# Disentangling Memorization in Language Models via Sparse Representation Intervention

**Anonymous ACL submission**

## Abstract

Pretrained large language models (LLMs) can reproduce near-verbatim sequences from their training data, a phenomenon commonly referred to as memorization. While prior work has studied memorization from data and optimization perspectives, the semantic mechanisms that trigger it remain poorly understood. Because transformer representations integrate contextual information from multiple tokens through self-attention, hidden-layer representations encode the compositional semantics of the input sequence. In this work, we investigate how such semantic information in hidden representations influences memorization behavior. We introduce a framework that decomposes hidden states into sparse semantic components using a Sparse Autoencoder (SAE), where each component captures a latent semantic pattern derived from contextualized token representations. We further apply Representation Fine-Tuning (REFT) to perform targeted interventions on these components, enabling causal analysis of their effects on memorization. Experiments on GPT-Neo and Pythia models show that memorization is driven by a small subset of prompt-related semantic features. By amplifying or suppressing these features, we can systematically modulate the model's memorization rate. Our findings suggest that memorization in LLMs is mediated by identifiable semantic patterns in hidden representations and can be controlled through representation-level interventions.

## 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) encode substantial knowledge from pretraining, enabling users to query information across many academic and professional domains, such as medicine (Yu et al., 2025; Yang et al., 2023), law (Lai et al., 2024; Shu et al., 2024), and finance (Li et al., 2023; Hazourli, 2022). Yet this ability can have a sharper edge: under certain prompts, LLMs may output specific content by reproducing near-verbatim spans from their pretraining data (Firstova et al., 2024; Carlini et al., 2021). This phenomenon, known as *memorization*, suggests that LLM behavior can reflect direct recall of training examples rather than generalization alone (Xiong et al., 2025).

There exist numerous approaches for modulating memorization behavior in LLMs, including data-level interventions such as deduplication and filtering (He et al., 2024; Ramakrishnan and Balaji, 2025), training-time regularization (Slack and Al Moubayed, 2025; Sakarvadia et al., 2024), and prompt or representation-level control methods (Ozdayi et al., 2023). Among these methods, Wang et al. (2024b) proposed a soft-prompt-based approach for estimating memorization in LLMs. However, soft prompt operates at the embedding level, making the resulting memorization behavior difficult to attribute to explicit semantic factors.

This motivates a fundamental question: which semantic components are sufficient to elicit memorization behavior in LLMs? We define semantic components as features in the hidden embedding space that constitute minimal units of meaning, encoding a concept's core attributes and informational content (Cunningham et al., 2023). For example, when an LLM receives the prompt "the meeting was held at the Seabank Centre on Marine Parade ...", the phrases "Seabank Centre" and "Marine Parade" may activate location-related semantic components. As shown in Fig. 1, memorization may be triggered by semantic components related to location, which can steer LLM to respond the text containing specific memory content.

Answering this question we face two major obstacles. (1) There exists a mismatch between the methods for modulating memorization and the mechanistic understanding of memorization capac-

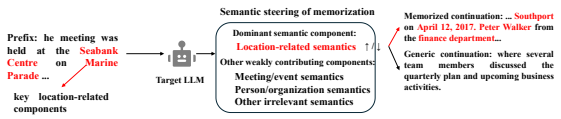


Figure 1: Illustration of semantic-driven memorization in LLMs.

ity. LLMs encode semantic information in high-dimensional distributed representations, so embedding dimensions do not correspond to isolated or human-interpretable concepts. (Liu et al., 2024b; Aghajanyan et al., 2020). Instead, semantic features are distributed across embedding dimensions and contextual token representations (Zhao et al., 2024; Colin et al., 2024), making direct attribution from individual dimensions to meaning challenging.

(2) Different semantic content affects memorization in LLMs differently. Prior work shows that memorization is more likely to be triggered by certain semantic content, especially named entities such as proper nouns, identifiers, or locations (Zhou et al., 2024). This suggests that memorization behavior is triggered by related semantic components. However, existing studies mainly focus on specific semantic units, such as named entities. They do not clearly explain which semantic components within a general prompt drive memorization behavior.

To address this problem, we designed a steer framework for analyzing memorization at the semantic level. Specifically, we use a sparse autoencoder (SAE) (Ng et al., 2011) to disentangle dense semantic representations in the hidden layers of LLMs. This map semantic features into a sparse representation, making it able to identify which semantic components that drive memorization. Subsequently, we employ ReFT (Wu et al., 2024a) to perform automated, controllable interventions on specific semantic features within the encoder layer of SAE. We evaluate memory performance using the memorization rate, which measures a model's tendency to memorize training data by its ability to reproduce suffixes given a prompt (Carlini et al., 2022). Experimental results show that intervening on activation patterns associated with specific semantic components changes their activation strengths, thereby modulating memorization behavior in LLMs. And we

This paper addresses the following central research question: ***Which neuron-level semantic representations significantly affect the model's memorization rate when perturbed?***
And further: ***Can controlled interventions on these semantic representations lead to predictable increases or decreases in the memorization rate, as measured by changes in the model's output probability or recall accuracy?***

Our contributions are summarized as follows:

- **Locating Activations and Corresponding Semantics Trigger Memorization:** In this paper, SAE enforces sparsity on hidden representations, making each sparse component more likely to capture a distinct semantic feature. and the representation easier to interpret. This helps us identify which semantic features, such as domain terms or syntactic patterns, trigger memorization. In this way, we provide a semantic-level explanation of the memory mechanism in LLMs.

- **Modulating Memorization via Representation Editing:** ReFT operates on LLM representations without changing the base model parameters. By introducing a lightweight semantic encoder and editing the identified memory-relevant semantic components, we can directly modulate the model's memorization rate. This shows that the discovered semantic triggers enable fine-grained control over how and when the LLM retrieves memorized content, providing a practical way to manipulate memorization.

- **Enabling Causal Analysis of Memorization:** We validate our framework through extensive experiments in diverse settings. The results show that memorization rate changes with the intervention direction. Increasing the activation of prompt-related semantics strengthens memorization, while decreasing it weakens memorization. This shows a clear causal link between semantic activation and memorization.

## 2 Related work

### 2.1 Memorization in LLMs

Memorization in LLMs refers to the reproduction of exact training sequences (Madaan et al., 2022; Ippolito et al., 2022). A more precise notion is *discoverable memorization*, which denotes content reproducible through a practical search procedure and provides a basis for measuring memorization risk (Carlini et al., 2022; Schwarzschild et al., 2024). This phenomenon has important practical

consequences: proprietary books and source code can be reproduced verbatim, and deployed systems such as GPT can leak substantial amounts of training data (Balloccu et al., 2024; Chang et al., 2023). These findings suggest that memorization is not merely a byproduct of scale, but can also be triggered by specific input patterns or semantic cues.

### 2.2 Interpretable semantic–neuron pathways for memorization

Interpreting memorization in LLMs requires tracing the pathways that lead to training data repetition in model outputs, including hidden-layer features, attention patterns, and parameter updates (Dang et al., 2024). SAEs can be inserted into LLM layers to compress and reconstruct hidden states (Ge et al., 2023), yielding sparse hidden features that can align with discrete memorization units, such as domain-specific terms, syntactic templates, or memorized segments (Cunningham et al., 2023; Seo et al., 2025).

### 2.3 Semantic Representation Interventions

ReFT (Wu et al., 2024a) is a representation-level intervention method that adapts pre-trained LLMs by modifying only a small subspace of hidden representations while keeping the base model parameters unchanged. This property makes it well suited to our goal of identifying which semantic components are sufficient to elicit memorization in LLMs. By intervening on selected semantic components, ReFT enables controlled testing of their causal effects on memorization behavior.

## 3 Method

### 3.1 Problem Formulation

Given a target LLM $f_\theta$ and an input prompt $x = (x_1, x_2, \ldots, x_n)$, we study how semantic features influence memorization. We view semantic features as latent components encoded in the hidden representations of the LLM (Dai et al., 2022). Let $\mathbf{h}^{(l)}(x) = (h_1^{(l)}(x), \ldots, h_d^{(l)}(x)) \in \mathbb{R}^d$ denote the hidden representation of the prompt $x$ at layer $l$. We define the activation of the $j$-th semantic component at layer $l$ as

$$z_j^{(l)}(x) = \phi_j\left(\sum_{m=1}^{d} w_{jm}^{(l)} h_m^{(l)}(x)\right), \qquad (1)$$

where $w_{jm}^{(l)}$ is a parameter of the mapping from $\mathbf{h}^{(l)}(x)$ to the $j$-th semantic component, and $\phi_j$ is a nonlinear activation function.

We intervene on the semantic component $z_j^{(l)}(x)$ by replacing its activation with $z_j^{(l)\prime}(x)$. For a knowledge item $k$, the intervention effect on memorization under input $x$ is defined as

$$\Delta_j(k \mid x) = P_\theta\Big(k \mid x;\, z_j^{(l)}(x) \to z_j^{(l)\prime}(x)\Big) - P_\theta(k \mid x)\,, \qquad (2)$$

where $P_\theta(k \mid x)$ denotes the probability that the model reproduces the target knowledge item $k$ given the prompt $x$.

Following Hayes et al. (2025), memorization can be quantified as the probability that the model produces $k$ in response to $x$. Therefore, the change in memorization induced by intervening on semantic component $z_j^{(l)}(x)$ can be written as

$$\Delta\mathrm{Mem}(k \mid x) \approx \Delta_j(k \mid x). \qquad (3)$$

This formulation allows us to study how specific semantic features of the input influence the model's tendency to reproduce memorized content.

### 3.2 Monosemantic Decomposition of Representations

Based on Equation. 1, decomposing entangled hidden representations into sparse semantic components is essential for analyzing how specific semantic components trigger memorization behavior in LLMs. We therefore apply a Sparse Autoencoder (SAE) to the hidden representation $\mathbf{h}^{(l)}(x)$. By imposing sparsity on the latent representation, SAE ensures that only a limited number of components are activated for a given input, thereby promoting the separation and decoupling of semantic information (Cunningham et al., 2023). This mechanism enables entangled semantic representations to be decomposed into more localized and interpretable latent components.

We adopt the TopK sparsity mechanism (Gao et al., 2024) to map $\mathbf{h}^{(l)}(x)$ into a sparse latent representation, explicitly restricting the number of active components during encoding. The SAE encoder is defined as

$$\mathbf{z}^{(l)}(x) = \mathrm{TopK}\Big(\mathrm{ReLU}\Big(\mathrm{W}_{\mathrm{enc}}\,\mathbf{h}^{(l)}(x) + \mathbf{b}_{\mathrm{enc}}\Big)\Big)\,, \qquad (4)$$

where $\mathrm{W}_{\mathrm{enc}} \in \mathbb{R}^{n\cdot d\times d}$ and $\mathbf{b}_{\mathrm{enc}} \in \mathbb{R}^{n\cdot d}$ are the weight matrix of encoder layer and bias vector, respectively. $\mathrm{ReLU}(\cdot)$ denotes the rectified linear

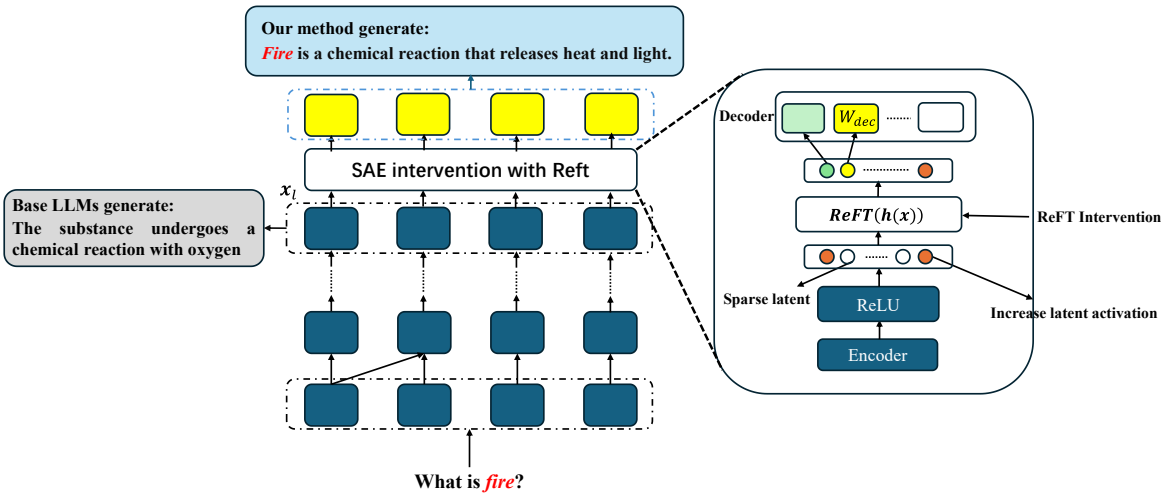


Figure 2: Architecture of SAE-Controlled Semantic Intervention with ReFT.

unit activation function, and $n$ is the expansion factor of the hidden dimension. A TopK operation then enforces sparsity by preserving only the top-$K$ activations, where $K \ll n \cdot d$, which can be expressed as

$$\mathrm{TopK}\left(\tilde{\mathbf{z}}^{(l)}(x), K\right) = \mathbf{P}_K\left(\tilde{\mathbf{z}}^{(l)}(x)\right) \odot \tilde{\mathbf{z}}^{(l)}(x), \tag{5}$$

where $\tilde{\mathbf{z}}^{(l)}(x) = \mathrm{ReLU}\left(\mathrm{W}_{\mathrm{enc}}\,\mathbf{h}^{(l)}(x) + \mathbf{b}_{\mathrm{enc}}\right)$, and $\mathbf{P}_K(\tilde{\mathbf{z}}^{(l)}(x))$ denotes a binary mask vector whose entries are 1 at the positions of the top-$K$ largest activations and 0 elsewhere. The decoder reconstructs the hidden representation $\mathbf{h}^{(l)}(x)$ from the sparse latent representation $\mathbf{z}^{(l)}(x)$, it can be expressed as

$$\widehat{\mathbf{h}}^{(l)}(x) = \mathrm{W}_{\mathrm{dec}}\,\mathbf{z}^{(l)}(x) + \mathbf{b}_{\mathrm{dec}}, \tag{6}$$

where $\mathrm{W}_{\mathrm{dec}} \in \mathbb{R}^{d \times nd}$ and $\mathbf{b}_{\mathrm{dec}} \in \mathbb{R}^d$ are the decoder weight matrix and bias vector, respectively. Equation 6 shows that the reconstructed hidden representation $\widehat{\mathbf{h}}^{(l)}(x)$ is obtained by a linear transformation of the sparse latent representation $\mathbf{z}^{(l)}(x)$. The overall training objective of the SAE is to minimize the reconstruction error of the hidden representation:

$$\mathcal{L}_{\mathrm{SAE}}(x) = \frac{1}{2}\left\|\mathbf{h}^{(l)}(x) - \widehat{\mathbf{h}}^{(l)}(x)\right\|_2^2, \tag{7}$$

where $\widehat{\mathbf{h}}^{(l)}(x)$ is the reconstruction produced by the decoder in Eq. 6.

The SAE is trained to reconstruct the hidden representation $\mathbf{h}^{(l)}(x)$ as accurately as possible. By mapping $\mathbf{h}^{(l)}(x)$ into an expanded latent space of dimension $n \cdot d$ while allowing only $K$ active components, the SAE induces sparse competition among latent units. This encourages different components to capture more distinct semantic patterns, thereby facilitating the identification of specific semantic components that influence memorization.

### 3.3 Efficient Representation Editing

Although semantic representations can be decomposed into more disentangled components, it remains unclear which semantic features in LLMs influence memorization. we therefore employ a low-rank geometric intervention module (LoREFT) (Wu et al., 2024b) applied at the SAE encoder layer, enabling controlled manipulation of related semantics to identify which semantic can trigger memorization.

Given the sparse latent representation $\mathbf{z}^{(l)}(x)$ produced by the SAE encoder, the intervention module transforms it in three steps. First, $\mathbf{z}^{(l)}(x)$ is projected into a low-dimensional subspace using a learned linear transformation parameterized by $\mathbf{W}$ and $\mathbf{b}$. This subspace is optimized to capture directions relevant to memorization. Second, a low-rank projection $\mathbf{R}\mathbf{z}^{(l)}(x)$ of the current latent representation is computed, where $\mathbf{R}$ is a low-rank projection matrix. The difference between the transformed target representation and the current projection is then used to construct an intervention residual:

$$\Delta\mathbf{z}^{(l)}(x) = \mathbf{W}\mathbf{z}^{(l)}(x) + \mathbf{b} - \mathbf{R}\mathbf{z}^{(l)}(x). \tag{8}$$

Finally, this residual is mapped back to the original latent space through $\mathbf{R}^\top$ and added to $\mathbf{z}^{(l)}(x)$,

yielding the intervened representation

$$\mathbf{z}^{(l)\prime}(x) = \mathbf{z}^{(l)}(x) + \mathbf{R}^{\top}\big(\mathbf{W}\mathbf{z}^{(l)}(x) + \mathbf{b} - \mathbf{R}\mathbf{z}^{(l)}(x)\big). \tag{9}$$

The orthogonality constraint on $\mathbf{R}$ helps preserve the original sparse geometry, so that most semantic components remain unchanged. As a result, ReFT enables controlled analysis of how targeted semantic directions in $\mathbf{z}^{(l)}(x)$ affect memorization behavior.

### 3.4 Interpretable Editing of Model Memorization

As shown in Fig. 2, we add SAE into the final hidden layer in the LLMs to distinguish semantics. When prompted with "The capital of Canada is," the LLMs often encodes multiple intertwined semantic features, such as "capital," "Canada," and "Toronto," within the embedding dimensions. This makes human difficult to determine why the LLMs produces an incorrect answer like "Toronto". After introducing SAE, the hidden representation is decomposed into sparse semantic units, each more likely to correspond to a single attribute. As a result, when LLMs answer "Toronto" rather than " Ottawa", we can observe that the unit associated with "Toronto" is highly activated, and the unit representing the "capital" attribute remains underactivated. The details are shown in Fig. 3. Next, we

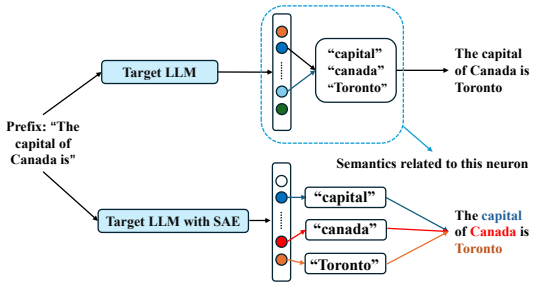


Figure 3: Semantic disentanglement enabled by SAE for interpreting memorization.

apply ReFT at the SAE encoder layer to dynamically intervene on selected semantic features during inference. Specifically, ReFT enables neuron-level interventions, allowing us to identify the semantic features associated with neurons that drive memorization. To further validate the causal role of the semantic features identified by ReFT, we perform additional manual interventions within the same prompt. Specifically, when ReFT enhances the unit associated with a particular semantic feature (e.g., "wager"), we manually suppress that unit while enhancing another semantic feature known to facilitate memorization. We then examine whether the model's output shifts toward the newly emphasized semantic feature.

Formally, let $s = [s_1, s_2, \ldots, s_n]$ denote the sparse activation vector produced by SAE, and let $\mathrm{reft}(s)$ represent the REFT-adjusted activations. For each feature $n$, we compute the difference $\Delta s_i = \mathrm{reft}(s_n) - s_n$. Manual intervention is then defined as a minimal step beyond REFT scaling:

$$\tilde{s}_k = \begin{cases} s_k - \alpha\,\Delta s_k, & \text{if } k = i, \\ s_k + \beta\,\Delta s_k, & \text{if } k = j, \\ s_k, & \text{otherwise.} \end{cases} \tag{10}$$

Then we know which semantic can trigger memorization behavior in LLMs.

## 4 Experiments

### 4.1 Experiments Setup

**Models.** In this section, the proposed method are used to two pretrained LLMs across three different 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.

**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 proposed method against three prompt-based methods (1) *without any prompt*, (2) *with constant hard prompt*, and (3) *with dynamic hard prompt* to explain the relationship between the semantics and LLM memorization mechanisms. For a prompt length of $N$, the constant hard prompt uses the first $N$ tokens from the LLM's vocabulary, while the dynamic hard prompt employs a mapping $m(\cdot)$ to transform the prompt into a hard prompt that is more contextually relevant to the input (Wang et al., 2024b). These prompt-based methods are primarily designed to improve model performance by altering the semantic content of input tokens, but they can only provide an indirect view of how semantics affect memorization. In contrast, our approach combines SAE disentangled sparse representations with ReFT causal interventions to directly identify and manipulate the semantic factors that trigger

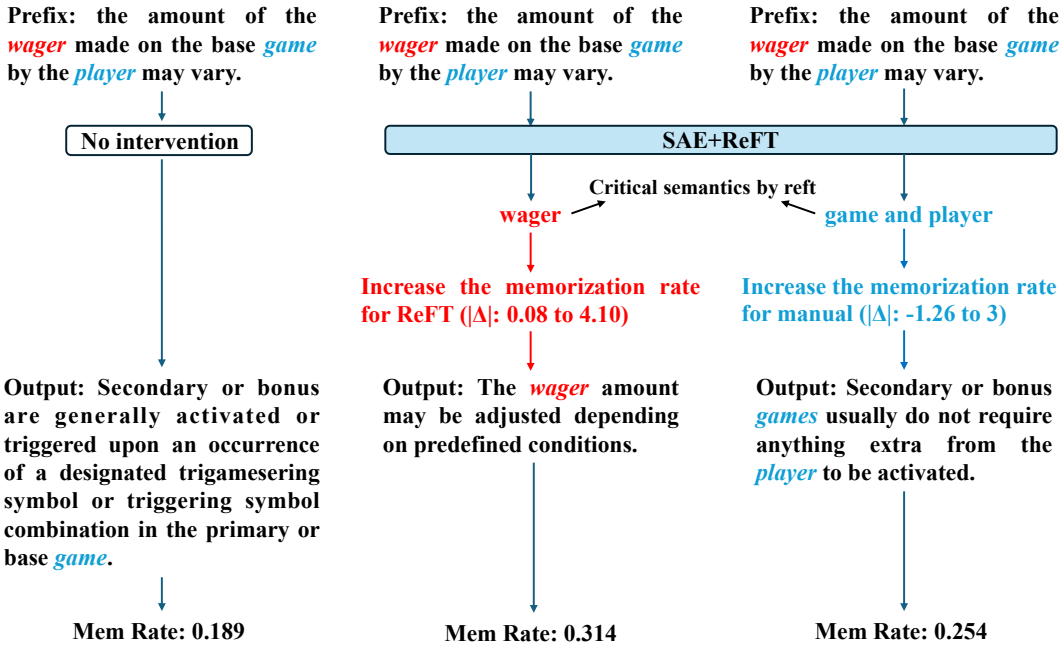


Figure 4: Comparison of token probabilities for selected keywords under no intervention, manual intervention, and REFT intervention. The colored tokens are identified by our method as carrying critical semantics in memorization

memorization. This enables us not only to measure memorization rate changes but also to explain why and how memorization occurs, and understanding of the underlying mechanisms of memorization. We use the Exact Extraction Rate (ER), Fractional Extraction Rate (FR) (Wang et al., 2024b), Test loss and Test perplexity (PPL) to evaluate the performance of our method.

## 4.2 Experiments Result

First, we show that memorization in LLMs is interpretable and can be attributed to semantic features with different contributions. As shown in Fig. 4, when prompt is "the amount of the wager made on the base game by the player may vary", our method identifies a set of latent semantic features associated with the tokens *wager* and *game* after training. These features correspond to dimensions in the sparse space that encode specific semantic meanings, such as betting-related or game-related concepts. Further, we suppress or amplify those selected semantics in prompts, respectively. When semantic features that are strongly associated with memorization (e.g., the *wager*-related feature) are selectively amplified, the model prioritizes retrieves memorized content and generates continuations consistent with gambling-specific interpretations. In contrast, suppressing these semantic features shifts the generation toward more generic responses, accompanied by a corresponding reduction in the memorization rate. This visualization provides direct evidence that modifying a small set of interpretable semantics is sufficient to causally alter the memorization behavior.

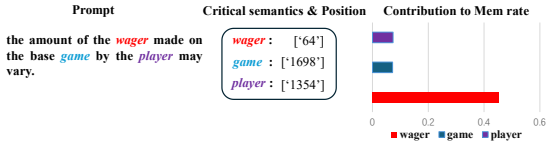


Figure 5: Semantic attribution of memorization for a single prompt.

Then, we use the same prompt to identify that memorization in LLMs is interpretable and can be attributed to semantic features with different contributions. As shown in Fig. 5, when prompt is "the amount of the wager made on the base game by the player may vary", the tokens of “wager”, “game”, and “player” are identified as key semantic components in the prompt after training. Specifically, “wager” accounts for most of the memorization effect, while “game” and “player” contribute less. This indicates the memorization behavior is mainly modulated by changes in a part of semantic features, and their effects are highly uneven: manipulating semantics related to ‘wager’ causes the largest change in memorization rate, while inter-

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 |
| | Dyn HP | 0.421 | +22.75 | 0.557 | +50.89 | 0.665 | 1.945 |
| | 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)− | **0.105** | -44.4 | **0.278** | -24.6 | 1.103 | 3.013 |
| | 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 |
| | Dyn HP | 0.651 | +41.52 | 0.772 | +20.04 | 0.114 | 1.121 |
| | 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)− | **0.256** | -44.35 | **0.402** | -37.48 | 0.234 | 1.264 |
| | Ours (REFT)+ | **0.522** | +13.48 | **0.672** | +4.51 | 0.221 | 1.247 |
| **GPT-2.7B** | No prompt | 0.540 | N/A | 0.702 | N/A | 0.127 | 1.135 |
| | Dyn HP | 0.702 | +30.0 | 0.82 | +16.83 | 0.075 | 1.078 |
| | 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)− | **0.332** | -38.52 | **0.47** | -33.05 | 0.111 | 1.117 |
| | Ours (REFT)+ | **0.612** | +13.3 | **0.767** | +9.3 | 0.106 | 1.112 |

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 |
| | Dyn HP | 0.513 | 117.37 | 0.683 | 49.09 | 0.283 | 1.328 |
| | 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)+ | **0.205** | -13.1 | **0.432** | -5.7 | 0.392 | 1.48 |
| | 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 |
| | Dyn HP | 0.617 | 48.32 | 0.786 | 21.36 | 0.109 | 1.115 |
| | 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)− | **0.278** | -33.17 | **0.546** | -15.74 | 0.229 | 1.257 |
| | 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 |
| | Dyn HP | 0.669 | 29.4 | 0.827 | 12.57 | 0.08 | 1.084 |
| | 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)− | **0.471** | -8.9 | **0.697** | -5.17 | 0.344 | 1.411 |
| | Ours (REFT)+ | **0.625** | +20.8 | **0.785** | +6.8 | 0.347 | 1.415 |

ventions on ‘game’ and ‘player’ produce smaller effects. More results are shown in 5.

In the text generation task, Tables 1 and 2 show that different intervention directions can affect the ER and the FR. In the GPT-Neo suite, positive intervention improves Exact ER by 65.7% (125M), 13.48% (1.3B), and 13.3% (2.7B) over the baseline, while negative intervention reduces Exact ER by 44.4%, 44.35%, and 38.52%, respectively. The Pythia models show the same trend: positive intervention yields Exact ER gains of 82.2% (410M), 29.8% (1.4B), and 20.8% (2.8B), whereas negative intervention leads to corresponding decreases of 13.1%, 33.17%, and 8.9%. Together, these results show that memorization in LLMs can be controlled by regulating the activation of specific semantic components.

| Metric | Baseline | SAE+ReFT |
|---|---|---|
| ROUGE-1 | 13.80 | 16.09 |
| ROUGE-2 | 4.77 | 8.21 |
| ROUGE-L | 10.38 | 11.34 |

Table 3: Performance comparison on 1,000 samples from the CNN/Daily Mail dataset using GPT-Neo 125M.

Since SAE reconstruction only approximates the input representation, its insertion inevitably causes some fidelity loss. Although the test loss of our method are increased slightly, our method consistently outperforms the baseline on ROUGE-1, ROUGE-2, and ROUGE-L on CNN/Daily Mail dataset, respectively. The result are shown in Tables 3. This indicates that the sparsity constraint introduced by SAE may cause a trade-off with the model’s original optimization objective, while still leading to better downstream generation performance.

### 4.3 Hyperparameter 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),the numerical result as shown in Fig. 6.

**Impact of Top-k Selection ($k$).** We study the Top-$k$ parameter by fixing the SAE encoding dimension ($n = 3$) and the ReFT rank. We vary $k$ across GPT-Neo and Pythia models, using larger values for larger models. Increasing $k$ initially improves Exact ER and Frac ER, but gains saturate beyond moderate values while computation increases. GPT-Neo (125M) performs best at $k = 300$. We therefore adopt this setting as a practical trade-off between performance and efficiency.

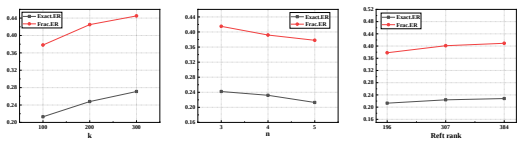


Figure 6: Effect of hyperparameter on memorization. Exact ER and Frac ER consistently improve as Top-$k$, SAE encoding dimension, and REFT rank increase, but plateau beyond moderate values.

**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 improves performance, indicating stronger representational capacity. However, larger dimensions also incur substantially higher computational cost. Therefore, we choose $n = 3$ in our experiments.

**Impact of REFT Intervention Rank.** Lastly, we vary the ReFT rank L and observe that higher ranks generally perform better. Low ranks fail to capture sufficient latent variation, while very high ranks increase computation with limited gains. We find a good trade-off at $\mathrm{L} = c \cdot d/10$ with $c \in 3, 4, 5$ (i.e., 196, 307, and 384). Both Exact ER and Frac ER improve slightly with rank but quickly saturate.

## 5 Conclusion

In this paper, we study which semantic features trigger memorization in LLMs and how they influence memorization behavior. Experiment results shows that memorization is not driven uniformly by the prompt, but is closely tied to a small set of semantic factors, often associated with factual cues or entity names. By identifying and adjusting these factors during inference, we show that memorization can be modulated in a fine-grained manner. Overall, our findings provide a more interpretable view of memorization and offer a practical direction for understanding and controlling it.

## Limitations

Although we 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*.

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.

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. 2023. Speak, memory: An archaeology of books known to chatgpt/gpt-4. *arXiv preprint arXiv:2305.00118*.

Julien Colin, Lore Goetschalckx, Thomas Fel, Victor Boutin, Jay Gopal, Thomas Serre, and Nuria Oliver. 2024. Local vs distributed representations: What is the right basis for interpretability? *arXiv preprint arXiv:2411.03993*.

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*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

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*.

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.

Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Ilia Shumailov, and Milad Nasr. 2025. Measuring memorization through probabilistic discoverable extraction. *NAACL*. ArXiv preprint arXiv:2410.19482.

Ahmed Hazourli. 2022. Financialbert-a pretrained language model for financial text mining. *Research Gate*, 2.

Nan He, Weichen Xiong, Hanwen Liu, Yi Liao, Lei Ding, Kai Zhang, Guohua Tang, Xiao Han, and Yang Wei. 2024. Softdedup: an efficient data reweighting method for speeding up language model pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4011–4022.

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.

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*.

Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S Yu. 2024. Large language models in law: A survey. *AI Open*, 5:181–196.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Jun Liu, Chaoyun Zhang, Jiaxu Qian, Minghua Ma, Si Qin, Chetan Bansal, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2024a. 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. 2024b. Understanding llms: A comprehensive overview from training to inference. *Preprint*, arXiv:2401.02038.

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*.

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*.

Badrinath Ramakrishnan and Akshaya Balaji. 2025. Assessing and mitigating data memorization risks in fine-tuned large language models. *arXiv preprint arXiv:2508.14062*.

Mansi Sakarvadia, Aswathy Ajith, Arham Khan, Nathaniel Hudson, Caleb Geniesse, Kyle Chard, Yaoqing Yang, Ian Foster, and Michael W Mahoney.

2024. Mitigating memorization in language models. *arXiv preprint arXiv:2410.02159*.

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.

Dong Shu, Haoran Zhao, Xukun Liu, David Demeter, Mengnan Du, and Yongfeng Zhang. 2024. Lawllm: Law large language model for the us legal system. In *Proceedings of the 33rd ACM International Conference on information and knowledge management*, pages 4882–4889.

Dean L Slack and Noura Al Moubayed. 2025. Early detection and reduction of memorization for domain adaptation and instruction tuning. *Transactions of the Association for Computational Linguistics*, 13:1459–1473.

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.

Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. 2024a. 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. 2024b. Unlocking memorization in large language models with dynamic soft prompting. *arXiv preprint arXiv:2409.13853*.

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.

Alexander Xiong, Xuandong Zhao, Aneesh Pappu, and Dawn Song. 2025. The landscape of memorization in llms: Mechanisms, measurement, and mitigation. *arXiv preprint arXiv:2507.05578*.

Rui Yang, Ting Fang Tan, Wei Lu, Arun James Thirunavukarasu, Daniel Shu Wei Ting, and Nan Liu. 2023. Large language models in health care: Development, applications, and challenges. *Health Care Science*, 2(4):255–263.

Erlan Yu, Xuehong Chu, Wanwan Zhang, Xiangbin Meng, Yaodong Yang, Xunming Ji, and Chuanjie Wu. 2025. Large language models in medicine: Applications, challenges, and future directions. *International Journal of Medical Sciences*, 22(11):2792.

Haiyan Zhao, Fan Yang, Bo Shen, Himabindu Lakkaraju, and Mengnan Du. 2024. Towards uncovering how large language model works: An explainability perspective. *arXiv preprint arXiv:2402.10688*.

Zhenhong Zhou, Jiuyang Xiang, Chaomeng Chen, and Sen Su. 2024. Quantifying and analyzing entity-level memorization in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19741–19749.

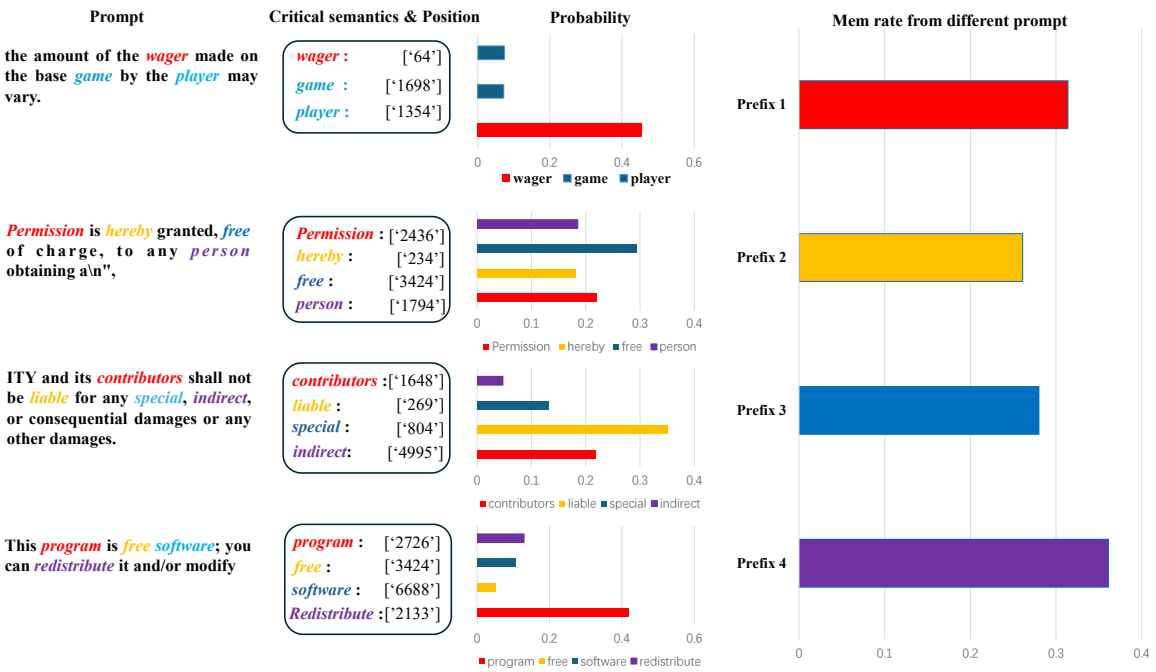


Figure 7: Semantic attribution of memorization in LLMs. Memorization is dominated by a small number of prompt-related semantic features, whose unequal contributions explain variations in memorization rates across different prompts.

## A Experiments result

As shown in Fig. 7, we provide additional examples to demonstrate that memorization is interpretable by establishing a clear connection between semantic features and memorization behavior. For each input prompt, we identify critical semantic tokens and their corresponding positions in the SAE latent space, and quantify their individual contributions to the overall memorization rate. The bar charts illustrate how different semantic components contribute unevenly to memorization, indicating that memorized content is driven by a small subset of prompt-related semantics rather than uniformly distributed across tokens. The right panel compares the overall memorization rates across different prompts, further demonstrating that variations in memorization behavior can be explained by differences in the activated semantic features and their relative contributions.

## B Ablation Study on Different Model

Fig. 8 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 4.3.

## C 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., 2024a). Broadly, research on memorization interpretability falls into two categories. The first includes a posteriori attribution methods (Hao et al., 2021; Wang et al., 2024a), 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).

## D 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

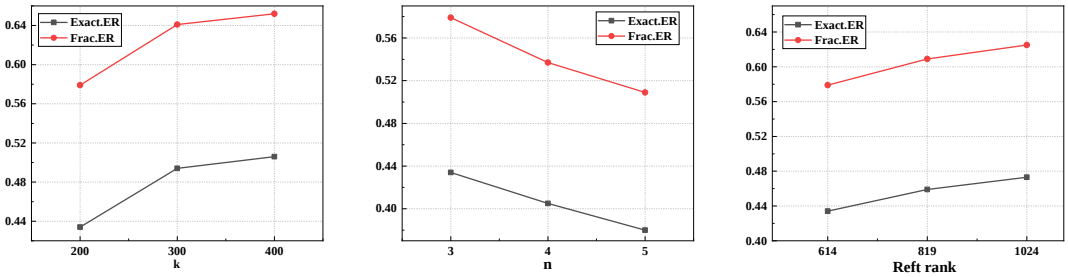


Figure 8: Ablation Study on k, n and reft rank with Exact ER and Frac ER in GPT-Neo (1.3B)

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., 2024b). All the codes and datasets we utilize are public and open-sourced. They support the usage in research 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.