Disentangling Memorization in Language Models via Sparse Representation Intervention

Anonymous ACL submission

Abstract

Pretrained large language models (LLMs) have become foundational tools in natural language processing (NLP), demonstrating strong performance across tasks such as summarization, question answering, and translation. However, their internal memorization mechanisms remain difficult to interpret and control. This challenge arises from the distributed and non-linear nature of memorization in LLMs, where learned information, such as specific phrases or facts, is entangled across billions of parameters. As a result, identifying how and when memorized content is retrieved during inference remains an open problem. In this work, we propose a novel framework to uncover the relationship between input semantics and memorization in LLMs. We employ a Sparse Autoencoder (SAE) at the final hidden layer to decompose high-dimensional activations into sparse, interpretable components. To further investigate how specific input features influence memorization, we introduce Representation Fine-Tuning (REFT), a mechanism that dynamically edits the SAE-encoded representations based on semantic interventions. Experimental results on the GPT-Neo and Pythia model families show that our method consistently outperforms both state-of-the-art baselines in extracting memorized data. Moreover, we demonstrate that our framework enables fine-grained analysis of how semantic variations in input tokens affect memorization behavior.

1 Introduction

Pretrained large language models (LLMs) (Guimaraes et al., 2024; Wang et al., 2023; Almazrouei et al., 2023; Brown et al., 2020; Gao et al., 2020b) have shown remarkable capabilities across a broad range of natural language processing (NLP) tasks, including summarization (Wei et al., 2021; Zhang et al., 2024; Van Veen et al., 2024), question answering (Zhuang et al., 2023; Li et al., 2024c), and dialogue generation (Liu et al., 2024a; Lu et al., 2025; Das et al., 2024). Beyond generalization, recent studies have revealed that LLMs often memorize and regurgitate training data in many downstream tasks (Firstova et al., 2024; Qin et al., 2024; Carlini et al., 2021). To quantify this behavior, Carlini et al. (2022) introduced the *discoverable memorization rate*, which measures how likely a model is to reproduce memorized sequences. Surprisingly, subsequent works (Chapman et al., 2024; Wang et al., 2024d; Zhao and Patras, 2023) have shown that memorization rates can be unexpectedly high, which can cause severe privacy and copyright issues (Carlini et al., 2021, 2022; Ozdayi et al., 2023; Nasr et al., 2023; Karamolegkou et al., 2023).

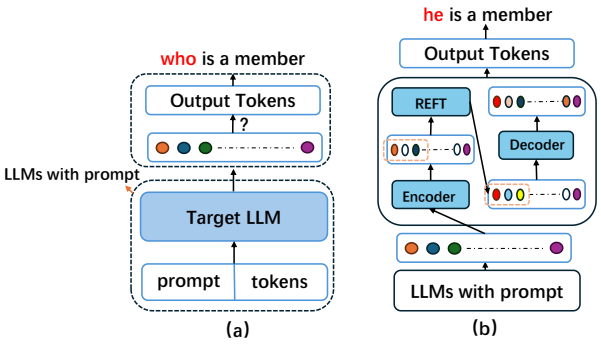


Figure 1: Interpretable and Controllable Memorization in LLMs (a) Standard LLM inference outputs tokens directly based on the prompt (b) Inserts a SAE and applies REFT to decompose and modulate internal representations, enabling controlled and interpretable memory retrieval.

Despite its significance, the mechanisms underlying memorization in LLMs remain poorly understood (De Wynter et al., 2023; Singh et al., 2023). This challenge stems from the mismatch between the non-linear, high-dimensional interactions of billions of model parameters and the need for interpretable, traceable memory representations (Arpit

et al., 2017). During training, LLMs encode data into high-dimensional feature fragments that are stored in a distributed, unstructured manner across the parameter space (Liu et al., 2024c; Aghajanyan et al., 2020). Consequently, pinpointing where specific memories reside and how they are reactivated remains elusive (Khalifa et al., 2024; Chang et al., 2023b). While attention mechanisms offer partial interpretability, they fall short in explaining how memorized content propagates across layers through complex transformations (Jain and Wallace, 2019; Chefer et al., 2021). LLMs often retrieve memorized information in response to semantically distant inputs (Huang et al., 2024; Sun et al., 2025), which is especially problematic in long-form generation (Li et al., 2024b), where irrelevant or weakly related training fragments may unexpectedly appear. Although several methods, such as static and dynamic soft prompts (Chapman et al., 2024; Wang et al., 2024d; Zhao and Patras, 2023), have been proposed to extract memorized content (see Fig. 1(a)), they primarily focus on improving retrieval performance. These methods do not provide a clear understanding of *which semantic features or input variations are critical for triggering memorization.* Thus, a semantic-level explanation of memorization activation remains lacking.

This paper addresses the following central research question: ***What semantic factors cause memorization in LLMs?*** And further: *Can we exploit them to control the memorization rate?* To answer these, we propose a novel framework for analyzing the relationship between input semantics and memorization behavior. As illustrated in Fig. 1(b), we insert a Sparse Autoencoder (SAE) (Ng et al., 2011) into the final hidden layer of an LLM to compress its activations into disentangled components that isolate memorized content. We further introduce Representation Fine-Tuning (REFT) (Wu et al., 2024a), which employs task-specific reference signals to guide semantic reconstruction. By comparing token-level and latent representations before and after intervention, we are able to identify the influence of semantic changes on memorization.

Our contributions are summarized as follows:

- **Locating Activations and Corresponding Semantics Driving Memorization:** We introduce a sparse autoencoder that decomposes hidden states into interpretable components, isolating memorization-related features (e.g., domain-specific terms, syntactic patterns) using ℓ_1-constrained bottlenecks. This enables precise identification of input elements that trigger memorization, improving the interpretability of LLM memory behavior.

- **Controlling Memorization via Representation Modification:** By freezing the LLM's parameters and introducing a lightweight semantic encoder, we edit the internal representations to control memorization behavior. Through pre- and post-intervention analysis, we uncover how subtle semantic changes in input affect memorization.

- **Comprehensive Empirical Evaluation:** We conduct experiments across multiple settings and demonstrate that our framework consistently enhances memory retrieval performance. Compared to baseline methods, our approach achieves relative improvements of up to 65.6% and 29.5% on text generation tasks. Furthermore, REFT-based interventions provide insight into how specific token-level modifications influence memorization, deepening our understanding of memory dynamics in LLMs.

2 Related work

2.1 Memorization in LLM

Early research has shown that LLMs are capable of reproducing exact phrases from their training data when presented with seemingly arbitrary prompts (Madaan et al., 2022; Ippolito et al., 2022). Researchers then defined the concept of discoverable memorization—content that a model can reproduce through a practical search—and proposed the first quantitative framework to assess memorization risk (Schwarzschild et al., 2024; Carlini et al., 2022). Later studies extended this analysis from pre-training to fine-tuning, showing that task-specific fine-tuning not only maintains but also alters memorization patterns, and different retention levels across downstream tasks (VM et al., 2024; Wang et al., 2024b). These findings have important practical implications: proprietary books and source code can be reproduced word-for-word, and production systems like GPT have been shown to leak large amounts of training data (Balloccu et al., 2024; Chang et al., 2023a). To estimate the limit of extractable content, Li et al.

Li et al. (2024a) introduced a constant soft prompt that, once learned, greatly increases data leakage (Hui et al., 2024).

2.2 Sparse Autoencoder

The interpretability of LLMs depends on tracing the pathways responsible for memorization in model outputs, that is identifying which hidden-layer features, attention patterns, or parameter updates directly lead to the repetition of training data (Dang et al., 2024). For example, a model may exactly reproduce training text, such as code snippets or sensitive information, when given certain prompts, or generate content by implicitly retrieving parameterized knowledge, such as factual statements. In this context, the SAE enables interpretability by separating features in a clear and dynamic way (Radhakrishnan et al., 2018). SAEs are added to a layer in LLMs, where they compress and reconstruct hidden states (Ge et al., 2023). This makes the model create sparse representations of hidden features. These representations correspond directly to discrete memorization units (e.g., domain-specific terms, syntactic templates, or memorized segments), creating an explicit mapping between inputs and memorization activations (Cunningham et al., 2023; Seo et al., 2025). Lightweight interventions to the SAE encoder can adjust which features are active. This can reduce unwanted memories or strengthen signals that matter for the task, making it easier to understand how the model remembers information in real time (Kong et al., 2024; Fang et al., 2022).

2.3 REFT Intervention

REFT (Wu et al., 2024a) is an intervention method that adapts pre-trained LLMs to new tasks by updating only a small part of the parameters, while keeping most weights unchanged. Common approaches include Low-Rank Adaptation (LoRA) (Hu et al., 2022), adapter modules (Di Orio, 2013), and prompt tuning (Su et al., 2021).par Recent studies have focused on improving the interpretability and efficiency of LLMs. Wang et al. (Wang et al., 2024a) proposed Low-Rank Adaptation with Gradient Approximation, which aligns the gradients of low-rank updates with those of full fine-tuning at initialization. Bałazy et al. (Bałazy et al., 2024) introduced LoRA-XS, a variant that drastically reduces the number of trainable parameters while maintaining competitive performance. Hu et al. (Hu et al., 2024) analyzed the computational limits of LoRA using fine-grained complexity theory, providing theoretical bounds that guide practical design. However, these methods do not verify how intervention affects the relationship between memorization and semantic factors.

3 Method

A challenge for LLMs is to understand how changes in tokens affect memorization. Existing methods can enhance memorization retrieval but fail to clarify which specific semantic features or tokens drive these changes, or how emphasizing particular words can impact the model's capacity to memorize related content. To address this problem, we introduce a framework that integrates SAE and REFT to facilitate semantic-level intervention and analysis of memorization mechanisms in LLMs. SAE generates sparse, disentangled latent representations that isolate key features related to memorization, while REFT dynamically modulates these latent activations based on targeted semantic cues. This approach allows LLMs to track how focusing on specific semantic units leads to changes in memorization, giving a new insight into how inputs affect what the LLMs remember. Unlike previous work, our method provides a practical way to identify and control the semantic factors that enhance memorization in LLMs.

3.1 Monosemantic Decomposition of Representations

SAEs are unsupervised models that learn important features from input data by making their representations sparse. Based on the standard autoencoder, SAEs reduce the number of activated hidden states to create sparse latent representations. Traditional approaches achieve this sparsity via ℓ_1 regularization or KL divergence penalties (Ng et al., 2011). More recently, Gao et al. (Gao et al., 2024) introduced a TopK sparsity mechanism that explicitly limits the number of active neurons during encoding. Following this strategy, given an input $x \in \mathbb{R}^d$, the function of SAE encoder can be expressed as.

$$h(x) = \mathrm{ReLU}(\mathrm{W}_{\mathrm{enc}} x + \mathrm{b}_{\mathrm{enc}}) \quad (1)$$

where $\mathrm{W}_{\mathrm{enc}} \in \mathbb{R}^{n \cdot d \times d}$ and $\mathrm{b}_{\mathrm{enc}} \in \mathbb{R}^{n \cdot d}$ are the encoder's weight matrix and bias vector, respectively, and $\mathrm{ReLU}(\cdot)$ denotes the rectified linear unit activation. n denotes the expansion rate of the hidden dimension. A TopK operation then enforces hard

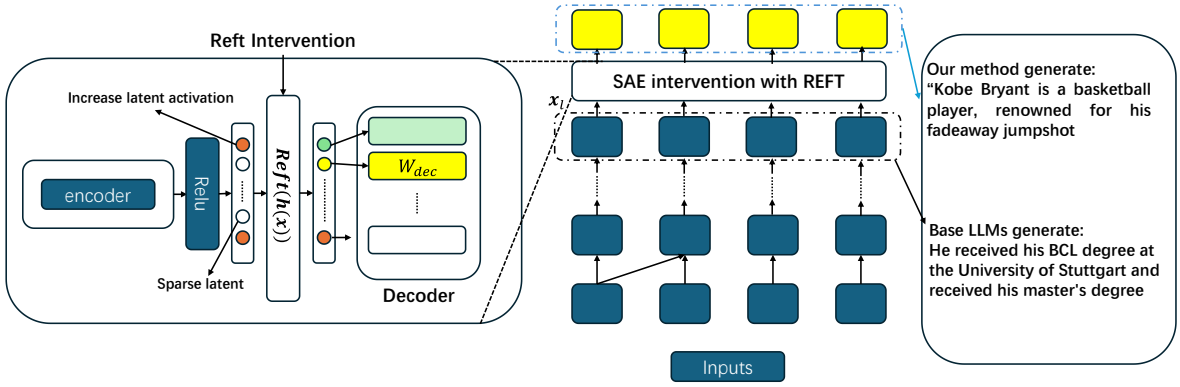


Figure 2: Architecture of Sparse Autoencoder-Controlled LLMs with REFT Intervention

sparsity by preserving only the k largest activations ($k \ll n \cdot d$), which can be expressed as

$$e(x) = \text{TopK}(h(x), k) = \text{P}_k(h(x)) \cdot h(x) \quad (2)$$

which $\text{P}_k(h(x))$ denoted the downsampling matrix. The decoder reconstructs the input $e(x)$ from these sparse activations, which can be expressed as

$$\text{SAE}(x) = \text{W}_{\text{dec}} e(x) + \text{b}_{\text{dec}}, \quad (3)$$

where $\text{W}_{\text{dec}} \in \mathbb{R}^{d \times n \cdot d}$ and $\text{b}_{\text{dec}} \in \mathbb{R}^d$ are the decoder's weight matrix and bias vector. Equation (3) shows that $\text{SAE}(x)$ can be reconstructed by a linear transformation of the encoded representation $e(x)$, using the decoder weights W_{dec} and bias b_{dec}. To encourage sparsity in the learned representations, a sparsity regularization term is added to the training objective. Accordingly, the overall loss function of the SAE can be formulated as

$$L(x) = \frac{1}{2} \|x - \text{SAE}(x)\|_2^2 + \lambda \|h(x)\|_1 \quad (4)$$

The SAE is trained to recover the input as closely as possible after x. Equation (4) simplifies optimization by using a single regularization parameter λ to control sparsity, rather than manually balancing multiple loss terms. By mapping inputs to a high-dimensional space ($n \cdot d$) but allowing only k features to be active, this approach offers three main benefits: (1) it is more robust because of redundant representations, (2) it separates semantic features through sparse competition, and (3) it is more efficient by reconstructing only the active parts.

3.2 Efficient Representation Editing

We use a low-rank geometric method (LoREFT) (Wu et al., 2024b) for the Intervention module, combining low-rank projection and rotation in the latent space. In this way, REFT can dynamically adjust SAE representations through a rotation. Specifically, for an input activation h, the transformation is given by:

$$\text{REFT}(\text{h}) = \text{h} + R^\top(\text{Wh} + \text{b} - R\text{h}) \quad (5)$$

where $\text{W} \in \mathbb{R}^{k \times d}$ denotes a learnable projection matrix encoding target memorization patterns, $R \in \mathbb{R}^{k \times d}$ is a low-rank rotation matrix with $k \ll d$, and $\text{b} \in \mathbb{R}^k$ is a learnable bias term. This transformation first projects the sparse activation h into a memorization-related subspace using $\text{Wh} + \text{b}$, and then measures the difference between this projection and the rotated representation $R\text{h}$. The aligned residual is finally reintegrated back into the original space via the inverse rotation $R^\top$. This process allows the model to inject memorization cues in a geometrically consistent and interpretable manner while preserving the original sparsity structure imposed by the SAE.

The REFT computation proceeds as follows. The learnable projection matrix W serves as a memorization bank, encoding target activation patterns derived from a probing dataset. Given an input representation h, REFT computes the residual vector

$$\Delta\text{h} = \text{Wh} + \text{b} - R\text{h} \quad (6)$$

which captures the discrepancy between the current activation and the desired memorization subspace. To control the correction magnitude, the residual is passed through a $\tanh(\cdot)$ activation. Importantly,

the rotation matrix R is constrained to be orthogonal, ensuring that the adjustment $R^\top \tanh(\Delta h)$ preserves the geometry of the original SAE manifold. This design mitigates interference with the pretrained sparse representations and enables precise, interpretable modulation of memorization-relevant features.

3.3 Interpretable Editing of Model Memorization

Our method uses SAE on the output of the final hidden layer in the target LLMs, enabling precise control over the learned feature representations. As shown in Fig. 2, the base model first generates a dense hidden vector $h_{\text{last}} \in \mathbb{R}^d$ from the input sequence. We then input h_{last} into the SAE encoder, which uses a Topk mechanisms and ReLU activation to encourage sparsity and produces an overcomplete latent vector $a(h_{\text{last}}) \in \mathbb{R}^{n \cdot d}$. Each nonzero activation $a_j(h_{\text{last}})$ in this high-dimensional space represents an approximately monosemantic feature direction, which is stored in the SAE decoder weight matrix $\mathbf{W}_{\text{dec}}$ as shown in Equation 1. We add the REFT intervention module to the SAE encoder to dynamically adjust the sparse activations during inference. REFT works in two stages.

Feature Detection: The module examines the sparse activation vector $a(h_{\text{last}})$ and identifies indices j where the values exceed a set threshold. These indices represent high-confidence feature directions, often associated with memorized content or domain-specific semantic patterns. Specifically, these high-activation features often match clear semantic attributes that are triggered by certain facts or entities in the input.

Actuation: For each selected index j, REFT scales the corresponding sparse activation by a factor α, updating the activation vector as $\tilde{a}_j = \alpha a_j$ while keeping other elements unchanged. The modified sparse vector $\tilde{a}(h_{\text{last}})$ is then passed through the SAE decoder to obtain the new reconstructed hidden state:

$$\hat{h}_{\text{last}}^{\text{new}} = \mathrm{W}_{\text{dec}} \cdot \tilde{a}(h_{\text{last}}) + \mathrm{b}_{\text{dec}} \tag{7}$$

This adjustment directly amplifies ($\alpha > 1$) or suppresses ($\alpha < 1$) the influence of specific features on the model's final output.

The modified hidden state $\hat{h}_{\text{last}}^{\text{new}}$ is sent to the output layer of the target LLM to generate the response. For example, when we prompt the LLM with the basketball player name "Kobe Bryant", LLMs reducing activations related to memorization (such as phrases like "renowned for jumpshot") can produce more general text (like "he received his BCL degree..."). On the other hand, increasing sports-related features can make the output include more details, such as team statistics or career highlights.

In summary, Our framework separates the dense representations in LLMs into a sparse and understandable feature space, where each activation stands for a clear semantic factor. By using the REFT intervention to increase or decrease these sparse features, we can directly find and control which semantic directions—such as factual details, writing style, or domain knowledge—cause performance of memorization changed in LLMs. This not only shows which semantic features cause memorization, but also reveals how LLMs store and express memorized content. It enables clear and targeted interventions, all without retraining the language model.

4 Experiments

4.1 Experiments Setup

Models. First, we evaluated the performance across two pretrained LLMs at three scales: GPT-Neo (125M, 1.3B, 2.7B) (Black et al., 2021) and Pythia (410M, 1.4B, 2.8B) (Biderman et al., 2023), respectively. All models were pretrained on the Pile dataset (Gao et al., 2020a) for text generation. Then we test the different results between the REFT and Manual intervention. Specifically, the changes in the generated text after manually intervening on a certain token and comparing the results of REFT intervention further illustrate What semantic factors cause memorization in LLMs.

Dataset. We extracted inputs using the Language Model Extraction Benchmark (Carlini et al., 2021), selecting 15 K English sequences sampled from the Pile dataset.

Baselines. We compare our method to three baselines: the base LLMs (1) *without any prompt*, (2) *with constant hard prompt*, and (3) *with dynamic hard prompt*, respectively. All methods incorporate an SAE in the final hidden layer. For a prompt length of N, the constant hard prompt uses the first N tokens from the LLM's vocabulary. The dynamic hard prompt employs the mapping $m(\cdot)$ to transform the prefix into a hard prompt. This mapping generates a prompt that is more contextually relevant to the input, aiming to enhance the

Table 1: Main Results on GPT-Neo Models

Model	Method	ExactER	Gain (%)	Frac.ER	Gain (%)	Test Loss	PPL
GPT-125M	No prompt	0.189	N/A	0.369	N/A	0.953	2.594
	Base+SAE	0.065	-65.6	0.238	-35.5	0.179	1.196
	Const HP+SAE	0.062	-67.2	0.226	-38.8	0.181	1.198
	Dyn HP+SAE	0.047	-75.1	0.187	-49.3	0.189	1.208
	Ours (REFT)	**0.314**	+65.7	**0.478**	+29.5	1.097	2.996
GPT-1.3B	No prompt	0.46	N/A	0.643	N/A	0.202	1.224
	Base+SAE	0.183	-60.2	0.371	-42.3	0.088	1.092
	Const HP+SAE	0.179	-61.1	0.363	-43.5	0.091	1.095
	Dyn HP+SAE	0.105	-77.2	0.255	-60.3	0.108	1.114
	Ours (REFT)	**0.547**	+18.9	**0.652**	+1.4	0.621	1.860
GPT-2.7B	No prompt	0.540	N/A	0.702	N/A	0.127	1.135
	Base+SAE	0.328	-39.3	0.511	-27.2	0.131	1.140
	Const HP+SAE	0.313	-42.0	0.486	-30.8	0.183	1.201
	Dyn HP+SAE	0.185	-65.7	0.338	-51.9	0.212	1.236
	Ours (REFT)	**0.612**	+13.3	**0.767**	+9.3	0.556	1.743

Table 2: Main Results on Pythia Models

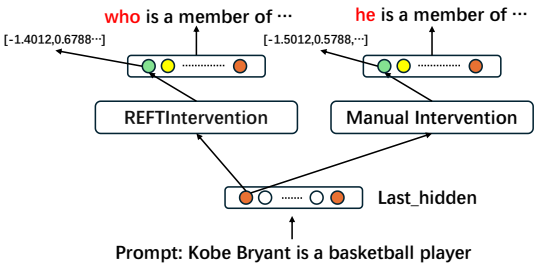
Model	Method	ExactER	Gain (%)	Frac.ER	Gain (%)	Test Loss	PPL
Pythia-410M	No prompt	0.236	N/A	0.458	N/A	0.473	1.605
	Base+SAE	0.180	-23.7	0.367	-19.9	0.905	2.473
	Const HP+SAE	0.080	-66.1	0.210	-54.2	0.720	2.054
	Dyn HP+SAE	0.030	-87.3	0.100	-78.2	1.250	3.490
	Ours (REFT)	**0.430**	+82.2	**0.610**	+33.2	0.380	1.462
Pythia-1.4B	No prompt	0.416	N/A	0.648	N/A	0.199	1.220
	Base+SAE	0.220	-47.1	0.460	-29.0	0.151	1.163
	Const HP+SAE	0.093	-77.6	0.519	-19.9	0.720	2.054
	Dyn HP+SAE	0.051	-87.7	0.578	-10.8	1.250	3.490
	Ours (REFT)	**0.540**	+29.8	**0.702**	+8.3	0.233	1.262
Pythia-2.8B	No prompt	0.517	N/A	0.735	N/A	0.144	1.155
	Base+SAE	0.317	-38.7	0.460	-37.4	0.046	1.047
	Const HP+SAE	0.080	-84.5	0.210	-71.4	0.700	2.014
	Dyn HP+SAE	0.045	-91.3	0.120	-83.7	1.189	3.284
	Ours (REFT)	**0.625**	+20.8	**0.785**	+6.8	0.347	1.415

model's performance without requiring additional training. (Wang et al., 2024d).

4.2 Experiments Result

For text generation tasks, The results show our proposed method has better Exact Extraction Rate (Exact ER) and Fractional Extraction Rate (Frac ER) scores across all model sizes. In the GPT-Neo suite, we achieve relative improvements in Exact ER over the vanilla SAE baseline of 65.7% (125 M), 18.9% (1.3 B), and $+13.3\%$ (2.7 B). The Pythia models exhibit even larger gains of 82.2% (410 M), 29.8% (1.4 B), and 24.8% (2.8 B) in Exact ER compared to other methods. Adding SAEs to different prompt methods will degrade performance due to information loss due to sparsity and reconstruction errors, but our approach improves the results by using REFT interventions in the inference process. Importantly, these improvements do not come from avoiding SAE constraints, but from clearly adjusting how features are encoded using our intervention method. A small increase in test loss was seen during the REFT intervention (for example, GPT-Neo 2.7B: 0.127 to 0.556), but the memorization metrics went up, showing that the model adapted to feature limitations. This suggests that parameter updates first explore the solution space, which can briefly disturb the balance between the SAE's sparsity goals and other regularization targets. The later recovery in memorization shows that our method can improve both memorization and the quality of generated text.

To verify that REFT intervenes in the mem-



(a) Intervention illustration

	GPT_125M	GPT_1.3B	Pythia_410M
REFT	who is a member of the…	renowned for his fade away jump-shot…	who is a member…
Manual	he played for the Lakers, famously rivaling Jordan's Bulls…	renowned for his iconic fade away jumpshot characterized by a 45° body tilt…	who partnered with Nike to release the "Kobe AD" sneaker line in 2017…

(b) Representative outputs under REFT and manual interventions.

Figure 3: Effects of Manual and REFT Interventions on SAE Latent Representations in LLMs.

orization of a certain token, we replaced automatic REFT adjustments with manual tuning of the SAE's latent representation and observed the resulting changes in generated text. Our manual method is to find key latent features in the SAE output—usually those linked to memorized facts and change them in two ways: by increasing important features and reducing low-confidence or irrelevant patterns. For a possible LLM vector $s = [s_1, s_2, ..., s_n]$, we first compute the REFT intervention as an update vector $\text{reft}(s)$ and look at the difference $\Delta s_i = \text{reft}(s_i) - s_i$ We then use these differences to guide manual adjustments. Specifically, we adjust for certain specific markers $\text{manual}(s_i)$ as follows

$$\text{manual}(s_i) = \begin{cases} \text{reft}(s_i) + 0.001, & \Delta s_i > 0 \\ \text{reft}(s_i) - 0.001, & \Delta s_i < 0 \end{cases} \tag{8}$$

then we use $\text{manual}(s_i)$ instead of $\text{reft}(s_i)$ to change the probability of that token and observe how the generated text changes.

As shown in Fig. 3, in GPT-Neo 125M, we manually adjusted the sparse activations linked to specific tokens (e.g., "who" to "it") and saw matching changes in the output text. When we increased the probability of the token "who," the model generated sentences with that word more often; when we reduced it, the word appeared less frequently. This shows that using REFT to adjust specific semantic features can directly affect the model's memory behavior and control how it recalls and generates certain facts.

Fig. 4 shows a case study with the prompt "Kobe Bryant is a basketball player". In this case, different intervention methods changed both the content of the model output and the probabilities of key tokens. When there was no intervention, the result text "He received his BCL degree..." shows the LLMs mainly gave general descriptions about a person, and features like "basketball player" were only weakly activated. After REFT intervention, the probabilities of tokens like "Bulls" and "Basketball" increased, and related information appeared in the output. With manual intervention, features related to Kobe such as "Lakers" were strongly activated, so the generate text more focused on Kobe's information.

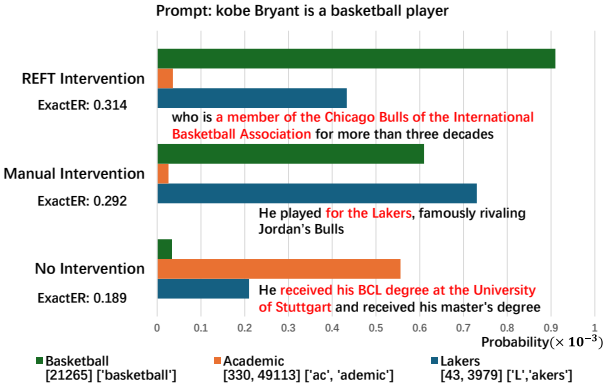


Figure 4: Comparison of token probabilities for selected keywords under no intervention, manual intervention, and REFT intervention.

5 Ablation Study

We test the performance of our proposed method by focusing on three critical hyperparameters the parameter k of Topk, the output dimension of the SAE encoding layer $n \cdot d$, and the REFT rank. And we test performance of different parameters in GPT-Neo (125M).

Impact of Top-k Selection (k). To evaluate the impact of the top-k parameter, we fix the SAE encoding dimension $n = 3$ and the REFT interven-

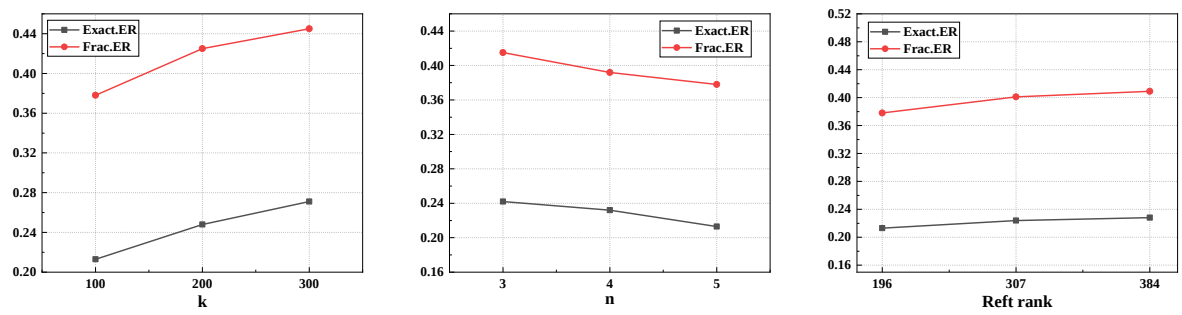


Figure 5: Ablation Study on k, n and reft rank with Exact ER and Frac ER in GPT-Neo (125M)

tion rank. We varied k in the range of 100 to 500 for several representative settings of GPT-Neo (125M, 1.3B, and 2.7B) and Pythia (410M, 1.4B, and 2.8B). Specifically, we set $k = 100$, 200, and 300 for GPT-Neo 125M and Pythia 410M; $k = 200$, 300, and 400 for GPT-Neo 1.3B and Pythia 1.4B; and $k = 300$, 400, and 500 for GPT-Neo 2.7B and Pythia 2.8B, respectively. The results show that our model consistently outperforms the baselines across all tested settings. Notably, increasing k initially leads to substantial performance gains; however, beyond a moderate value (e.g., $k = 300$ in the 410M suite), the improvement plateaus while computational complexity continues to rise. As shown in Fig. 5, GPT-Neo (410M) achieves the best Exact ER and Frac ER when $k = 300$.

Impact of SAE Encoding Dimension ($n \cdot d$). To explore the effect of SAE's encoding dimension, we fix the top-k selection parameter at 100, 200, 300 and REFT rank at 384, 614 and 768 on GPT-Neo (125M, 1.3B, 2.7B), respectively. We vary the SAE encoding dimension from smaller ($n = 3$) to larger dimensions ($n = 5$), respectively. The results show that reducing the SAE encoding dimension can improve performance, reflecting stronger representation ability. However, as the dimension increases, computational cost also grows significantly. Therefore, we chose $n = 3$ as the experimental setting.

Impact of REFT Intervention Rank. Lastly, we vary the REFT rank across different levels, which denoted as L. The experimental results show that higher ranks outperform lower ranks. Specifically, low ranks has low performance to capture relevant variations in latent representations, while overly high ranks introduce complexity without substantial performance gain. Consequently, our findings suggest the optimal balance for the REFT intervention rank is

$$\mathrm{L} = c \cdot d/10 \tag{9}$$

where c represents the coefficients, in our experiments we chose c = 3,4 and 5, corresponding to the reft rank obtained as 196, 307 and 384, respectively.

The experiments showed that both Exact ER and Frac ER slightly increased as the ReFT rank grew, indicating that a higher intervention rank helps the model better activate and extract memory content. However, a too high rank increases computational cost, while performance gains start to level off. Therefore, we choose $\mathrm{L} = 384$ as the experimental setting.

6 Conclusion

To address what semantic factors cause memorization in LLMs, we use a SAE to project the hidden states of LLMs into a sparse space, making it possible to analyze and locate semantic features associated with memorization. By examining the sparse activations, we are able to identify features that are consistently triggered by certain factual prompts or entity names, suggesting their direct involvement in factual recall. To answer whether these factors can be used to control the memorization rate, we apply targeted interventions through REFT, adjusting the influence of specific features on memorization by scaling their activations during inference. Our experiments show that this approach improves LLM performance, clarifies the semantic factors behind memory, and allows effective control of memory behavior. It also makes clear how specific semantic features in the model directly lead to memorization.

Limitations

Although our method improves understanding of memorization in LLMs, its effectiveness is limited by the nature of open-source models. First, biases in open-source training data (like LLaMA and BLOOM) reduce coverage of niche domain concepts, leading to unclear memorization interpretations. Second, the fixed model architectures only allow interventions on shallow activations, not deep memorization changes, which limits meaningful corrections. Lastly, findings from open-source models may not apply to closed-source systems because of unknown differences in data processing and design. While our framework helps extract and analyze memorized information, it could be misused to extract sensitive data or violate privacy if applied to models trained on private datasets. We strongly advise future users to follow ethical guidelines and safeguards to prevent misuse and protect privacy. The code for our SAE and REFT framework will be released on GitHub under the MIT License after publication. All datasets and pretrained models used (such as The Pile, GPT-Neo, Pythia) are open-source and used according to their licenses and research purposes. Our released code is for research use only.

Acknowledgments

References

Divyansh Agarwal, Alexander Richard Fabbri, Ben Risher, Philippe Laban, Shafiq Joty, and Chien-Sheng Wu. 2024. Prompt leakage effect and mitigation strategies for multi-turn llm applications. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1255–1275.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *Preprint*, arXiv:2012.13255.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, and 1 others. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. *Preprint*, arXiv:1706.05394.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. 2021. Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 25–34.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2024. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. *arXiv preprint arXiv:2402.03927*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow.

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.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Kent K Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023a. Speak, memory: An archaeology of books known to chatgpt/gpt-4. *arXiv preprint arXiv:2305.00118*.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2023b. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. *arXiv preprint arXiv:2311.09060*.

Patrick J Chapman, Cindy Rubio-González, and Aditya V Thakur. 2024. Interleaving static analysis and llm prompting. In *Proceedings of the 13th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis*, pages 9–17.

Hila Chefer, Shir Gur, and Lior Wolf. 2021. Transformer interpretability beyond attention visualization. *Preprint*, arXiv:2012.09838.

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*.

Yunkai Dang, Kaichen Huang, Jiahao Huo, Yibo Yan, Sirui Huang, Dongrui Liu, Mengxi Gao, Jie Zhang, Chen Qian, Kun Wang, and 1 others. 2024. Explainable and interpretable multimodal large language models: A comprehensive survey. *arXiv preprint arXiv:2412.02104*.

Trisha Das, Dina Albassam, and Jimeng Sun. 2024. Synthetic patient-physician dialogue generation from clinical notes using llm. *arXiv preprint arXiv:2408.06285*.

Adrian De Wynter, Xun Wang, Alex Sokolov, Qilong Gu, and Si-Qing Chen. 2023. An evaluation on large language model outputs: Discourse and memorization. *Natural Language Processing Journal*, 4:100024.

Giovanni Di Orio. 2013. Adapter module for self-learning production systems. Master's thesis, Universidade NOVA de Lisboa (Portugal).

Haishuo Fang, Ji-Ung Lee, Nafise Sadat Moosavi, and Iryna Gurevych. 2022. Transformers with learnable activation functions. *arXiv preprint arXiv:2208.14111*.

Kristina Firstova, Edward Ramirez, Thomas Castillo, Kenneth Arvidsson, and Anthony Larsen. 2024. Investigating contextual layer fusion in recent open source large language models for context retention and comprehension.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020a. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020b. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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*.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Nuno Guimaraes, Ricardo Campos, and Alípio Jorge. 2024. Pre-trained language models: What do they know? *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(1):e1518.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Jerry Yao-Chieh Hu, Maojiang Su, En-Jui Kuo, Zhao Song, and Han Liu. 2024. Computational limits of low-rank adaptation (lora) for transformer-based models. *arXiv preprint arXiv:2406.03136*.

Baixiang Huang, Canyu Chen, and Kai Shu. 2025. Authorship attribution in the era of llms: Problems, methodologies, and challenges. *ACM SIGKDD Explorations Newsletter*, 26(2):21–43.

Jing Huang, Diyi Yang, and Christopher Potts. 2024. Demystifying verbatim memorization in large language models. *arXiv preprint arXiv:2407.17817*.

Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. 2024. Pleak: Prompt leaking attacks against large language model applications. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3600–3614.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. 2022. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *Preprint*, arXiv:1902.10186.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412.

Muhammad Khalifa, David Wadden, Emma Strubell, Honglak Lee, Lu Wang, Iz Beltagy, and Hao Peng. 2024. Source-aware training enables knowledge attribution in language models. *arXiv preprint arXiv:2404.01019*.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. 2024. Aligning large language models with representation editing: A control perspective. *Advances in Neural Information Processing Systems*, 37:37356–37384.

Diya Li, Asim Kadav, Aijing Gao, Rui Li, and Richard Bourgon. 2024a. Automated clinical data extraction with knowledge conditioned llms. *arXiv preprint arXiv:2406.18027*.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024b. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024c. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18608–18616.

Chenxiao Liu, Zheyong Xie, Sirui Zhao, Jin Zhou, Tong Xu, Minglei Li, and Enhong Chen. 2024a. Speak from heart: an emotion-guided llm-based multimodal method for emotional dialogue generation. In *Proceedings of the 2024 International Conference on Multimedia Retrieval*, pages 533–542.

Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2024b. Large language models can deliver accurate and interpretable time series anomaly detection. *arXiv preprint arXiv:2405.15370*.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, and 2 others. 2024c. Understanding llms: A comprehensive overview from training to inference. *Preprint*, arXiv:2401.02038.

Haitian Lu, Gaofeng Cheng, Liuping Luo, Leying Zhang, Yanmin Qian, and Pengyuan Zhang. 2025. Slide: Integrating speech language model with llm for spontaneous spoken dialogue generation. *arXiv preprint arXiv:2501.00805*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.

Andrew Ng and 1 others. 2011. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

Mustafa Ozdayi, Charith Peris, Jack FitzGerald, Christophe Dupuy, Jimit Majmudar, Haidar Khan, Rahil Parikh, and Rahul Gupta. 2023. Controlling the extraction of memorized data from large language models via prompt-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1512–1521.

A Paszke. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Haotong Qin, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. 2024. Accurate lora-finetuning quantization of llms via information retention. *arXiv preprint arXiv:2402.05445*.

Adityanarayanan Radhakrishnan, Karren Yang, Mikhail Belkin, and Caroline Uhler. 2018. Memorization in overparameterized autoencoders. *arXiv preprint arXiv:1810.10333*.

Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary Lipton, and J Zico Kolter. 2024. Rethinking llm memorization through the lens of adversarial compression. *Advances in Neural Information Processing Systems*, 37:56244–56267.

Seong Hoon Seo, Junghoon Kim, Donghyun Lee, Seonah Yoo, Seokwon Moon, Yeonhong Park, and Jae W Lee. 2025. Facil: Flexible dram address mapping for soc-pim cooperative on-device llm inference. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1720–1733. IEEE.

Chandan Singh, Armin Askari, Rich Caruana, and Jianfeng Gao. 2023. Augmenting interpretable models with large language models during training. *Nature Communications*, 14(1):7913.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, and 1 others. 2021. On transferability of prompt tuning for natural language processing. *arXiv preprint arXiv:2111.06719*.

Chen Sun, Renat Aksitov, Andrey Zhmoginov, Nolan Andrew Miller, Max Vladymyrov, Ulrich Rueckert, Been Kim, and Mark Sandler. 2025. How new data permeates llm knowledge and how to dilute it. *arXiv preprint arXiv:2504.09522*.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerová, and 1 others. 2024. Adapted large language models can outperform medical experts in clinical text summarization. *Nature medicine*, 30(4):1134–1142.

Kushala VM, Harikrishna Warrier, Yogesh Gupta, and 1 others. 2024. Fine tuning llm for enterprise: Practical guidelines and recommendations. *arXiv preprint arXiv:2404.10779*.

Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. 2023. Pre-trained language models and their applications. *Engineering*, 25:51–65.

Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. 2021. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1571–1580.

Shaowen Wang, Linxi Yu, and Jian Li. 2024a. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931.

Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2024b. Generalization vs memorization: Tracing language models' capabilities back to pretraining data. *arXiv preprint arXiv:2407.14985*.

Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. 2024c. Gradient based feature attribution in explainable ai: A technical review. *arXiv preprint arXiv:2403.10415*.

Zhepeng Wang, Runxue Bao, Yawen Wu, Jackson Taylor, Cao Xiao, Feng Zheng, Weiwen Jiang, Shangqian Gao, and Yanfu Zhang. 2024d. Unlocking memorization in large language models with dynamic soft prompting. *arXiv preprint arXiv:2409.13853*.

Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34:16158–16170.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024a. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2024b. ReFT: Representation finetuning for language models.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Zengqun Zhao and Ioannis Patras. 2023. Prompting visual-language models for dynamic facial expression recognition. *arXiv preprint arXiv:2308.13382*.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

A Sparse Autoencoder

The dynamic decoupling and intervention paradigm enables interpretability through features endogenous to the model architecture, eliminating the reliance on post hoc static probing techniques (Wang et al., 2021; Liu et al., 2024b). Broadly, research on memorization interpretability falls into two categories. The first includes a posteriori attribution methods (Hao et al., 2021; Wang et al., 2024c), which rely on correlation-based inference and are vulnerable to confounding effects. The second encompasses static intervention techniques (Bai et al., 2021; Huang et al., 2025), such as soft cue engineering and memorization editing, which, despite enabling targeted suppression of memories, fail to explain the underlying activation pathways (Agarwal et al., 2024).

B Detailed Experimental Setup

Training and Evaluation Settings. The random train/test split is 14k/1k samples, 9686/1k samples, and 38k/2k samples for GPT-Neo and Pythia, respectively. For evaluation, the generation's decoding method is greedy decoding. All the experiments are conducted on a single NVIDIA A6000 GPU with 48GB memorization in less than 12 hours for trianing SAE and 18 hours for training REFT. During the training of the SAE, we used a batch size of 128 and an Adam optimizer for 200 epochs. We tried the learning rate from the range of $[10^{-3}, 10^{-6}]$. During the training of the REFT intervention, we utilize a batch size of 128 for GPT-Neo 125M and 1.3B, as well as Pythia 410M and 1.4B. For GPT-Neo 2.7B and Pythia 2.8B, we use a batch size of 64. All methods employ the AdamW optimizer for 500 epochs, with the learning rate set within the range of $[10^{-3}, 10^{-6}]$. In the ablation study, we reduced the number of epochs from 500 to 200.

Discuss on the Artifacts. The source code of our method is implemented with Pytorch (Paszke, 2019) and HuggingFace Accelerate (Gugger et al., 2022). And the implementation is built upon the open-sourced code released by (Wang et al., 2024d). All the codes and datasets we utilize are public and open-sourced. They support the usage in research

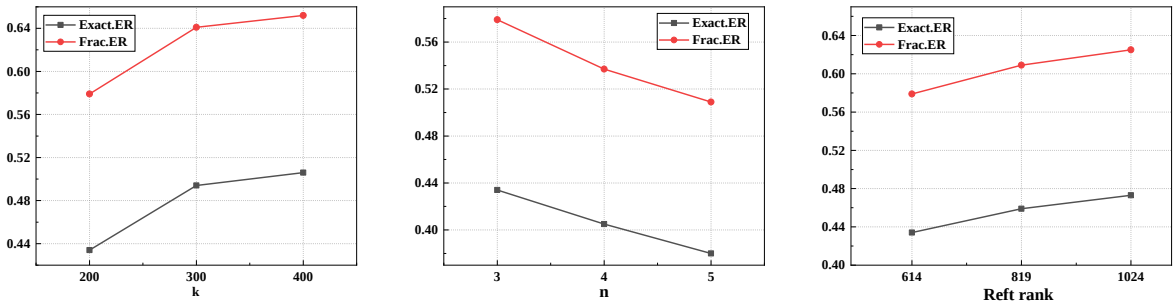


Figure 6: Ablation Study on k, n and reft rank with Exact ER and Frac ER in GPT-Neo (1.3B)

and we use them for research purpose only. We did not check whether the used data contains any information that names or uniquely identifies individual people or offensive content. We left this work to the institution that released the data.

C Ablation Study on Different Model

Fig. 6 shows the ablation study on the parameter k of Topk, expansion number n and reft rank for GPT-Neo (1.3B) in terms of *ExactER* and *Frac ER*. And we can conclude the same conclusion as the ones in Section 5.