset. Our training and evaluation data are post-residual-stream activations extracted from language-model forward passes on WikiText-103 (Merity et al., 2017). We split sequences into train/validation/test subsets as follows: $50{,}000/10{,}000/5{,}000$ for GPT-2-small and $20{,}000/2{,}000/5{,}000$ for Llama-3.1-8B. Each sequence is truncated or padded to a maximum length of 128, giving at most about 6.4M/1.3M/0.6M token positions for GPT-2-small and 2.6M/0.3M/0.6M token positions for Llama-3.1-8B before padding positions are masked. We run the language model on the tokenized sequences and

extract post-residual-stream activations online with the `nnsight` (Fiotto-Kaufman et al., 2025) tracing framework. We use two decoder-only Transformer checkpoints to cover a smaller GPT-2 setting and a larger Llama setting:

- **GPT-2-small**: a GPT-2 checkpoint used for full layer-wise analysis. It has model dimension $d_{\text{model}} = 768$, vocabulary size $|\mathcal{V}| = 50{,}257$, and 12 layers (Radford et al., 2019).
- **Llama-3.1-8B**: a larger open-weight Llama checkpoint used to test the method in a higher-dimensional vocabulary and residual-stream geometry. It has model dimension $d_{\text{model}} = 4096$, vocabulary size $|\mathcal{V}| = 128{,}256$, and 32 layers (Grattafiori et al., 2024).

VASAE and all baselines are trained and evaluated on these extracted post-residual streams. Padding positions are excluded from training and evaluation with the tokenizer attention mask.

Baselines.

- **Standard SAE**: a SAE trained directly on post-residual streams without explicit anchoring to fixed token-embedding directions.
- **Hard-tied decoder baseline**: a comparison method that uses exact token embeddings as features. The decoder is fixed during training. Only the encoder is optimized.

The hard-tied decoder baseline tests the direct alternative of representing post-residual streams only with token-embedding features. Because its decoder features are exactly the vocabulary embeddings, it necessarily has $d_{\text{sparse}} = |\mathcal{V}|$. It sets $\mathbf{W}_{\mathcal{D}}^{\top} = \mathbf{W}_E$ in Equation 5 and reconstructs as $\tilde{\mathbf{h}} = \mathbf{z}\mathbf{W}_E$. The encoder is trained, but the decoder weights are frozen. This baseline tests whether hard-tying the decoder to token embeddings is sufficient.

4.2. Reconstruction Preservation

We first ask whether vocabulary-aligned anchoring changes reconstruction quality relative to a standard SAE. We report reconstruction error and cross-entropy loss after residual-stream substitution.

Training protocol. All three models, standard SAE, hard-tied SAE, and VASAE, use vocabulary-sized sparse code dimension $|\mathcal{V}|$, top-k sparsity with $k = 32$, nonnegative sparse code, a linear encoder, and Adam with learning rate 10^{-3}. For standard SAE and VASAE, the vocabulary-sized dimension is chosen to match the hard-tied baseline. Training runs for at most 20 epochs with patience-3 early stopping on validation reconstruction loss, and we evaluate the best checkpoint on the test split. For VASAE, the anchor term is optimized during training but is not included in validation checkpoint selection. The default anchor coefficient is $\lambda_{\text{anchor}} = 10^{-4}$. For Llama-3.1-8B geometric-alignment and case-study diagnostics, we additionally use $\lambda_{\text{anchor}} = 5 \times 10^{-3}$; the supporting anchor-strength sweep is reported in Appendix B. GPT-2-small runs use float32 with batch size 32. Llama-3.1-8B runs use bfloat16 with batch size 8; for Llama VASAE runs, the anchor term is evaluated once every 50 optimizer steps, and skipped-anchor steps optimize only the reconstruction loss. Each reported training run uses a single NVIDIA GH200 GPU with 96GB GPU memory. Nearest-token lookup is computed over the model tokenizer's full vocabulary. Padding positions are excluded from activation training and evaluation through the attention mask.

Evaluation metrics. Table 1 reports four metrics. Exact formulas are given in Appendix A.

- **Mean Squared Error (MSE)**: residual-stream reconstruction error. Lower values mean the reconstruction is closer to the target activation.
- **Variance Explained (VE)**: scale-normalized reconstruction quality. Higher values mean the reconstruction preserves more target activation variance.
- **Cross-Entropy Loss (CE loss)**: downstream next-token loss after substituting the reconstruction at the evaluated residual-stream site. Lower values mean better preservation of model predictions.
- **Cross-Entropy Recovery (CE rec.)**: downstream loss recovery relative to the gap between the original residual stream and a zero-vector control. Values near 1 match the original stream, values near 0 match the zero-vector control, and negative values are worse than that control.

Table 1 reports mean $\pm$ standard deviation across layers. The standard deviations are across layers rather than random seeds. VASAE has reconstruction metrics comparable to a standard SAE in both tested model families. On GPT-2-small, the two learned-dictionary methods have the same VE (0.965) and CE recovery (0.975). On Llama-3.1-8B, they also have the same VE (0.931) and CE recovery (0.906). The hard-tied decoder baseline performs much worse, showing that directly forcing every feature to be an exact token embedding is not sufficient. This contrast supports the central design choice in VASAE: keep the dictionary learnable while anchoring features to token embeddings. Layerwise VE, CE recovery, and auxiliary LogitLens diagnostics are shown in Appendix C.

Table 1. Results, reported as mean ± standard deviation across layers. Arrows indicate the preferred direction for each metric. Shaded rows denote the proposed method. Best values within each model block are bolded.

Model	Method	MSE ↓	VE ↑	CE loss ↓	CE rec. ↑
GPT-2-small	Standard SAE	$\mathbf{2.04 \pm 2.78}$	$\mathbf{0.965 \pm 0.053}$	4.18 ± 0.11	$\mathbf{0.975 \pm 0.028}$
GPT-2-small	Hard-tied decoder	110.39 ± 62.53	-0.434 ± 1.046	14.64 ± 10.65	-0.606 ± 2.642
GPT-2-small	**VASAE**	$\mathbf{2.04 \pm 2.78}$	0.965 ± 0.054	$\mathbf{4.17 \pm 0.11}$	$\mathbf{0.975 \pm 0.028}$
Llama-3.1-8B	Standard SAE	$\mathbf{0.057 \pm 0.096}$	$\mathbf{0.931 \pm 0.091}$	3.55 ± 0.68	0.906 ± 0.075
Llama-3.1-8B	Hard-tied decoder	0.672 ± 0.236	-0.021 ± 0.058	11.46 ± 0.92	0.032 ± 0.101
Llama-3.1-8B	**VASAE**	0.058 ± 0.097	$\mathbf{0.931 \pm 0.091}$	$\mathbf{3.55 \pm 0.67}$	$\mathbf{0.906 \pm 0.074}$

4.3. Geometric Token Alignment

Having evaluated reconstruction, we next measure whether learned features become geometrically token aligned using the alignment score defined in Equation 7. This section checks whether the anchor loss produces token-aligned decoder directions.

Evaluation setup. For GPT-2-small, we compare VASAE with a standard SAE using $\lambda_{\text{anchor}} = 10^{-4}$. For Llama-3.1-8B, we use $\lambda_{\text{anchor}} = 5 \times 10^{-3}$ because $\lambda_{\text{anchor}} = 10^{-4}$ preserves reconstruction but produces lower token alignment scores; the appendix sweep shows that 5×10^{-3} changes VE and CE recovery only slightly on representative layers. For both model families, we use the alignment score s_i and its associated nearest vocabulary item $v_i^\star$, defined in Equation 7 and Equation 8, respectively. With threshold $\tau = 0.8$, the strongly aligned feature-index set is $\mathcal{A}_\tau = \{i : s_i \geq \tau\}$. This value was chosen from the observed separation in alignment-score distributions: standard SAE features concentrate well below 0.3, while many VASAE features cluster near 1.0, as shown in Figure 2. The threshold is a diagnostic cutoff for geometric alignment; the full score distribution remains the primary evidence. We report the geometric feature-alignment rate $|\mathcal{A}_\tau|/d_{\text{sparse}}$ and vocabulary coverage $|\{v_i^\star : i \in \mathcal{A}_\tau\}|/|\mathcal{V}|$, since many features can be aligned to the same token. Coverage is the fraction of vocabulary tokens used as nearest-token names by at least one strongly aligned feature. The denominator is the full-learned dictionary, so this rate should be read as a geometric property of the feature set.

GPT-2-small shows high alignment across most layers. For GPT-2-small, Figure 2 left summarizes representative even-numbered layers from $L0$ to $L10$. The alignment distributions are clearly separated: about 93% of VASAE features in shallow and middle layers have alignment scores near 1.0, whereas standard SAE features remain concentrated around 0.1–0.2 and never exceed the diagnostic cutoff. Under this criterion, post-hoc nearest-token naming of a standard SAE is a low-alignment baseline: the nearest token exists by definition, but it is typically far from the learned feature direction. Across layers, the geometric feature-alignment rate stays between 89% and 94% for $L0$–$L10$ and is 68.5% in $L11$. Coverage of unique aligned tokens is around 53%–56%, indicating that multiple features can share an intrinsic token name. At this coverage level, the naming map is many-to-one: many aligned features share nearest-token names. These results support the main alignment claim: in GPT-2-small, vocabulary-aligned anchoring converts a standard SAE dictionary into a geometrically token-aligned dictionary with comparable reconstruction metrics.

Llama-3.1-8B alignment is shallower. At $\lambda_{\text{anchor}} = 5 \times 10^{-3}$, the right panel of Figure 2 shows a depth-dependent pattern: alignment is highest at $L0$, lower at $L15$, and not comparable at $L31$. The $L0$ dictionary has 119,016 strongly aligned features out of 128,256 (92.8%), with 77,371 unique aligned token names (60.33% vocabulary coverage). These Llama-3.1-8B results indicate higher alignment in shallow layers and less stable alignment in deeper layers, matching the hypothesis that vocabulary anchoring is more compatible with residual states closer to embedding space.

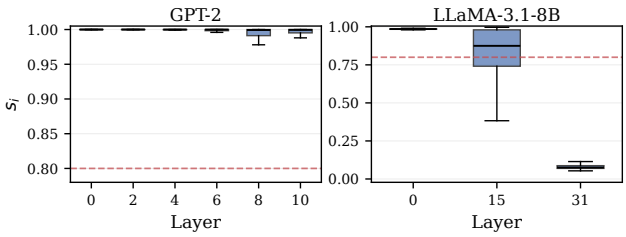

Figure 2. Geometric alignment diagnostics. Left: GPT-2-small alignment score distributions for representative even-numbered layers from $L0$ to $L10$. Right: Llama-3.1-8B alignment score distributions at $\lambda_{\text{anchor}} = 5 \times 10^{-3}$ for $L0$, $L15$, and $L31$. The dashed line marks the diagnostic strong-alignment threshold $s_i = 0.8$.

4.4. Case Studies of Intrinsic Token Names

Case-study setup. We next inspect the intrinsic token names assigned to aligned features that are active in context. These case studies show token-level intrinsic names on individual prompts. The main location example uses the fixed input sentence "The cafe is located on Baker Street,

just around the corner from the avenue". We tokenize this sentence with each model's tokenizer and display the first 12 non-padding token positions. The GPT-2-small visualization uses representative even-numbered layers from $L0$ to $L10$; the Llama-3.1-8B visualization uses layers $L0, L15, L31$. For each displayed layer ℓ, let $\mathbf{z}_p \in \mathbb{R}^{d_{\text{sparse}}}$ be the sparse code coefficient at token position p in a sentence of length T, and let $\mathcal{A}_\tau$ be the strongly aligned feature-index set from the previous subsection. If a feature is active at many positions in the same sentence, raw activation alone can choose that same feature repeatedly for display. We therefore choose the aligned feature that is most elevated relative to the sentence average:

$$\bar{\mathbf{z}} = \frac{1}{T}\sum_{q=1}^{T}\mathbf{z}_q, \qquad i_p^\star \in \arg\max_{i\in\mathcal{A}_\tau}[\mathbf{z}_p - \bar{\mathbf{z}}]_i. \tag{11}$$

We display the intrinsic token name of the chosen feature, namely the token string corresponding to $v_{i_p^\star}^\star$. The relative sparse code is used only to choose the displayed feature. The color shows the raw sparse-code activation $[\mathbf{z}_p]_{i_p^\star}$.

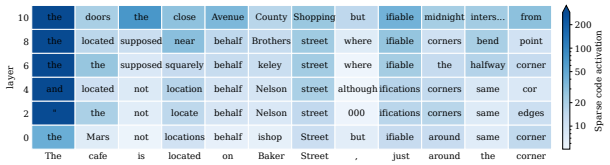

Figure 3. GPT-2-small location case study. Each cell shows the intrinsic token name of the aligned feature with the largest sentence-centered sparse-code value at a layer and input position. Color indicates the raw sparse-code activation of that feature. Shallow and middle layers show location-related intrinsic token names.

GPT-2-small example. Figure 3 shows a representative GPT-2-small location example. Shallow and middle layers repeatedly display location-related intrinsic token names such as *located*, *location*, *Street*, *around*, *corner*, and *corners* around the input phrase "Baker Street" and the surrounding spatial context. Additional examples are shown in Appendix E, including adjective/adverb words, award-related words, and self-introduction words. These visualizations show intrinsic token names after geometric alignment has been established and provide qualitative inspection examples.

Llama-3.1-8B example. Figure 4 applies the same sentence-centered rule to Llama-3.1-8B at $\lambda_{\text{anchor}} = 5 \times 10^{-3}$ for the representative layers $L0$, $L15$, and $L31$. The shallow layer shows location-related token names, the middle layer is less consistent, and the final layer is mostly unstable. This qualitative pattern matches the geometric diagnostics in Figure 2.

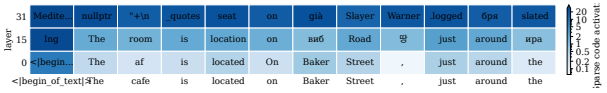

Figure 4. Llama-3.1-8B case study at $\lambda_{\text{anchor}} = 5\times10^{-3}$. Each cell shows the intrinsic token name of the feature chosen by the same sentence-centered sparse-code rule as Figure 3. Color indicates the raw sparse-code activation of that feature. The shallow representative layer shows location-related token names, the middle representative layer is less consistent, and the final representative layer is unstable.

5. Discussion

What VASAE names do and do not mean. A VASAE name is a vocabulary-indexed handle for a learned decoder direction: the token whose embedding is nearest to that direction. Its role is to attach a model-internal token reference to the feature during training, rather than adding a separate naming step after feature learning.

The main point to avoid over-reading is that this token handle is not a feature explanation. It gives a starting point for inspection, but the meaning or role of the feature still has to be established from how it activates, how it connects to other model components, or what changes under intervention.

What counts as explaining a feature. Feature interpretation depends on the research goal. The examples below illustrate a distinction that is easy to blur in SAE analysis: a feature can be useful because information is readable from it, because it participates in an internal computation, or because intervening on it changes behavior. Each use calls for its own evidence. This matters because a named feature can make an interpretation look more complete than it is. A safer use of feature names is to state which question the feature is being used to answer and what evidence supports that question.

For example, **semantic readout** treats a feature as a direction from which some information may be readable. A feature whose nearest token is location-related may be a candidate direction for residual-stream information about place names, street names, spatial prepositions, or local descriptions of where something is; evidence for this use comes from top-activating contexts, activation distributions across prompts, and comparisons with nearby candidate features. **Computational role** asks whether the feature participates in an internal calculation, such as tracking an intermediate variable for entity resolution, maintaining a syntactic constraint, combining local context into a next-token preference, or passing information from one component to another; this requires circuit-level evidence such as ablations, activation patching, path patching, or dependencies between upstream and downstream features and modules. **Behavioral control** asks whether intervening on the feature changes outputs in a

predictable way, for example shifting generated text toward a more formal, cautious, specific, or creative style. In that setting, the interpretation is tied to the intervention effect, not only to the contexts where the feature activates or to its nearest-token name.

For the goal of understanding how a language model implements a behavior, this suggests shifting attention from whether an isolated feature has the right name to how named features enter circuits. The central object is the mechanism: how features, attention heads, MLP components, and residual-stream directions pass information forward, transform it, and affect behavior. In this view, feature naming is a preliminary indexing tool. The analysis should ask which components use the feature, which downstream paths change when it is perturbed, and which behavior is explained by the resulting circuit.

6. Related Work

6.1. Vocabulary Readout and Representation Geometry

Probing work formalizes how to test whether intermediate post-residual streams linearly encode linguistic or task-relevant attributes (Tenney et al., 2019; Hewitt & Liang, 2019). In autoregressive Transformers, layerwise vocabulary-readout methods, from LogitLens-style readouts to tuned-lens variants, show how token-level predictions can be decoded and analyzed across Transformer layers (Belrose et al., 2025). Our work is complementary to this literature: rather than studying decodability alone, we study decompositions of the residual stream itself.

Related work on representation geometry studies how information is arranged in language-model vector spaces, including whether representations are anisotropic or dominated by common directions (Ethayarajh, 2019; Mu & Viswanath, 2018). These works motivate geometric analysis of language-model representations. We use token embeddings as the reference frame because they provide vocabulary-indexed directions in the residual-stream space, while weight-tying work shows that embedding and output-token geometry are central interfaces in language models (Press & Wolf, 2017; Inan et al., 2017). Our question is more specific: can a learned dictionary stay close enough to vocabulary directions to provide intrinsic token names while still reconstructing the post-residual stream from sparse codes, without being forced into the exact token-embedding directions? This motivates our anchor choice: when token embeddings share dimensionality with the residual-stream SAE decoder, they define vocabulary-indexed directions against which learned decoder columns can be compared.

6.2. Sparse Codes and Mechanistic Analysis

Superposition motivates sparse feature dictionaries as a way to separate features that share model dimensions (Elhage et al., 2022). Dictionary-learning work decomposes language-model activations into sparse features that are more interpretable than individual neurons (Bricken et al., 2023; Huben et al., 2024). Subsequent work scales and evaluates SAEs across language-model settings (Gao et al., 2025; Karvonen et al., 2025), and sparse feature circuits use these features for causal graph analysis and editing (Marks et al., 2025). Existing interpretation workflows usually inspect top-activating contexts or apply automated natural-language explanations after training (Bills et al., 2023; Paulo et al., 2025). Mechanistic interpretability work more broadly studies how Transformer computations factor into internal mechanisms, including module-level accounts such as feed-forward key–value memory (Geva et al., 2021). Our contribution keeps the SAE reconstruction objective and adds vocabulary-aligned anchoring, so learned features can receive candidate intrinsic token names. The main claim is reconstruction-preserving geometric token alignment.

7. Conclusion

We presented VASAE, a residual-stream SAE whose learned dictionary is vocabulary-aligned to token embeddings. The difference from a standard SAE is the anchor term: both models learn sparse codes to reconstruct the post-residual stream, but VASAE also encourages each feature to remain close to a nearest fixed token-embedding direction, giving the feature an intrinsic token name. This added constraint yields reconstruction metrics comparable to a standard SAE in the tested GPT-2-small and Llama-3.1-8B settings, while producing strongly token-aligned features in GPT-2-small and in shallow- and middle-layer Llama settings at $\lambda_{\text{anchor}} = 5 \times 10^{-3}$. Case studies show intrinsic token names in shallow and middle GPT-2 layers, including location words, award-related words, self-introduction words, and adjective/adverb words. Llama-3.1-8B also shows the main boundary condition: token alignment depends on anchor coefficient and remains unstable in the final layer under the tested settings. Overall, VASAE keeps the sparse reconstruction role of a standard SAE but adds geometrically supported nearest-token names for features.

Limitations. Our analysis is limited to residual-stream SAEs on two open-weight language-model families. The main alignment evidence is highest for GPT-2-small. In Llama-3.1-8B, stable alignment appears under the larger tested anchor coefficient and in the tested shallow and middle representative layers, while the final layer remains unstable. We anchor only to the input embedding matrix in this work; for untied models, testing anchors based on the

unembedding matrix is left open. Broader anchor-strength sweeps, additional model families, additional model scales, and additional initialization baselines would further test the scope of the geometric alignment result. The reported evidence supports geometric alignment, not causal interpretation of the token names.

References

Belinkov, Y. and Glass, J. R. Analysis Methods in Neural Language Processing: A Survey. *Trans. Assoc. Comput. Linguistics*, 7:49–72, 2019. doi: 10.1162/TACL_A_00254.

Belrose, N., Ostrovsky, I., McKinney, L., Furman, Z., Smith, L., Halawi, D., Biderman, S., and Steinhardt, J. Eliciting Latent Predictions from Transformers with the Tuned Lens, November 2025.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models, 2023.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy Models of Superposition, September 2022.

Ethayarajh, K. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 55–65. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1006.

Fiotto-Kaufman, J. F., Loftus, A. R., Todd, E., Brinkmann, J., Pal, K., Troitskii, D., Ripa, M., Belfki, A., Rager, C., Juang, C., Mueller, A., Marks, S., Sharma, A. S., Lucchetti, F., Prakash, N., Brodley, C. E., Guha, A., Bell, J., Wallace, B. C., and Bau, D. NNsight and NDIF: Democratizing Access to Open-Weight Foundation Model Internals. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 5484–5495. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.446.

Ghandeharioun, A., Caciularu, A., Pearce, A., Dixon, L., and Geva, M. Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models. In Salakhutdinov, R., Kolter, Z., Heller, K. A., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 15466–15490. PMLR / OpenReview.net, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong,

C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao,

Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The Llama 3 Herd of Models, November 2024.

Hewitt, J. and Liang, P. Designing and Interpreting Probes with Control Tasks. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 2733–2743. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1275.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. An empirical analysis of compute-optimal large language model training. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Huben, R., Cunningham, H., Smith, L. R., Ewart, A., and Sharkey, L. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Inan, H., Khosravi, K., and Socher, R. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J. I., Chanin, D., Lau, Y., Farrell, E., McDougall, C., Ayonrinde, K., Till, D., Wearden, M., Conmy, A., Marks, S., and Nanda, N. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K., and Zhu, J. (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/karvonen25a.html.

Makhzani, A. and Frey, B. J. K-Sparse Autoencoders. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Mu, J. and Viswanath, P. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Paulo, G., Mallen, A., Juang, C., and Belrose, N. Automatically Interpreting Millions of Features in Large Language Models. In Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K., and Zhu, J. (eds.), *Forty-Second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

Press, O. and Wolf, L. Using the Output Embedding to Improve Language Models. In Lapata, M., Blunsom, P., and Koller, A. (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pp. 157–163. Association for Computational Linguistics, 2017. doi: 10.18653/V1/E17-2025.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Tenney, I., Das, D., and Pavlick, E. BERT Rediscovers the Classical NLP Pipeline. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4593–4601. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1452.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention

is All you Need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

A. Evaluation Metrics

This appendix collects the evaluation and diagnostic metrics used in the main text and sensitivity-analysis tables.

MSE. Given evaluation post-residual streams $\{\mathbf{h}_i\}_{i=1}^{M}$ and reconstructions $\{\tilde{\mathbf{h}}_i\}_{i=1}^{M}$, where M counts valid token positions after flattening token positions across sequences, we define mean-squared error (MSE) as

$$\mathrm{MSE} = \frac{1}{M d_{\mathrm{model}}} \sum_{i=1}^{M} \left\| \mathbf{h}_i - \tilde{\mathbf{h}}_i \right\|_2^2 . \tag{12}$$

VE. Let

$$\bar{\mathbf{h}} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{h}_i \tag{13}$$

denote the empirical mean target post-residual stream. The target variance is

$$\mathrm{Var} = \frac{1}{M d_{\mathrm{model}}} \sum_{i=1}^{M} \left\| \mathbf{h}_i - \bar{\mathbf{h}} \right\|_2^2 . \tag{14}$$

Using MSE and this variance, VE is

$$\mathrm{VE} = 1 - \frac{\sum_{i=1}^{M} \left\| \mathbf{h}_i - \tilde{\mathbf{h}}_i \right\|_2^2}{\sum_{i=1}^{M} \left\| \mathbf{h}_i - \bar{\mathbf{h}} \right\|_2^2} = 1 - \frac{\mathrm{MSE}}{\mathrm{Var}} . \tag{15}$$

Higher VE means more variance is recovered.

CE loss. For the CE reconstruction check, we evaluate three residual-stream substitutions at the evaluated site: the SAE-style reconstruction $\tilde{\mathbf{h}}$, the original $\mathbf{h}$, and the zero vector $\mathbf{0}$. For substitution $r \in \{\mathrm{SAE}, \mathrm{Id}, 0\}$, let $\mathbf{p}_i^{(r)}$ be the resulting next-token distribution at evaluated position i, and let y_i be the target next token. The average next-token cross-entropy is

$$\mathrm{CE}_r = -\frac{1}{M} \sum_{i=1}^{M} \log \mathbf{p}_i^{(r)}(y_i). \tag{16}$$

Here $r = \mathrm{SAE}$ uses $\tilde{\mathbf{h}}$, $r = \mathrm{Id}$ uses the original $\mathbf{h}$, and $r = 0$ uses $\mathbf{0}$. The CE loss reported in Table 1 is $\mathrm{CE}_{\mathrm{SAE}}$.

CE rec. We compute CE recovery as

$$\mathrm{CERec} = 1 - \frac{\mathrm{CE}_{\mathrm{SAE}} - \mathrm{CE}_{\mathrm{Id}}}{\mathrm{CE}_0 - \mathrm{CE}_{\mathrm{Id}}} . \tag{17}$$

A value of 0 corresponds to the zero-vector control. A value of 1 corresponds to the original residual stream under this normalization. Equivalently, CE recovery measures how much of the CE gap between the zero-vector control and the original residual stream is recovered by the SAE-style reconstruction: values near 1 preserve the original next-token loss, values near 0 match the zero-vector control, and negative values are worse than that control under this normalization.

LogitLens accuracy. As an auxiliary direct-readout diagnostic, we compare the top-1 token obtained from the original post-residual stream with the top-1 token obtained from its reconstruction after projection through the unembedding matrix:

$$\begin{aligned} \mathrm{LogitlensAcc} &= \frac{1}{M} \sum_{i=1}^{M} \mathbf{1}[a_i = \tilde{a}_i] , \\ a_i &= \arg\max_{v \in \mathcal{V}} \mathbf{h}_i \mathbf{u}_v^\top , \\ \tilde{a}_i &= \arg\max_{v \in \mathcal{V}} \tilde{\mathbf{h}}_i \mathbf{u}_v^\top . \end{aligned} \tag{18}$$

Here a_i and $\tilde{a}_i$ are the original and reconstructed top-1 tokens under the direct readout, and $\mathbf{u}_v$ denotes the row of the unembedding matrix $\mathbf{W}_U$ for token v. This metric checks agreement under a direct layerwise readout. It is separate from CE recovery, which substitutes the reconstruction into the model and evaluates the final next-token loss after the remaining computation.

B. Anchor-Strength Sensitivity Analysis

Table 2 reports a representative sweep over λ_{anchor} while keeping the same top-k SAE architecture. The setting $\lambda_{\text{anchor}} = 0$ disables the anchor term and serves as the component ablation.

Changing λ_{anchor} from 0 to 5×10^{-3} changes VE and CE recovery only slightly on these representative GPT-2-small and Llama-3.1-8B layers. This sweep indicates empirical robustness over the tested range; λ_{anchor} remains a tunable hyperparameter in other architectures or layers.

C. Layerwise Diagnostics

The aggregate benchmark in Table 1 reports mean and standard deviation across layers. The layerwise diagnostics in Figure 5 show layer-specific failures that are averaged out in aggregate means. We group the diagnostics by what they

Table 2. Representative anchor-strength sensitivity analysis supporting the larger-coefficient Llama alignment setting. Arrows indicate the preferred direction for each metric. Best values within each model-layer block and metric column are bolded.

GPT-2-small				Llama-3.1-8B			
Layer	λ_{anchor}	VE ↑	CE rec. ↑	Layer	λ_{anchor}	VE ↑	CE rec. ↑
L5	0	**0.9938**	**0.9865**	L15	0	**0.9818**	0.9352
	10^{-4}	**0.9938**	0.9863		10^{-4}	0.9817	0.9351
	5×10^{-3}	0.9937	0.9854		5×10^{-3}	**0.9818**	**0.9357**
L11	0	**0.8155**	0.8906	L31	0	0.6141	0.6303
	10^{-4}	0.8152	**0.8915**		10^{-4}	0.6147	**0.6434**
	5×10^{-3}	0.8149	0.8904		5×10^{-3}	**0.6151**	0.6391

measure: VE measures residual-space reconstruction, CE recovery measures downstream loss after running the remaining model computation, and LogitLens accuracy measures agreement under direct unembedding.

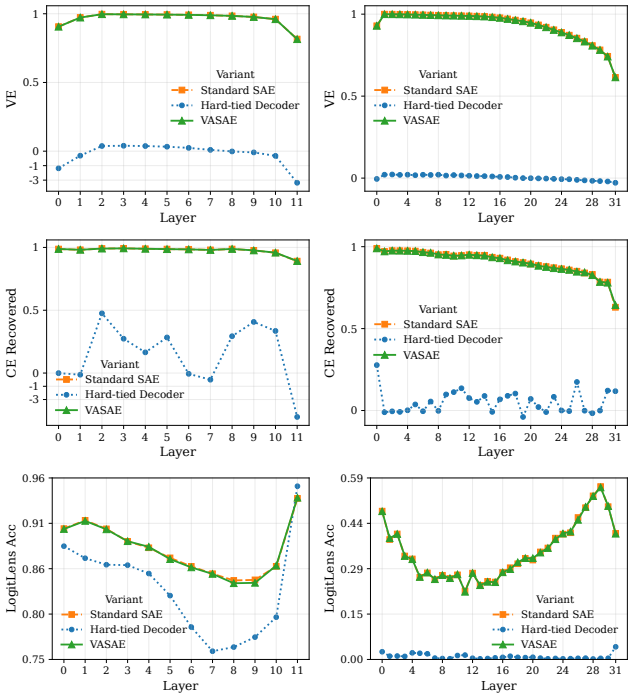

Figure 5. Layerwise benchmark diagnostics for GPT-2-small and Llama-3.1-8B. Top: VE. Middle: CE recovery. Bottom: LogitLens top-1 agreement under direct unembedding.

VASAE and the standard SAE nearly overlap in VE and CE recovery across layers in both model families, matching the aggregate benchmark. This supports reconstruction preservation in the tested cases; trade-offs in other settings remain empirical. The hard-tied decoder baseline is much lower on these two metrics, especially in GPT-2-small late layers and across Llama-3.1-8B. The LogitLens curves show a different direct-readout pattern: GPT-2-small hard-tying keeps higher top-1 agreement in several layers, while its CE recovery remains lower. This indicates that direct unembedding agreement and downstream functional recovery capture different aspects of reconstruction. Llama-3.1-8B has lower LogitLens agreement overall than GPT-2-small under this direct readout.

D. Geometric Alignment Metrics

For each feature i, we use the nearest-token alignment score s_i and nearest vocabulary item $v_i^\star$ defined in Subsection 3.2. Let $\mathcal{A}_\tau = \{i \in \{1, \ldots, d_{\text{sparse}}\} : s_i \geq \tau\}$ denote the strongly aligned feature set for the diagnostic cutoff τ used in Subsection 4.3. Geometric alignment rate is the proportion of features that are aligned with tokens:

$$\text{GeomAlignRate}_\tau = \frac{|\mathcal{A}_\tau|}{d_{\text{sparse}}}. \tag{19}$$

Vocabulary coverage is the proportion of the vocabulary that has at least one feature aligned to it:

$$\text{Coverage}_\tau = \frac{|\{v_i^\star : i \in \mathcal{A}_\tau\}|}{|\mathcal{V}|}. \tag{20}$$

The geometric alignment denominator is the full-learned feature set.

E. Additional Intrinsic-Name Case Studies

These examples extend the main GPT-2-small location case study by applying the same sentence-centered feature-display rule to other prompt types. Figure 6 uses an adjective/adverb sentence and reports intrinsic token names related to feasibility, degree, and evaluation, including *perfect*, *possible*, *entirely*, *suited*, *different*, *enough*, *completely*, *logically*, *elegant*, *frankly*, and *flexible*. Figure 7 covers a named-entity award sentence and a self-introduction sentence. In the Nicole Kidman sentence, reported names include award and social-context tokens such as *accept*, *acceptance*, *awarded*, *winner*, *praise*, *friend*, *friends*, and *members*. In the self-introduction sentence, reported names include personal-reference and fandom-related tokens such as *name*, *am*, *'m*, *fan*, and *fans*. These examples show the same sentence-centered feature-display pattern beyond the location prompt, while remaining qualitative case studies.

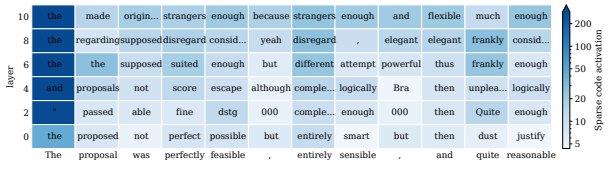

Figure 6. Additional GPT-2-small adjective/adverb example. Each cell shows the intrinsic token name of the feature chosen by the same sentence-centered sparse-code rule as Figure 3. Color indicates the raw sparse-code activation of that feature. Displayed names often track feasibility, degree, and evaluation words in the prompt.

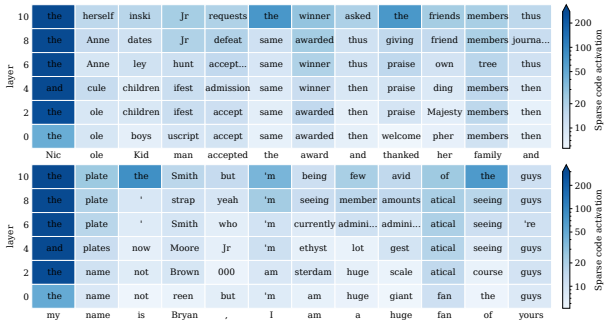

Figure 7. Additional qualitative GPT-2-small intrinsic-name examples. Each cell shows the intrinsic token name of the feature chosen by the same sentence-centered sparse-code rule as Figure 3. Color indicates the raw sparse-code activation of that feature. The examples show displayed names around award-related and self-introduction contexts.

F. Implementation Details for Similarity Score

Computing the similarity scores $\{s_i\}_{i=1}^{d_{\text{sparse}}}$ defined in Equation 7 independently for each feature requires a nested loop over the vocabulary size $|\mathcal{V}|$ and sparse code dimension d_{sparse}. To leverage the highly parallel architecture of modern GPUs, we reframe this computation into dense matrix multiplications.

The cosine similarity between a feature $\mathbf{d}_i$ and a token embedding $\mathbf{w}_v$ is equivalent to the dot product of their L_2-normalized vectors:

$$\cos(\mathbf{d}_i, \mathbf{w}_v) = \left(\frac{\mathbf{d}_i}{\|\mathbf{d}_i\|_2}\right)\left(\frac{\mathbf{w}_v}{\|\mathbf{w}_v\|_2}\right)^\top. \tag{21}$$

We perform row-wise L_2-normalization on $\mathbf{W}_\mathcal{D}^\top$ and $\mathbf{W}_E$ to obtain the normalized feature-by-model matrix $\hat{\mathbf{D}} \in \mathbb{R}^{d_{\text{sparse}} \times d_{\text{model}}}$ and normalized embedding matrix $\hat{\mathbf{W}}_E \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{model}}}$. The full pairwise similarity matrix $\mathbf{S} \in \mathbb{R}^{d_{\text{sparse}} \times |\mathcal{V}|}$ can then be represented via a single matrix multiplication:

$$\mathbf{S} = \hat{\mathbf{D}}\hat{\mathbf{W}}_E^\top, \tag{22}$$

where the element $\mathbf{S}_{i,v} = \cos(\mathbf{d}_i, \mathbf{w}_v)$. The final similarity score s_i for each feature is extracted by taking the maximum value along the rows of $\mathbf{S}$:

$$s_i = \max_{1 \le v \le |\mathcal{V}|} \mathbf{S}_{i,v}. \tag{23}$$

While mathematically straightforward, materializing the dense matrix $\mathbf{S}$ requires $\mathcal{O}(d_{\text{sparse}} \times |\mathcal{V}|)$ memory. For typical SAE dictionary sizes and vocabulary size, this rapidly leads to Out-Of-Memory (OOM) errors.

To resolve this, we implement a chunked computation strategy. We partition $\hat{\mathbf{D}}$ along the row dimension into contiguous blocks of size B. For each block $\hat{\mathbf{D}}_k \in \mathbb{R}^{B \times d_{\text{model}}}$, we compute the local similarity sub-matrix $\mathbf{S}_k = \hat{\mathbf{D}}_k \hat{\mathbf{W}}_E^\top \in \mathbb{R}^{B \times |\mathcal{V}|}$, immediately apply the row-wise maximum reduction to obtain the local scores $\mathbf{s}_k \in \mathbb{R}^B$, and discard $\mathbf{S}_k$. This strategy strictly bounds the peak memory footprint to $\mathcal{O}(B \times |\mathcal{V}|)$.

The PyTorch implementation, mapping directly to our defined matrices $\mathbf{W}_\mathcal{D}$ and $\mathbf{W}_E$, is provided in Listing 1.

Listing 1 Memory-efficient PyTorch implementation for computing Equation 7.

```
def compute_sae_similarity(
    W_dec: torch.Tensor,
    W_E: torch.Tensor,
    chunk_size: int = 2048,
) -> torch.Tensor:
    """Compute nearest-token cosine scores.

    Args:
        W_dec: (d_model, d_sparse) decoder
  ↪  weights.
        W_E: (d_vocab, d_model) token
  ↪  embeddings.
        chunk_size: Number of features per
  ↪  block.

    Returns:
        s: (d_sparse,) max score per
  ↪  feature.
    """

    # Normalize features and token
      ↪  embeddings.
    D_hat = F.normalize(W_dec.T, p=2,
      ↪  dim=1)
    E_hat =
      ↪  F.normalize(W_E.to(D_hat.dtype),
      ↪  p=2, dim=1)

    s_parts = []
    d_sparse = D_hat.size(0)

    # Chunk along features to prevent OOM.
    for i in range(0, d_sparse,
      ↪  chunk_size):
        # S_k shape: (current_chunk_size,
          ↪  d_vocab)
        S_k = D_hat[i : i + chunk_size] @
          ↪  E_hat.T

        # Reduce immediately over tokens.
        s_k, _ = torch.max(S_k, dim=1)
        s_parts.append(s_k)

    # Concatenate chunks into the score
      ↪  vector.
    return torch.cat(s_parts)
```