Rep2Text: Decoding Full Text from a Single LLM Token Representation

Haiyan Zhao[1] **Zirui He**[1] **Yiming Tang**[2] **Fan Yang**[3] **Ali Payani**[4]
Dianbo Liu[2] **Mengnan Du**[5†]
[1]New Jersey Institute of Technology [2]National University of Singapore [3]Wake Forest University
[4]Cisco Research [5]School of Artificial Intelligence, The Chinese University of Hong Kong, Shenzhen
{hz54,zh296}@njit.edu {yiming,dianbo}@nus.edu.sg
yangfan@wfu.edu apayani@cisco.com mengnandu@cuhk.edu.cn
[†]Corresponding author

Abstract

Large language models (LLMs) have achieved remarkable progress across diverse tasks, yet their internal mechanisms remain largely opaque. In this work, we investigate a fundamental question: to what extent can the original input text be recovered from a single last-token representation in an LLM? To this end, we propose Rep2Text, a novel framework for decoding text from last-token representations. Rep2Text employs a trainable adapter that maps a target model's last-token representation into the token embedding space of a decoding language model, which then autoregressively reconstructs the input text. On Wikipedia-derived 16-token sequences, Rep2Text recovers roughly half of the tokens from a single last-token representation across multiple target and decoding model combinations, while preserving strong semantic coherence. Further analysis reveals a clear information bottleneck effect: as sequence length increases, token-level recovery declines, while semantic information remains relatively well preserved. We also find that decoder family matters more than decoder scale. Finally, our framework shows partial generalization to out-of-distribution clinical data. Our code is available at https://github.com/hy-zhao23/Rep2Text.

1 Introduction

Large language models (LLMs) have achieved significant progress across a wide array of tasks. Despite their impressive performance, these models are often regarded as "black boxes", limiting our understanding of their internal mechanisms. Consequently, a growing body of research has sought to decode the information encoded in LLMs. These approaches vary widely, ranging from training linear probes [1, 2] or sparse autoencoders (SAEs) [3] to interpret specific features, to mapping internal representations directly to the vocabulary space through methods like Logit Lens [4] and Tuned Lens [5]. In this work, we focus on a distinct but related challenge: *representation decoding*, which aims to reconstruct the full original text from the internal representations of language models.

Building on this perspective, we ask: To what extent can we recover the information contained in the last-token representation of an input sequence? Our overall goal is to explore single-token, activation-based input inversion. This is particularly challenging because the last-token representation is optimized for next-token prediction and can therefore be viewed as an information bottleneck. Through quantitatively comparing the inverted text with the original input, we aim to provide insight into what knowledge is preserved and encoded in the last-token representation of LLMs.

Motivated by this, we propose **Rep2Text** (Representation to Text), a framework for decoding text from the last-token representations of LLMs, as illustrated in Figure 1. Inspired by large vision

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

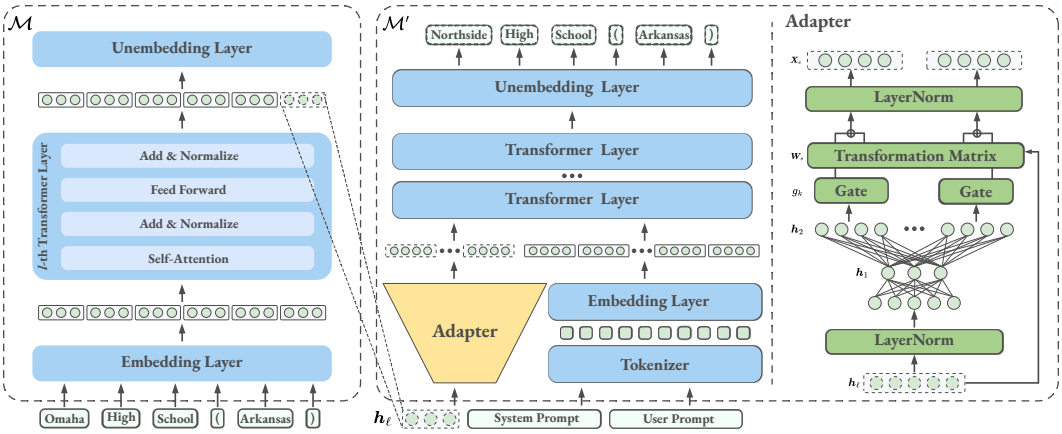


Figure 1: Overview of Rep2Text. The last-token representation obtained from the l-th layer of the target model $\mathcal{M}$ is projected into the embedding space of the decoding model $\mathcal{M}'$ via the adapter. The projected embeddings, together with those of the system and the user prompts, are then fed into the decoding model to reconstruct the corresponding text sequence.

language models (LVLMs) such as LLaVA [6], Rep2Text trains a representation inverter that consists of a decoding language model and an adapter. The adapter maps the target representation into the decoding model's token embedding space, aligning their latent spaces. These projected embeddings are subsequently fed into the decoding LLM, enabling it to interpret them and generate text consistent with the original input sequence. By comparing the inverted text against the original text, we quantify the information retained in the last-token representation.

Our experiments reveal that, remarkably, *on held-out Wikipedia sequences of length 16, Rep2Text recovers roughly half of the original tokens from a single last-token representation, while maintaining strong semantic coherence and requiring no iterative search*. This finding directly answers our central research question and demonstrates that, although last-token representations are optimized as an information bottleneck for next-token prediction, they still retain a substantial amount of recoverable information about the input sequence. To validate the effectiveness of our approach, we combine established quantitative metrics with LLM-as-a-judge evaluations to measure information retention at the token, structural, and semantic levels. Our results show that representations from different models exhibit varying recovery rates, revealing potential vulnerabilities in some models, while decoder family matters more than decoder scale. Further analysis shows that structural information is most prominent in early-to-middle layers, whereas semantic information becomes more pronounced in middle-to-late layers. We also find that recovery is strong for sequences shorter than 16 tokens but degrades for longer inputs.

2 Related Work

Embedding Inversion. The inversion is typically formulated as an optimization problem in which the attack model attempts to generate hypotheses that produce embeddings as close as possible to the target embeddings. A few works attempt to recover ground-truth sequences using sentence embeddings from BERT models. Song and Raghunathan [7] invert the sentence embeddings into a bag of words. Some works attempt to train an attacker model to decode the ground-truth sequence utilizing sentence embeddings and text embeddings [8–10]. Further, Dong et al. [11] extends embedding inversion to LLM's internal states at a certain layer, by learning token embeddings that can produce similar internal states. This method needs the internal states of all tokens. Thus, it does not apply to a single last-token representation. However, to fully recover the input text, these papers either rely heavily on iterative optimization or incorporate all sentence embeddings and token embeddings. Nikolaou et al. [12] prove that LLMs map different inputs to different hidden states. They also recover the exact input from these hidden states. In contrast, we ask how much of the input a single last-token representation retains.

Activation Decoding. Some work decodes activations into natural language. Recent work such as SelfIE [13], and Patchscopes [14] interpret representations through patching them into the forward pass of LLMs to decode natural language explanations. Besides, LIT [15] finetunes target model to answer questions related to given activations patched within. PCD [16] maps activations into concept vectors then train a model to answer questions about activations. Universal Activation Verbalizer [17] trains a shared decoder to explain activations from different donor models. Zhao et al. [18] first reconstruct the source text from an activation and then use preference optimization to reduce hallucinated descriptions. SAEExplainer [19] uses activation scores as rewards to train an explainer for SAE features. Some work explains activations by assuming that similar meanings produce similar activations. They invert an activation to find inputs that would recreate it, and use those inputs as the explanation. InverseView [20] trains a decoder to sample the input distribution for a given activation. InverseScope [21] explore task-specific features encoded in the input distribution. In contrast, we recover information solely from the last-token representation, without using the original input, partial input spans, or task-specific auxiliary hints.

3 Rep2Text Framework

In this section, we introduce the proposed *Rep2Text* framework (see Figure 1). Rep2Text employs a trainable adapter that bridges the target model's representation space to a decoding language model's embedding space, enabling us to systematically investigate what information is preserved in compressed last-token representations and how much of the original input can be recovered. The decoding LLM then autoregressively reconstructs the text from these projected embeddings.

3.1 Problem Statement

Given a token-level representation from an LLM, our objective is to reconstruct its original input sequence and quantify how much input information is retained. Throughout this work, we use *representation* and *activation* interchangeably to denote hidden states from decoder-only LLMs.

Formally, let $S = \langle s_1, \ldots, s_n \rangle$ be an input sequence and $\mathcal{M}$ be a target model with L layers. For layer $\ell \in \{1, \ldots, L\}$, let $\boldsymbol{h}^\ell$ denote the residual-stream representation of the last token after processing S. We aim to decode $\boldsymbol{h}^\ell$ into an inverted sequence $\hat{S} = \langle \hat{s}_1, \ldots, \hat{s}_m \rangle$, and measure the information preserved in $\boldsymbol{h}^\ell$ by comparing $\hat{S}$ with the original input S.

3.2 Rep2Text Inverter Design

To invert the representation, we propose an *inverter* architecture inspired by the design of typical large vision-language models such as LLaVA. The inverter consists of two key components: (1) a trainable *adapter* that projects the target model's internal representation into the input token embedding space of the decoding language model, and (2) a *decoding language model* that generates the inverted text from these projected embeddings.

Specifically, we introduce a decoding model $\mathcal{M}'$ that can either be a copy of the target model $\mathcal{M}$ or a different LLM. To bridge the representation space of $\mathcal{M}$ and the embedding space of $\mathcal{M}'$, we train an adapter to project the token representation $\boldsymbol{h}^\ell \in \mathbb{R}^d$ from the target model $\mathcal{M}$ into the token embedding space of the decoding model $\mathcal{M}'$. The adapter is implemented as a two-layer MLP with gated skip connection with optional projection, defined as:

$$\begin{aligned} &\boldsymbol{h}_1 = \mathrm{GELU}(\boldsymbol{W}_1 \cdot \mathrm{LN}(\boldsymbol{h}_\ell) + \boldsymbol{b}_1), \quad \boldsymbol{h}_2 = \boldsymbol{W}_2 \cdot \boldsymbol{h}_1 + \boldsymbol{b}_2, \quad [\tilde{\boldsymbol{x}}_1; \ldots; \tilde{\boldsymbol{x}}_k] = \mathrm{reshape}(\boldsymbol{h}_2, k, d'), \\ &\boldsymbol{x}_i = \mathrm{LN}(\boldsymbol{W}_s \cdot \boldsymbol{h}_\ell + g_i \cdot \tilde{\boldsymbol{x}}_i), \quad i = 1, \ldots, k, \quad \boldsymbol{X}_e = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_k]. \end{aligned} \tag{1}$$

where $\mathrm{LN}(\cdot)$ and $\mathrm{GELU}(\cdot)$ represent the norm layer and activation function respectively. $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d^{\mathrm{hid}}}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d^{\mathrm{hid}} \times k \cdot d'}$ refer to linear transformations in the first and second layers respectively, where d and d' represent the hidden dimensions of the target model and decoding model. Note that we set $d^{\mathrm{hid}} = f \cdot d$, where f is an expansion factor. $\boldsymbol{W}_s \in \mathbb{R}^{d \times d'}$ denotes the transformation matrix of the skip connection. When $d = d'$, $\boldsymbol{W}_s$ is an identity matrix enabling a true residual connection; when $d \neq d'$, $\boldsymbol{W}_s$ serves as a learned projection matrix to match dimensions. $\boldsymbol{h}_2 \in \mathbb{R}^{k \cdot d'}$ is reshaped into (k, d'), which can be regarded as k token embeddings. Each token embedding is

constructed with a gated combination of the skip path and the MLP-transformed path to preserve the representation information as much as possible. The projected token embedding can be denoted as $\boldsymbol{X}_e = [\boldsymbol{x}_1; \cdots; \boldsymbol{x}_k]$, where the number k of projected tokens is a hyperparameter, and g_i is a learnable scalar gate for the i-th projected token embedding, and the skip-path vector $\boldsymbol{W}_s \cdot \boldsymbol{h}^\ell$ is broadcast to each projected token position.

For each representation, the projected token embedding $\boldsymbol{X}_e$ is combined with system prompt embedding $\boldsymbol{X}_{\text{sys}}$ and user prompt embedding $\boldsymbol{X}_u$. The combined sequence $[\boldsymbol{X}_e; \boldsymbol{X}_{\text{sys}}; \boldsymbol{X}_u]$ is fed into the first layer of decoding model $\mathcal{M}'$ (after its embedding layer), bypassing the embedding layer. The decoding model then autoregressively generates the inverted text $\hat{S}$.

3.3 Rep2Text Inverter Training

For an output sequence of length T, at each decoding step t, the inverter predicts its probability conditioned on the projected embeddings, prompt embeddings, and all previously generated tokens. The joint probability of inverted sequence $\hat{S}$ is:

$$p\left(\hat{S} \mid \boldsymbol{X}_e, \boldsymbol{X}_{\text{sys}}, \boldsymbol{X}_u\right) = \prod_{t=1}^{T} p_\theta\left(\hat{s}_t \mid \boldsymbol{X}_e, \boldsymbol{X}_{\text{sys}}, \boldsymbol{X}_u, \hat{S}_{<t}\right) \tag{2}$$

where $\hat{S}_{<t}$ are the inverted tokens generated before step t, and θ denotes the trainable parameters. We consider two training schemes: (1) *adapter-only tuning*, where only the adapter parameters are optimized; (2) *full fine-tuning*, where the adapter is first fine-tuned independently and then the adapter continues to be fine-tuned while the decoding model is updated via LoRA [22]. Accordingly, θ refers to the trainable parameters under the chosen scheme.

During training, we employ teacher forcing: the generated prefix $\hat{S}_{<t}$ is replaced with the ground-truth prefix $S_{<t}$, and the model is optimized to maximize the log-likelihood of the ground-truth token s_t at each step. To stabilize training, we use label smoothing to soften the one-hot target distribution. The ground-truth token vocabulary distribution is denoted as

$$q_t(v_i) = (1-\epsilon)\mathbf{1}\left[v_i = s_t\right] + \frac{\epsilon}{|V|}, \tag{3}$$

where $\mathbf{1}(\cdot)$ is an indicator function that equals 1 if the condition holds and 0 otherwise, and ϵ is the label smoothing factor, set to be 0.075. The training objective is the smoothed cross-entropy loss:

$$\mathcal{L}_t = -\sum_{i=1}^{|V|} q_t(v_i)\, \log p_\theta(v_i \mid \mathbf{X}_e, \mathbf{X}_{\text{sys}}, \mathbf{X}_u, S_{<t}), \qquad \mathcal{L}_{\text{LS}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_t. \tag{4}$$

This training objective optimizes the adapter (and optionally the decoding model via LoRA) to minimize the prediction error across all token positions in the inverted sequence. A label smoothing term is incorporated to prevent overconfidence in token predictions, thereby improving generalization to unseen representations. Through this training process, the adapter learns to effectively map the compressed last-token representation from the target model's latent space into the decoding model's token embedding space, enabling the reconstruction of the original input sequence.

4 Experiments

In this section, we first examine the training strategy and data scale (§4.2). Next, we evaluate Rep2Text across target and decoding model combinations (§4.3). We then conduct a representation–text correspondence analysis to demonstrate that inversion relies on primarily representations rather than decoder language priors (§4.4). To analyze the information bottleneck in last-token representations, we study inversion performance across varying input lengths (§4.5) and different target-model layers (§4.6). Finally, we compare Rep2Text with the baseline method on both in-distribution and out-of-distribution datasets to evaluate its generalizability (§4.7).

4.1 Experimental Setup

Datasets. We train adapters on passages randomly truncated from Wikipedia articles in *The Pile* [23]. Each passage contains n non-overlapping tokens, where $n \in \{8, 16, 32, 64, 128, 256\}$ depending on

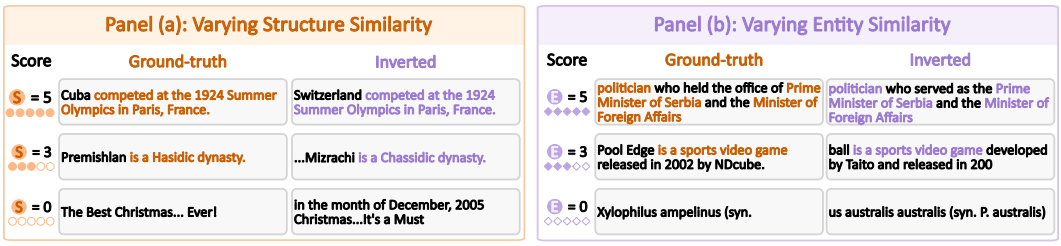


Figure 2: Examples of structural and entity preservation scores. S and E denote structural and entity similarity, respectively, rated on a 0–5 scale using an LLM-as-a-judge.

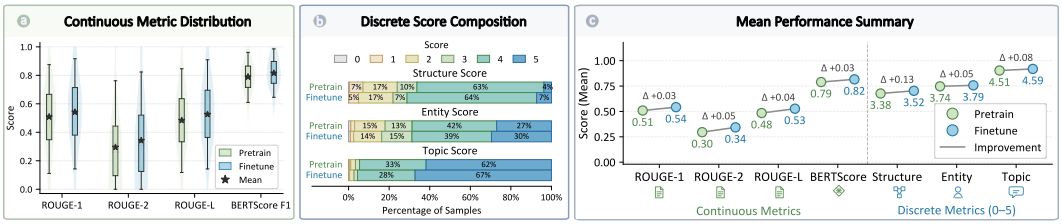


Figure 3: Adapter-only tuning versus full fine-tuning. Decoder fine-tuning improves token recovery but yields limited gains on semantic and LLM-judge metrics.

the experimental setting. Each training example pairs the last-token representation from a fixed layer of the target model with its corresponding ground-truth input sequence. We use 640K sequences for adapter-only tuning and add 960K sequences during full fine-tuning. For evaluation, we randomly sample 1,000 sequences as the test set and assess the inverted outputs using both quantitative metrics and LLM-as-a-judge evaluations.

Models. We evaluate multiple target–decoder model combinations. The decoding models include Llama-3.2-3B [24], Llama-3.1-8B [25], Qwen-2.5-14B [26], Qwen-2.5-32B [26], and Qwen-2.5-72B [26]. The target models include Llama-3.2-3B, Llama-3.1-8B, Mistral-7B-v0.1 [27], and Gemma-7B [28]. To study cross-model inversion, we fix Llama-3.1-8B as the decoder and evaluate it across all target models. To analyze scaling behavior, we invert representations from Mistral-7B-v0.1 using decoders of increasing size, including Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-14B, Qwen-2.5-32B, and Qwen-2.5-72B.

Implementation Details. Our main experiments use sequences with $n = 16$ tokens. For models up to 14B, adapter-only tuning on two A100 GPUs takes 7 hours, while full fine-tuning requires additional 12 hours. We use a peak adapter learning rate of 10^{-3}, except for Qwen models. For full fine-tuning, we use learning rates of 5×10^{-4} for the adapter and 2×10^{-4} for LoRA parameters. Models are trained for 3 epochs, except Qwen models, which are trained for 5 epochs; longer training yields only marginal gains. Additional hyperparameters are provided in Appendix I. Unless otherwise specified, all experiments fine-tune only the adapter while keeping the decoder frozen.

Evaluation Measurements. We evaluate inverted sequences along three dimensions: token-level recovery, structural and entity preservation, and semantic similarity. We use *ROUGE-1*, *ROUGE-2*, and *ROUGE-L* to measure token-level recovery. We use GPT-4.1-mini to rate structural and entity preservation on a 0–5 scale, which is normalized to 0–1. As shown in Figure 2, these scores assess whether the inverted text preserves the grammatical structure and key entities of the ground truth. We quantify semantic similarity using *BERTScore* and LLM-based topic relevance. Detailed metric definitions are provided in Appendix A.

4.2 Training Strategy and Data Scale

We first examine two practical factors in training Rep2Text: whether the decoder needs to be updated, and how much adapter training data is required.

Table 1: Performance comparison of representation inversion across target and decoding model combinations. Group (A) uses the best layer of each target model, which is the layer with the highest ROUGE-1 in the layer-wise sweep. Groups (B) and (C) fix layer 10, so that all decoders invert the same representations. Bold marks the best score in each column within each group.

Target Model	Decoding Model	Layer	ROUGE-1 (0-1)↑	ROUGE-2 (0-1)↑	ROUGE-L (0-1)↑	BERTScore (0-1)↑	Structure (0-1)↑	Entity (0-1)↑	Topic (0-1)↑
(A) Fixed Llama-3.1-8B decoder, best layer per target: target invertibility									
Gemma-7B	Llama-3.1-8B	L26	**0.64** (0.24)	**0.42** (0.30)	**0.61** (0.25)	0.80 (0.14)	**0.71** (0.22)	0.64 (0.27)	0.86 (0.22)
Mistral-7B-v0.1	Llama-3.1-8B	L15	0.55 (0.25)	0.35 (0.28)	0.53 (0.25)	**0.81** (0.13)	0.70 (0.23)	**0.75** (0.25)	**0.92** (0.17)
Llama-3.1-8B	Llama-3.1-8B	L10	0.48 (0.23)	0.28 (0.24)	0.47 (0.22)	0.78 (0.11)	0.66 (0.22)	0.74 (0.23)	0.91 (0.14)
Llama-3.2-3B	Llama-3.1-8B	L9	0.46 (0.22)	0.25 (0.22)	0.44 (0.21)	0.76 (0.11)	0.64 (0.22)	0.72 (0.24)	0.89 (0.16)
(B) Self vs. cross-model decoding (layer 10)									
Llama-3.2-3B	Llama-3.2-3B	L10	**0.46** (0.22)	**0.26** (0.23)	**0.45** (0.21)	**0.76** (0.11)	**0.64** (0.22)	0.69 (0.24)	**0.88** (0.16)
Llama-3.2-3B	Llama-3.1-8B	L10	0.45 (0.22)	0.25 (0.22)	0.43 (0.22)	**0.76** (0.11)	**0.64** (0.22)	**0.72** (0.23)	**0.88** (0.16)
(C) Decoder scale and family (Mistral-7B-v0.1, layer 10)									
Mistral-7B-v0.1	Llama-3.2-3B	L10	**0.52** (0.23)	**0.32** (0.26)	0.50 (0.23)	0.80 (0.12)	0.70 (0.21)	0.73 (0.25)	0.90 (0.17)
Mistral-7B-v0.1	Llama-3.1-8B	L10	**0.52** (0.23)	**0.32** (0.26)	**0.51** (0.23)	**0.81** (0.11)	**0.71** (0.21)	**0.75** (0.23)	0.90 (0.18)
Mistral-7B-v0.1	Qwen-2.5-14B	L10	0.48 (0.21)	0.27 (0.23)	0.47 (0.21)	0.78 (0.11)	0.65 (0.23)	0.67 (0.25)	0.89 (0.18)
Mistral-7B-v0.1	Qwen-2.5-32B	L10	0.47 (0.21)	0.25 (0.23)	0.45 (0.21)	0.76 (0.12)	0.65 (0.24)	0.65 (0.26)	0.88 (0.19)
Mistral-7B-v0.1	Qwen-2.5-72B	L10	0.51 (0.23)	0.30 (0.25)	0.49 (0.23)	0.78 (0.12)	0.68 (0.23)	0.74 (0.25)	**0.91** (0.17)

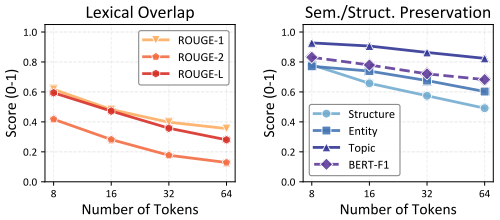
Effect of Decoder Fine-tuning. We compare *adapter-only tuning*, where the decoder is frozen, with *full fine-tuning*, where the trained adapter is further optimized together with LoRA parameters on the decoder. Figures 3(a) and 3(b) show that full fine-tuning improves inversion performance across all metrics, but the gains are modest. As summarized in Figure 3(c), full fine-tuning increases ROUGE-1, ROUGE-2, and ROUGE-L by 0.03, 0.05, and 0.04, respectively, with similarly limited gains on semantic and LLM-judge metrics. The score distributions further show that full fine-tuning mainly shifts samples toward higher token-overlap scores, while semantic metrics remain largely similar. These results suggest that representation-to-embedding alignment is primarily learned by the adapter. Therefore, all following experiments use adapter-only tuning and keep the decoder frozen.

Effect of Training Dataset Size. We additionally study how the amount of adapter training data affects inversion performance. As shown in Appendix K, increasing the dataset size from 10K to 640K leads to monotonic improvements across all eight metrics. However, the improvement from 320K to 640K shows diminishing returns around the 640K setting, making it a reasonable compute-performance trade-off for our main experiments.

4.3 Cross-Model Representation Inversion

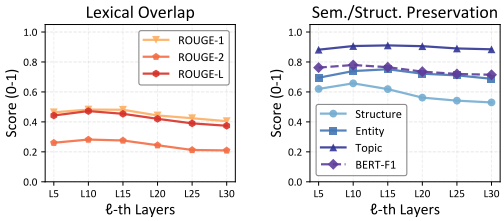
In this section, we study whether a single last-token representation can be inverted across different target and decoding models. We aim to answer four questions: (i) how much information can be recovered from a single last-token representation, (ii) whether some target models expose more recoverable information than others, (iii) whether cross-model decoding is as effective as self-model decoding, and (iv) whether larger decoding models improve inversion quality. We evaluate 16-token Wikipedia sequences and report results in Table 1. For (ii), we use the best layer of each target model, selected by ROUGE-1 in the layer-wise sweep (§4.6). For (iii) and (iv), we fix layer 10, so that all decoders invert the same representations. Across all aforementioned experiments, only the adapter is trained and the decoding model is kept frozen.

Single-token representations retain substantial recoverable information. Across target–decoder combinations, Rep2Text recovers substantial information from a single last-token representation. ROUGE-1 ranges from 0.45 to 0.64, suggesting that roughly half of the unigrams are recovered in most settings, while ROUGE-2 remains at 0.25–0.42, indicating non-trivial local phrase recovery. The inverted sequences also preserve strong semantic similarity, with BERTScore between 0.76 and 0.81 and topic preservation between 0.86 and 0.92. These results show that the last-token representation is a strong but not opaque information bottleneck, retaining both lexical fragments and high-level meaning.



(a) (b)

Figure 4: Performance comparison under different input sequence lengths.



(a) (b)

Figure 5: Performance comparison with layer-wise representation inversions.

Target models differ in representation invertibility. We fix Llama-3.1-8B as the decoder and compare representations from Gemma-7B, Mistral-7B-v0.1, Llama-3.1-8B, and Llama-3.2-3B, each at its best layer. Gemma-7B achieves the strongest lexical and structural recovery at layer 26, with 0.64 ROUGE-1, 0.42 ROUGE-2, 0.61 ROUGE-L, and 0.71 structure score. Mistral-7B-v0.1 achieves the strongest semantic recovery at layer 15, with 0.81 BERTScore, 0.75 entity score, and 0.92 topic score. By contrast, Llama-family targets are less recoverable, with Llama-3.1-8B and Llama-3.2-3B achieving 0.48 and 0.46 ROUGE-1, respectively. This suggests that sharing the same model family with the decoder does not necessarily make inversion easier; instead, recoverable input-specific information in last-token representations varies across model families. The best layer also varies across model families (§4.6).

Cross-model decoding is robust. We then compare self-model and cross-model decoding at layer 10. Using Llama-3.2-3B as both the target and decoder achieves 0.46 ROUGE-1, 0.26 ROUGE-2, and 0.45 ROUGE-L, comparable to decoding the same target representations with Llama-3.1-8B. Similarly, for Mistral-7B-v0.1 representations, Llama-3.1-8B and Llama-3.2-3B perform nearly identically, both reaching 0.52 ROUGE-1 and 0.32 ROUGE-2. These results show that successful inversion does not require the decoder to match the target model. Instead, the adapter can bridge the target representation space and decoder embedding space, suggesting that recoverable input information is partially aligned across different LLMs.

Decoder family matters more than decoder scale. Finally, we examine whether increasing decoder size improves inversion by replacing the Llama decoder with Qwen-2.5-14B, Qwen-2.5-32B, and Qwen-2.5-72B. Despite having more than twice as many parameters, Qwen-2.5-32B does not improve over Qwen-2.5-14B. Instead, performance slightly decreases from 0.48 to 0.47 in ROUGE-1, from 0.27 to 0.25 in ROUGE-2, and from 0.78 to 0.76 in BERTScore. Qwen-2.5-72B improves over Qwen-2.5-32B, but it still underperforms the smaller Llama decoders on most metrics when inverting Mistral-7B-v0.1 representations. These results suggest that cross-model decoding is robust to decoder scale but sensitive to decoder family.

4.4 Representation-Text Correspondence Analysis

To test whether Rep2Text relies on representation-specific information rather than decoder language priors, we compare three settings: *Baseline*, which uses correctly paired representations and original inputs; *Reverse*, which reconstructs word-level reversed and retokenized inputs; and *Random*, which pairs each representation with an unrelated target sequence and breaks the alignment. Random serves as a negative control. Any performance under Random comes from decoder language priors alone.

Figure E.1 shows that breaking the representation–text correspondence leads to a clear performance collapse. The *Random* setting performs substantially worse than the correctly paired *Baseline* setting across loss, lexical overlap, and semantic similarity, indicating that Rep2Text cannot recover meaningful text from unrelated representation–text pairs and does not rely solely on decoder language priors. In contrast, *Reverse* remains partially learnable despite violating the natural left-to-right word order favored by LLMs. This suggests that the last-token representation still contains recoverable input-specific information under a deterministic word-order transformation. Detailed experimental settings and quantitative results are provided in Appendix E.

Table 2: Inversion comparison between Vec2Text and our method with input sequences of no more than 32 tokens.

Target model/ decoding model	Data	Inversion Model	R1 (0-1)↑	R2 (0-1)↑	RL (0-1)↑	Token F1 (0-1)↑	BLEU (0-1)↑	BS (0-1)↑	SS (0-1)↑	ES (0-1)↑	TS (0-1)↑
Mistral-7B-v0.1/ Qwen-2.5-14B	Wiki	Rep2Text (Ours)	0.41	0.19	0.38	0.42	0.15	0.73	0.57	0.66	0.86
		Vec2Text Base	0.38	0.17	0.35	0.31	0.15	0.71	0.53	0.53	0.80
		* + Corrector (50 Steps)	0.38	0.17	0.35	0.31	0.15	0.71	0.53	0.55	0.79
	Clinical	Rep2Text (Ours)	0.37	0.15	0.34	0.26	0.10	0.74	0.59	0.48	0.64
		Vec2Text Base	0.14	0.02	0.13	0.13	0.04	0.53	0.17	0.16	0.21
		* + Corrector (50 Steps)	0.13	0.01	0.12	0.11	0.03	0.52	0.14	0.13	0.20
Llama-3.1-8B/ Qwen-2.5-14B	Wiki	Rep2Text (Ours)	0.36	0.15	0.32	0.39	0.12	0.70	0.53	0.63	0.85
		Vec2Text Base	0.37	0.16	0.34	0.29	0.14	0.70	0.54	0.51	0.78
		* + Corrector (50 Steps)	0.35	0.15	0.31	0.27	0.12	0.68	0.50	0.49	0.76
	Clinical	Rep2Text (Ours)	0.34	0.12	0.31	0.24	0.08	0.71	0.50	0.47	0.56
		Vec2Text Base	0.13	0.01	0.12	0.12	0.03	0.55	0.23	0.18	0.25
		* + Corrector (50 Steps)	0.12	0.01	0.11	0.11	0.03	0.51	0.17	0.14	0.22

* denotes Vec2Text

4.5 Effects of Sequence Length

We next examine how input length affects recoverability from a single last-token representation. Because this representation summarizes all preceding tokens, longer inputs should create a stronger information bottleneck. We train separate adapters to invert layer-10 representations from 8-, 16-, 32-, and 64-token Wikipedia sequences, using Llama-3.1-8B as both the target and decoding model based on Section 4.3.

Figure 4 shows that lexical recovery consistently decreases with input length. ROUGE-1 drops from about 0.60 for 8-token inputs to about 0.36 for 64-token inputs, with similar declines in ROUGE-2 and ROUGE-L. Structural preservation also degrades, suggesting that longer contexts make detailed wording and grammar harder to recover from a single representation. In contrast, semantic metrics decline more slowly: BERTScore, entity preservation, and topic preservation remain relatively high for longer inputs. Overall, a single last-token representation preserves the general topic of longer contexts better than exact wording or sentence structure. We further test 128- and 256-token inputs with Llama-3.2-3B as both the target and decoder. ROUGE-1 drops from 0.46 at 16 tokens to 0.30 at 128 and 256 tokens. In contrast, BERTScore stays at 0.63 and the topic score stays at or above 0.68, as detailed in Appendix G.

4.6 Layer-wise Invertibility

Prior work shows that middle-to-deep layers capture high-level semantics more effectively [29, 30]. We therefore ask which layer contains the most recoverable input information. To answer this, we train separate adapters on last-token representations from layers 5, 10, 15, 20, 25, and 30 of Llama-3.1-8B, using 16-token Wikipedia sequences.

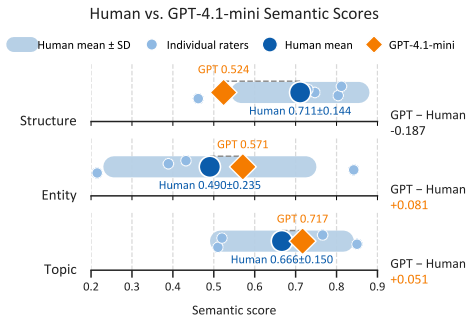


Figure 6: Human evaluation sanity check for GPT-4.1-mini semantic scores.

Figure 5 shows that recoverability is strongest around L10–L15, with different metrics peaking at slightly different depths. ROUGE-L, structure score, and BERTScore peak around layer 10, suggesting that sentence structure, local phrase organization, and semantic similarity are already well captured by early-to-middle layers. ROUGE-1 is nearly identical at layers 10 and 15, while entity recovery peaks at layer 15, indicating that entity-level details may require deeper processing. Topic preservation remains stable from layers 10 to 20, suggesting that high-level semantic information persists across middle-to-late layers. Appendix F reports layer-wise sweeps for the other three target models. The best layer depends on the model architecture.

Table 3: Inverted examples of the three methods on the same inputs. Bold marks the best Token F1 in each group.

Method	Ground-truth Sequence	Inverted Sequence	Token F1	BLEU
In-distribution, Llama-3.1-8B target				
Ours	Phil LaMarr Phillip LaMarr (born January 24, 1967) is an American actor, voice actor, comedian and writer.	Phil LaMarr Philip LaMarr (born January 24, 1967) is an American actor, voice actor, comedian and writer.	**0.94**	0.89
Vec2Text Base		David A. Smith David A. Smith (born December 16, 1964) is an American actor, voice actor, writer, and voice actor.	0.43	0.32
Vec2Text + Corrector (50 Steps)		David A. Smith David A. Smith (born August 24, 1964) is an American actor, voice actor, writer, and voice actor.	0.49	0.33
In-distribution, Mistral-7B-v0.1 target				
Ours	List of submissions to the 37th Academy Awards for Best Foreign Language Film The following 18 films, all from different countries, were	List of films submitted to the 80th Academy Awards for Best Foreign Language Film The following 14 films, from 14 different countries,	0.77	0.46
Vec2Text Base		List of submissions to the 59th Academy Awards for Best Foreign Language Film. The following eight films, all from different countries, were	**0.91**	0.76
Vec2Text + Corrector (50 Steps)		Same as Vec2Text Base.	**0.91**	0.76
Out-of-distribution (OOD), Mistral-7B-v0.1 target				
Ours	A 35-year-old female with abnormal heavy periods, fatigue, and dark spots on her hands and neck.	A 35-year-old woman with a BMI of 300 and a history of acne, hirsutism, and infertility.	**0.31**	0.15
Vec2Text Base		He was born in New York City and died in New York City in January 2013. During his stay at the YMCA,	0.05	0.02
Vec2Text + Corrector (50 Steps)		A rare condition characterized by a swollen thigh, a swollen thigh, a swollen thigh, and a swollen thigh, on	0.11	0.04

4.7 Inversion Performance Analysis

To further evaluate Rep2Text and its usefulness for representation interpretation, we compare it with Vec2Text [9], a strong embedding inversion baseline that uses a hypothesizer and a corrector model. Because our setting reconstructs text from last-token representations rather than sentence embeddings, we adapt Vec2Text to the same representation-inversion setting.

We train the Vec2Text hypothesizer and corrector on the same Wikipedia data used for Rep2Text, following the original setup and training each model for 10 epochs. Both methods invert last-token representations from Mistral-7B-v0.1, and Rep2Text uses Qwen-2.5-14B as the decoding model. We evaluate on held-out Wikipedia data and OOD clinical summaries containing patient names, ages, and symptoms, using an inversion length of 32 tokens. We use 32 tokens because the clinical summaries contain up to 25 tokens, so this length does not truncate clinical information. Since the hypothesizer is the core inversion model in Vec2Text and the corrector mainly refines its output, we treat the hypothesizer as the primary baseline. Appendix D discusses the difference in decoder size between the two methods. Additional details are provided in Appendix H.

Table 2 reports token-level, semantic, and LLM-judge metrics, including *Token-F1* and *BLEU* for comparison with Vec2Text. On in-distribution Wikipedia data, both methods achieve comparable semantic performance, while Rep2Text recovers more token-level information, improving *Token F1* by 11% and *Entity Score* by 13%. On OOD clinical data, Rep2Text generalizes substantially better, achieving 0.26 *Token F1*, 0.37 *ROUGE-1*, 0.64 *Topic Score*, and 0.10 *BLEU*, whereas Vec2Text Base reaches only 0.13 *Token F1*, 0.14 *ROUGE-1*, 0.21 *Topic Score*, and 0.04 *BLEU*.

To assess the reliability of the LLM-judge metrics, we further conduct a human evaluation check. As shown in Figure 6, GPT-4.1-mini produces scores in a similar range to human annotations, but

tends to underestimate structure preservation and slightly overestimate entity and topic preservation. This suggests that LLM-as-a-judge is a useful scalable proxy, while its absolute scores should be interpreted with metric-specific caution. Detailed settings and discussion are provided in Appendix C.

We also evaluate Vec2Text with 50 corrector steps, in which the corrector iteratively refines the hypothesizer's output. The corrector provides only marginal gains on OOD data, despite converged training metrics. This may be because Vec2Text was designed for text encoder embeddings, while last-token representations may provide a less informative feedback signal for iterative editing. In contrast, Rep2Text directly aligns the representation space with the decoder embedding space, which appears more robust under distribution shift.

Table 3 shows the outputs of the three methods on the same inputs. On in-distribution data, the methods show mixed results. Rep2Text is nearly exact on the first example, while Vec2Text performs better on the second. On OOD clinical data, Rep2Text better preserves the clinical context. In contrast, Vec2Text produces unrelated or repetitive text. Rep2Text can still hallucinate specific attributes, such as an implausible BMI. Thus, exact recovery of clinical facts remains imperfect. Overall, these examples agree with Table 2. The two methods perform similarly on in-distribution data, but Rep2Text generalizes better to OOD data.

5 Conclusions

In this work, we explore the research question that to what extent can the original input text be recovered from a single last-token representation of LLMs? To answer this question, we proposed Rep2Text, a novel framework that employs a trainable adapter to project a target model's last-token representation into the embedding space of a decoding language model, which then autoregressively reconstructs the input text. Our comprehensive evaluations indicate that roughly half of the tokens in 16-token sequences can be recovered from the compressed last-token representation while maintaining strong semantic integrity. Besides, experimental results show longer sequences lead to decreased inversion performance, with reliable recovery achieved for sequences under 16 tokens. Additionally, Rep2Text also shows partial transfer to out-of-distribution clinical data, recovering coarse structural and semantic information, although exact recovery of critical attributes remains imperfect. Rep2Text shows that a single intermediate representation can reveal part of the input text. Thus, deployed systems should protect intermediate representations. Our clinical evaluation uses LLM-generated summaries of a public dataset. Developing safeguards against representation inversion is future work.

References

[1] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[2] Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*, 2025.

[3] Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. *EMNLP Findings*, 2025.

[4] nostalgebraist. interpreting gpt: the logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. LessWrong.

[5] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[6] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[7] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 377–390, 2020.

[8] Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14022–14040, 2023.

[9] John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, 2023.

[10] Yu-Hsiang Huang, Yuche Tsai, Hsiang Hsiao, Hong-Yi Lin, and Shou-De Lin. Transferable embedding inversion attack: Uncovering privacy risks in text embeddings without model queries. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4193–4205, 2024.

[11] Tian Dong, Yan Meng, Shaofeng Li, Guoxing Chen, Zhen Liu, and Haojin Zhu. Depth gives a false sense of privacy: Llm internal states inversion. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 1629–1648, 2025.

[12] Giorgos Nikolaou, Tommaso Mencattini, Donato Crisostomi, Andrea Santilli, Yannis Panagakis, and Emanuele Rodolà. Language models are injective and hence invertible. In *International Conference on Learning Representations*, volume 2026, pages 57927–57984, 2026.

[13] Haozhe Chen, Carl Vondrick, and Chengzhi Mao. Selfie: self-interpretation of large language model embeddings. In *Proceedings of the 41st International Conference on Machine Learning*, pages 7373–7388, 2024.

[14] Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: a unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 15466–15490, 2024.

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

[16] Vincent Huang, Dami Choi, Daniel D Johnson, Sarah Schwettmann, and Jacob Steinhardt. Predictive concept decoders: Training scalable end-to-end interpretability assistants. *arXiv preprint arXiv:2512.15712*, 2025.

[17] Haiyan Zhao, Zirui He, Guanchu Wang, Ali Payani, Yingcong Li, and Mengnan Du. Universal activation verbalizer: A unified framework for cross-model activation explanation. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

[18] Haiyan Zhao, Zirui He, Wei Shi, Huiqi Deng, Na Zou, and Mengnan Du. Faithful activation verbalization: Reducing hallucinations in llm representation interpretation. *arXiv preprint arXiv:2609.34033*, 2026.

[19] Jingyi He, Haiyan Zhao, Ruxue Shi, Yanguang Liu, Xin Wang, Fei Sun, and Mengnan Du. Saeexplainer: Interpreting sae features with activation-guided preference optimization. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

[20] Xinting Huang, Madhur Panwar, Navin Goyal, and Michael Hahn. Inversionview: a general-purpose method for reading information from neural activations. *Advances in Neural Information Processing Systems*, 37:137903–137964, 2024.

[21] Yifan Luo, Zhennan Zhou, and Bin Dong. Inversescope: Scalable activation inversion for interpreting large language models. *arXiv preprint arXiv:2506.07406*, 2025.

[22] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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

[24] Meta AI. Llama-3.2-3b. https://huggingface.co/meta-llama/Llama-3.2-3B, 2025. [Accessed 29-10-2025].

[25] Meta AI. Llama-3.1-8b. https://huggingface.co/meta-llama/Llama-3.1-8B, 2024. [Accessed 29-10-2025].

[26] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[27] Mistral AI. Mistral-7b-v0.1. https://huggingface.co/mistralai/Mistral-7B-v0.1, 2023. [Accessed 29-10-2025].

[28] Google. Gemma-7b. https://huggingface.co/google/gemma-7b, 2024. [Accessed 29-10-2025].

[29] Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *The 31st International Conference on Computational Linguistics (COLING 2025)*, 2025.

[30] James Campbell, Richard Ren, and Phillip Guo. Localizing lying in llama: Understanding instructed dishonesty on true-false questions through prompting, probing, and patching. *arXiv preprint arXiv:2311.15131*, 2023.

[31] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[32] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *The International Conference on Learning Representations (ICLR)*, 2020.

A Metrics

All used metrics are defined as follows:

Token-level Accuracy. We adopt ROUGE scores to measure the token-level accuracy [31]. Specifically, *ROUGE-1*, and *ROUGE-2* measure the recovery rate of individual tokens and 2-grams, respectively, while *ROUGE-L* captures the longest common subsequence between the ground-truth and the inverted sequence. Detailed definitions of these metrics are as below:

- **ROUGE-1 and ROUGE-2**: For ROUGE-k, it computes the F-measure of k-grams extracted from a sequence. Suppose the set of k-grams is denoted as $G_k(S)$ and $G_k(\hat{k})$ for ground-truth sequence and inverted sequence respectively. ROUGE-k is computed as follows:

$$\begin{aligned} \text{Overlap}_k &= \sum_{g \in G_k(S) \cap G_k(\hat{S})} \min\left(\text{cnt}_S(g), \text{cnt}_{\hat{S}}(g)\right) \\ R_k &= \frac{\text{Overlap}_k}{|G_k(S)|}, \quad P_k = \frac{\text{Overlap}_k}{|G_k(\hat{S})|} \\ \text{ROUGE}_k &= \frac{(1+\beta^2) R_k P_k}{R_k + \beta^2 P_k} \end{aligned} \tag{5}$$

  where $\text{cnt}(\cdot)$ denotes the count of the set.

- **ROUGE-L**: Give an ground-truth sequence $S = \langle s_1, \ldots, s_n \rangle$ and an inverted sequence $\hat{S} = \langle \hat{s}_1, \ldots, \hat{s}_m \rangle$, the length of their longest common subsequence is $LCS(S, \hat{S})$. The ROUGE-L score F_{LCS} is defined as follows:

$$\begin{aligned} R_{LCS} &= \frac{LCS(S,\hat{S})}{n}, \quad P_{LCS} = \frac{LCS(S,\hat{S})}{m} \\ F_{LCS} &= \frac{(1+\beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \end{aligned} \tag{6}$$

Sentence Structure and Entity Preservation. To evaluate the preservation of sentence structure and entities, we use GPT-4.1-mini to rate the degree of preservation ranging on a 0-5 scale (normalized to 0-1), yielding the *Structure Score* and *Entity Score*, respectively. The structure score assesses how well the grammatical structure and sentence skeleton are preserved in the inverted sequences. While the entity score measures how accurately entity names and their associated attributes are inverted. Detailed rating criteria are provided in Appendix L.1.

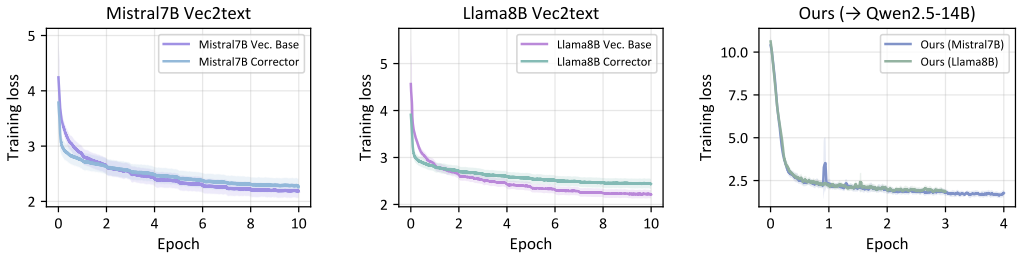
Semantic Similarity. We use BERTScore F1 and LLM-as-a-judge to collectively evaluate the semantic similarity between the ground-truth and inverted sequences [32]. BERTScore quantifies similarity in the embedding space, whereas the LLM-based evaluation measures topic relevance between ground-truth and inverted sequences. The scoring guidelines for LLM-as-a-judge evaluation are included in Appendix L.1.

B Inverted Examples with Varying Token Lengths and Recovery Rate

As shown in Table L.1, for sequences with 8 tokens and 16 tokens, some inverted sequences fully recover the original text, while others fail to capture fine-grained details yet still preserve clear grammatical structures. For example, when inverting 16-token sequence, although the ROUGE-1 score is only 0.4, the original sentence "*Rob James may refer to:*\n\n*Rob James (singer) (*" and the inverted sequence "*Mark Jones*\n\n*Mark Jones may refer to:*\n\n*Mark Jones (singer) (*" share the same syntactic pattern, "[NAME]\n\n[NAME] (singer) (", and convey equivalent topic-level information. However, when the number of tokens exceed 16, the inverted sequences remain highly topic-relevant but tend to lose their global structural coherence, even when achieving a reasonable ROUGE-1 score.

C Human Sanity Check

The reported human scores correspond to the mean ratings over the 200 sampled sequences. The five annotators are volunteers and receive no monetary compensation. To characterize the variability of human judgment, we also report the standard deviation across the mean scores of the five annotators.



(a) (b) (c)

Figure D.1: Training dynamics of both Vec2text and our methods.

As shown in Figure 6, GPT-4.1-mini produces scores that are close to the range of human annotations across all three dimensions. In particular, its entity and topic scores are close to the human mean, while its structure score is lower than the human average but still comparable to the ratings assigned by several individual annotators. This suggests that GPT-4.1-mini does not behave as an outlier evaluator, but rather reflects one plausible annotation tendency within the range of human judgments.

At the same time, the comparison reveals metric-specific bias. GPT-4.1-mini is more conservative than human annotators when evaluating structure preservation, whereas it gives slightly higher scores for entity and topic preservation. Therefore, we use LLM-as-a-judge as a scalable proxy for semantic evaluation, while interpreting the absolute scores with caution, especially for structure preservation.

D More Results on Inversion Performance Analysis

We compare Rep2Text with the adapted Vec2Text baseline under the last-token representation inversion setting. For Vec2Text, both the hypothesizer and corrector follow the original two-stage design with T5-base, using a learning rate of 10^{-3} and a warmup ratio of 0.10. For Rep2Text, we use Qwen-2.5-14B as the decoding model and train only the adapter while keeping the decoder frozen, using a learning rate of 1.5×10^{-3} and a warmup ratio of 0.30. All runs are conducted on two A100 GPUs.

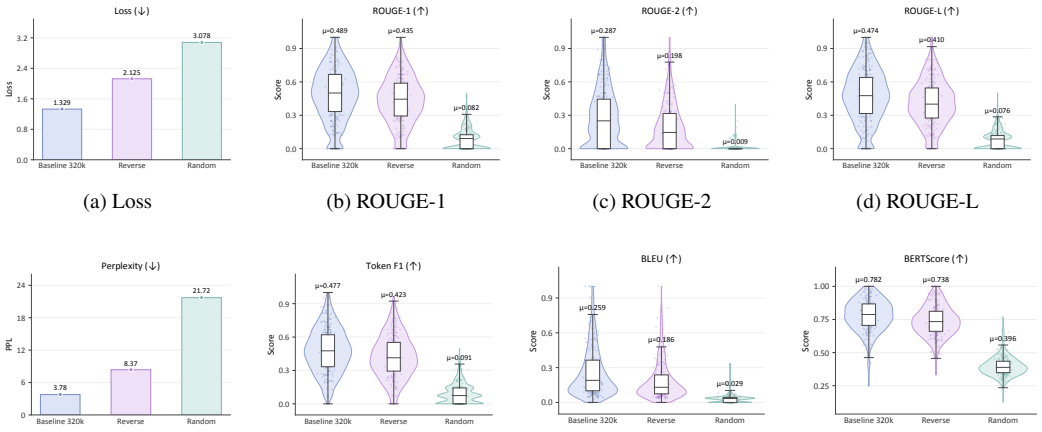
The two methods use decoders of different sizes. Vec2Text uses T5-base, while Rep2Text uses Qwen-2.5-14B. However, decoder size has a limited effect on Rep2Text. In the 16-token setting, the Llama-3.2-3B decoder inverts Mistral-7B-v0.1 representations at least as well as Qwen-2.5-14B on all metrics, as shown in Table 1. Meanwhile, Rep2Text outperforms Vec2Text Base by a large margin on OOD clinical data. It reaches 0.37 ROUGE-1 and 0.64 topic score, while Vec2Text Base reaches 0.14 and 0.21. Thus, the gap is unlikely to come from decoder capacity alone.

Figure D.1 shows that the Vec2Text corrector reduces training loss, but this does not translate into better inversion performance. As shown in Table 2, adding the corrector brings no consistent improvement and even degrades the base model, especially in the OOD clinical setting. This suggests that the corrector does not learn an effective refinement strategy for last-token representation inversion.

In contrast, Rep2Text rapidly reduces training loss and achieves stronger downstream inversion performance. This supports our hypothesis that directly aligning the target representation space with the decoder embedding space is more effective than iteratively editing decoded hypotheses when the input signal is a compressed last-token representation.

E Results on Representation-Text Correspondence Analysis

We conduct three experiments to examine whether Rep2Text relies on the correspondence between representations and reconstruction targets. All settings use Mistral-7B-v0.1 as the target and Llama-3.2-3B as the decoder. The maximum sequence length is set to 16 tokens, and the number of projected vectors is set to 16. Experiments are trained with an effective batch size of 512, a learning rate of 7×10^{-4}, a warmup ratio of 0.15, and 4 epochs on two A100 GPUs.



(a) Loss (b) ROUGE-1 (c) ROUGE-2 (d) ROUGE-L

(e) Perplexity (f) Token F1 (g) BLEU (h) BERTScore F1

Figure E.1: Performance comparison under different representation–text correspondence settings. The baseline uses correctly paired representations and original input sequences, Reverse uses word-level reversed input sequences as reconstruction targets, and Random randomly pairs representations with unrelated target sequences. The strong degradation under Random indicates that Rep2Text relies on representation-specific information rather than only decoder language priors.

In the *Baseline* setting, each representation is paired with its original input sequence. In the *Reverse* setting, we reverse the input sequence at the word level and then retokenize it as the reconstruction target. In the *Random* setting, each representation is paired with an unrelated target sequence, thereby breaking the representation–text correspondence.

As shown in Figure E.1, *Random* performs substantially worse than *Baseline*: the training loss increases from 1.329 to 3.078, while ROUGE-L, Token F1, and BERTScore decrease from 0.474, 0.477, and 0.782 to 0.076, 0.091, and 0.396, respectively. This confirms that the inverter cannot learn meaningful reconstruction when representations are paired with unrelated text.

By contrast, *Reverse* achieves 0.410 ROUGE-L, 0.423 Token F1, and 0.738 BERTScore. Although it underperforms the original-order *Baseline*, it remains far above *Random*, suggesting that the model can recover input-specific information even when the target follows an unnatural word-order.

F Layer-wise Invertibility across Target Models

We sweep the layers of Mistral-7B-v0.1, Gemma-7B, and Llama-3.2-3B at roughly matched relative depths. All runs use Llama-3.1-8B as the decoder and 16-token Wikipedia sequences. Table F.1 reports the results together with the Llama-3.1-8B sweep from §4.6. Mistral-7B-v0.1 performs best around its middle layers. Llama-3.2-3B performs similarly from layer 4 to layer 13 and degrades in later layers. Gemma-7B performs best in its late layers. Thus, the best inversion layer depends on the model architecture.

G Inversion of Longer Sequences

We extend the sequence-length analysis in §4.5 to 128 and 256 tokens. These runs use Llama-3.2-3B as both the target and decoder. We also train a 16-token model under the same setting as an anchor. Table G.1 reports the results.

Lexical and structural recovery drop sharply as inputs grow. Under the 3B setting, ROUGE-1 drops from 0.46 at 16 tokens to 0.30 at 128 tokens and stays at 0.30 at 256 tokens. ROUGE-2, ROUGE-L, entity, and structure scores keep declining. Semantic metrics drop much more slowly. At 256 tokens, the topic score is 0.68 and BERTScore is 0.63. This BERTScore is well above the 0.40 of the Random control in Appendix E. Thus, a single last-token representation loses most of the exact wording of long inputs but keeps their coarse topic.

Table F.1: Layer-wise inversion performance of four target models with Llama-3.1-8B as the decoder. Bold rows mark the layer with the highest ROUGE-1, which Table 1 uses. The L10 rows of Mistral-7B-v0.1, Gemma-7B, and Llama-3.2-3B come from the main runs in Table 1.

Layer	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	Structure	Entity	Topic
Llama-3.1-8B (32 layers)							
L5	0.46	0.26	0.44	0.76	0.62	0.69	0.88
L10	**0.48**	**0.28**	**0.47**	**0.78**	**0.66**	**0.74**	**0.91**
L15	0.48	0.28	0.45	0.77	0.62	0.75	0.91
L20	0.44	0.24	0.42	0.74	0.56	0.72	0.91
L25	0.42	0.21	0.39	0.72	0.54	0.71	0.89
L30	0.40	0.21	0.37	0.71	0.53	0.69	0.88
Mistral-7B-v0.1 (32 layers)							
L5	0.52	0.32	0.50	0.79	0.67	0.71	0.89
L10	0.52	0.32	0.51	0.81	0.71	0.75	0.90
L15	**0.55**	**0.35**	**0.53**	**0.81**	**0.70**	**0.75**	**0.92**
L20	0.53	0.33	0.52	0.79	0.64	0.74	0.92
L25	0.49	0.29	0.47	0.76	0.60	0.72	0.90
L30	0.47	0.26	0.45	0.75	0.60	0.72	0.91
Gemma-7B (28 layers)							
L4	0.45	0.24	0.43	0.72	0.63	0.56	0.76
L9	0.53	0.31	0.51	0.76	0.68	0.59	0.80
L10	0.51	0.28	0.49	0.75	0.66	0.59	0.79
L13	0.53	0.31	0.51	0.76	0.68	0.58	0.79
L18	0.50	0.28	0.49	0.74	0.68	0.56	0.77
L22	0.57	0.35	0.55	0.77	0.68	0.61	0.82
L26	**0.64**	**0.42**	**0.61**	**0.80**	**0.71**	**0.64**	**0.86**
Llama-3.2-3B (28 layers)							
L4	0.45	0.25	0.43	0.75	0.60	0.67	0.87
L9	**0.46**	**0.25**	**0.44**	**0.76**	**0.64**	**0.72**	**0.89**
L10	0.45	0.25	0.43	0.76	0.64	0.72	0.88
L13	0.46	0.25	0.44	0.76	0.62	0.73	0.90
L18	0.42	0.22	0.39	0.72	0.55	0.71	0.90
L22	0.39	0.19	0.37	0.71	0.53	0.70	0.89
L26	0.38	0.18	0.36	0.70	0.53	0.69	0.88

H More Details on OOD Experiments

H.1 OOD Dataset

We use an open-source clinical dataset from Hugging Face[1]. To fit patient histories within 32 tokens, we use Qwen-2.5-32B to summarize 2,500 randomly sampled patient notes. The instruction is:

> You're given a piece of clinical note in the following context. Summarize the clinical note into ONE fluent sentence (<=25 tokens). Must include: patient name, date of birth, gender, and key symptoms. Start with "[ans]". Output only the summary.
> {note}

[1] https://huggingface.co/datasets/Sadaftb/clinical-nlp-patient-notes

Table G.1: Inversion performance across input lengths. 8B denotes Llama-3.1-8B and 3B denotes Llama-3.2-3B. Each model serves as both the target and the decoder.

Tokens	Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	Structure	Entity	Topic
8	8B	0.62	0.42	0.59	0.83	0.78	0.77	0.93
16	8B	0.48	0.28	0.47	0.78	0.66	0.74	0.91
32	8B	0.40	0.18	0.36	0.72	0.57	0.68	0.86
64	8B	0.36	0.13	0.28	0.68	0.49	0.60	0.82
16	3B	0.46	0.26	0.45	0.76	0.64	0.69	0.88
128	3B	0.30	0.08	0.22	0.63	0.37	0.41	0.72
256	3B	0.30	0.07	0.20	0.63	0.36	0.39	0.68

H.2 Experimental Details on Vec2Text

We adapted the original Vec2Text into inverting last-token representation. Here, we use Mistral-7B-v0.1 as our embedding model. The training of both base model and corrector adopts default training parameters of Vec2Text. To make sure the training is converged well without overfitting, we set up both training process to be 10 epochs which takes more than 18 hours separately. Base model is set to be t5-base, num repeat tokens are 16 and max sequence length is set to be 64.

H.3 Inverted Examples with OOD Clinical Notes

Table 3 compares the three methods on the same inputs.

I More Implementation Details

This section provides additional implementation details and experiment-specific hyperparameters omitted from the main text. Unless otherwise specified, all experiments use the same default training configuration described below.

Unless otherwise specified, all experiments are trained on Wikipedia with 640K training examples, using embeddings extracted from layer 10. We train for 3 epochs with a cosine learning rate scheduler and a warmup ratio of 0.15. The default learning rate is 10^{-3}, except where noted otherwise. We use Adam with $\beta = (0.9, 0.95)$, weight decay 0.01, label smoothing 0.075, and adapter dropout 0.1. Unless otherwise stated, the number of projected token vectors is set equal to the number of tokens being inverted, as this gives the best performance (Appendix J), and the hidden expansion factor is fixed at 0.5. Most experiments are run on two NVIDIA A100 GPUs (80GB each), with a global batch size of 1024 implemented through per-GPU batch size and gradient accumulation. More detailed setups are shown in Table I.1.

J Ablation Study on Adapter Hidden Dimensions

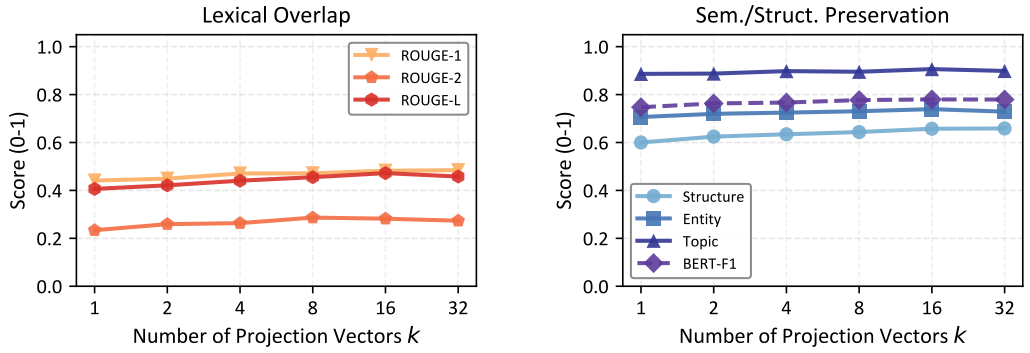
We study how adapter design affects inversion performance by varying the hidden and output dimensions of the two-layer MLP adapter. All experiments are conducted on 16-token sequences using layer-10 representations from Llama-3.1-8B.

As defined in Section 3.2, the hidden dimension of the first adapter layer is $d^{\text{hid}} = f \cdot d$, where d is the input representation dimension and f is an expansion factor. To evaluate the effect of hidden-layer capacity, we vary $f \in \{0.5, 1, 2, 4, 8\}$ while keeping the second-layer output dimension fixed. As shown in Figure J.2, inversion performance is largely robust to changes in the hidden dimension. Semantic-level recovery is nearly unchanged, while token-level recovery, measured by ROUGE-1, improves slightly as the hidden dimension increases. This suggests that a larger hidden layer may provide additional capacity for recovering lexical information, but is not critical for preserving semantic content.

We further vary the output dimension of the second adapter layer. Specifically, we set the output dimension to $k \cdot d'$, where d' is the embedding dimension of the decoder. This corresponds to

Table I.1: Experiment-specific hyperparameters for transfer, token-length, and layer-sweep experiments. All unspecified settings follow the shared default configuration described in Appendix I. The Qwen2.5-72B run uses a tensor-parallel size of 4, so its 8 GPUs form 2 data-parallel replicas.

Emb Model	Dec Model	Max Len	n_{vecs}	#GPU	Mini batch	Grad Accum	Eff. Batch	LR	Warmup Ratio
Transfer experiments									
Gemma-7B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Llama-3.2-3B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Mistral-7B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Mistral-7B	Llama-3.2-3B	16	16	2	64	8	1024	1e−3	0.15
Mistral-7B	Qwen2.5-14B	16	16	2	32	16	1024	1e−3	0.30
Mistral-7B	Qwen2.5-14B	32	32	4	16	8	512	1.5e−3	0.30
Mistral-7B	Qwen2.5-32B	16	16	8	28	9	2016	1.5e−3	0.30
Mistral-7B	Qwen2.5-72B	16	16	8	64	8	1024	1e−3	0.15
Token-length experiments (Llama-3.1-8B ↔ Llama-3.1-8B)									
Llama-3.1-8B	Llama-3.1-8B	8	8	2	32	16	1024	1e−3	0.15
Llama-3.1-8B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Llama-3.1-8B	Llama-3.1-8B	32	32	2	32	16	1024	1e−3	0.15
Llama-3.1-8B	Llama-3.1-8B	64	64	4	16	16	1024	3e−3	0.15
Token-length experiments (Llama-3.2-3B ↔ Llama-3.2-3B)									
Llama-3.2-3B	Llama-3.2-3B	128	128	2	16	32	1024	1e−3	0.15
Llama-3.2-3B	Llama-3.2-3B	256	256	2	8	64	1024	1e−3	0.15
Layer sweeps (all swept layers)									
Mistral-7B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Gemma-7B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15
Llama-3.2-3B	Llama-3.1-8B	16	16	2	32	16	1024	1e−3	0.15

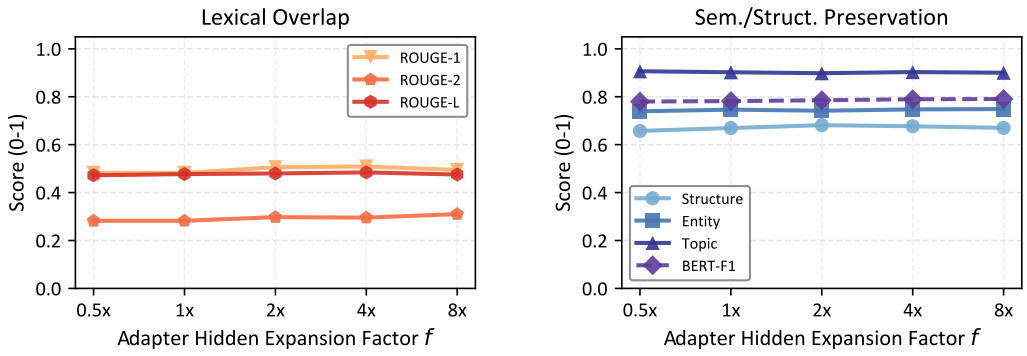


(a) (b)

Figure J.1: Inversion performance on different number of inverted embedding tokens.

projecting the target representation into k token-embedding slots. We vary $k \in \{1, 2, 4, 8, 16, 32\}$ to test whether more projected embeddings help recover more information from the last-token representation. As shown in Figure J.1, both token-level and semantic-level performance improve as k increases up to 16. However, performance degrades when $k = 32$, which exceeds the ground-truth sequence length. This suggests that adding more projected embeddings than the target length may introduce redundant or noisy information.

Overall, the first-layer expansion factor has limited impact on inversion performance, whereas the number of projected token embeddings is more important. Performance is strongest when the number of projected embeddings matches the ground-truth token length. Based on these findings, we use $f = 0.5$ and set the number of projected embeddings to the ground-truth token length in our main experiments.



(a) (b)

Figure J.2: Inversion Performance on varying expansion factors.

K Training Dataset Size Ablation

To examine whether Rep2Text benefits smoothly from additional training data, we train adapters with 10K, 50K, 100K, 320K, and 640K Wikipedia examples while adjusting the effective batch size and learning rate using square-root scaling. The 640K setting corresponds to the default configuration used in the main experiments.

Table K.1: Training configurations for the dataset size ablation.

Train Size	Epochs	Per-GPU BS	Grad. Accum.	GPUs	Eff. BS	LR
10K	15	32	2	1	64	3×10^{-4}
50K	6	32	4	1	128	4×10^{-4}
100K	5	32	4	2	256	5×10^{-4}
320K	4	64	4	2	512	7×10^{-4}
640K	3	64	8	2	1024	1×10^{-3}

The experimental setups are shown in Table K.1. The learning rate is scaled according to $\sqrt{\text{effective batch size}/1024} \times 10^{-3}$. Unless otherwise specified, all models are trained under the same setting as the main experiments: we use 16-token Wikipedia sequences, extract layer-10 last-token representations, and fine-tune only the adapter while keeping the decoder frozen.

The results in Figure K.1 show a clear monotonic trend across all metrics: as the number of training examples increases, loss and perplexity decrease, while token-level and semantic reconstruction metrics improve consistently. In particular, BERTScore increases from 0.636 at 10K examples to 0.798 at 640K examples, suggesting that larger training sets help the adapter learn a more effective mapping from last-token representations to the decoder embedding space. At the same time, the marginal gains shrink at larger data scales. For example, increasing the dataset from 320K to 640K improves BERTScore only from 0.782 to 0.798 and Token F1 from 0.477 to 0.505, which is smaller than the improvements observed in earlier scaling regimes. This indicates that the data scaling curve starts to flatten at the 640K examples. Therefore, we adopt 640K as the default setting in the main experiments, as it offers a favorable balance between inversion quality and training cost.

L Instruction Examples

We include prompts for GPT-4.1-mini as below.

L.1 Scoring Instructions

Structure Score

You are given two sentences:

[GT]: `self.gt_sen`
[GEN]: `self.gen_sen`

Evaluate **Structural Frame Similarity**. Focus on whether [GEN] preserves the core grammatical structure and sentence skeleton of [GT]: same basic clause structure, same number/type of major phrases, similar syntactic relationships, consistent verb tense/aspect.

Scoring:

5: Identical structure — same words in same order
4: Nearly identical — same pattern with entity substitutions or minor reordering
3: Moderately similar — core structure maintained but notable changes (e.g., active to passive)
2: Somewhat similar — recognizable elements but significant differences
1: Minimally similar — only basic sentence type matches
0: Completely different structures

Answer: `[ANS] structure: [score]/5`

Entity Score (Clinical)

You are given two sentences:

[GT]: `self.gt_sen`
[GEN]: `self.gen_sen`

Evaluate the **Entity/Role Consistency and Plausibility** between these snippets.

Clinical Entity Type Matching Rules:

- Patient Demographics: Age (±5 years = high), same biological sex
- Symptoms / Chief Complaint: Same symptom category; penalise crossing body systems
- Diagnosis / Condition: Same condition class; similar severity
- Medications: Same drug class or mechanism
- Vital Signs / Lab Values: Numerical proximity (BP $\pm$10 mmHg, O2 sat $\pm$3%, glucose $\pm$30 mg/dL)
- Anatomical Location: Same body region or system
- Dates / Admission Timing: ±30 days = high, ±1 year = moderate, >5 years = low
- Medical Procedures: Same procedure category

Scoring Guidelines: 0–5

Answer: `[ANS] entity: [score]/5`

Entity Score (Wiki)

You are given two sentences:

[GT]: `self.gt_sen`
[GEN]: `self.gen_sen`

Evaluate **Entity Preservation**. Focus on whether [GEN] preserves key entities (people, places, organizations) from [GT]: same named entities, same key objects/concepts, equivalent entities in corresponding roles, preservation of entity relationships.

Scoring:

5: All entities preserved — all key entities from GT appear in GEN with same references
4: Nearly all preserved — minor omissions of non-critical entities or slight variations
3: Most preserved — majority of key entities maintained, some important ones missing/substituted
2: Some preserved — recognizable overlap but significant differences in key entities
1: Few preserved — minimal overlap, only generic categories match
0: No overlap — completely different entities

Answer: `[ANS] entity: [score]/5`

Topic Score (Wiki)

You are given two sentences:

[GT]: `self.gt_sen`
[GEN]: `self.gen_sen`

Evaluate **Topic Consistency**. Focus on whether [GEN] maintains the same main subject/topic as [GT]: same primary entity/concept, same domain/field, same general subject matter, maintains relevance to original topic.

Scoring:

5: Identical topic — exactly the same specific topic/entity with same focus
4: Highly similar — same main topic with slightly different aspects/perspectives
3: Related topic — closely related subjects within same domain/category
2: Loosely related — some connection but notably different topics/focuses
1: Minimally related — tangentially connected or only shares broad category
0: Unrelated — completely different subjects with no meaningful connection

Answer: `[ANS] topic: [score]/5`

Topic Score (Clinical)

You are given two clinical text snippets:

[GT]: `self.gt_sen`
[GEN]: `self.gen_sen`

Evaluate the **Topical Relevance** between these snippets.

Clinical Topic Categories: Cardiology / Pulmonology / Infectious Disease / Neurology / Gastroenterology / Endocrinology / Musculoskeletal / Mental Health / Nephrology / General & Post-op

Scoring Guidelines: 0-5

5: Same specific clinical topic (same disease/condition/care scenario)
4: Same clinical domain with minor shifts
3: Related clinical domains
2: Loosely related
1: Both clinical but completely different physiological systems
0: No meaningful clinical connection

Answer: `[ANS] topic: [score]/5`

L.2 System Prompt

System Prompt

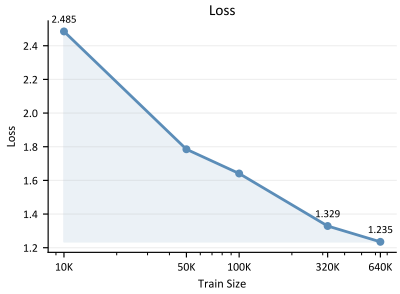
You are an AI assistant that can decode the hidden representation vector from the intermediate layer of a language model. You receive a hidden representation vector, which is the representation of an input textual sequence's final token at a fixed layer. The ask is using this representation vector to completely reveal the original text it encodes. Always produce the possible input text as exactly as you can, and avoid rephrasing it.

L.3 User Prompt Examples

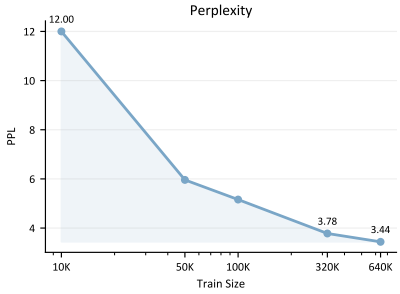
User Prompts

- What type of context is this representation most likely encoding?
- What does this embedding reveal about the input sequence?
- Describe the underlying meaning this hidden state is capturing.
- What kind of sentence or phrase could generate this vector?
- What semantic information is likely contained in this hidden representation?
- What real-world context could this representation be associated with?
- What kind of textual environment might lead to this hidden state?

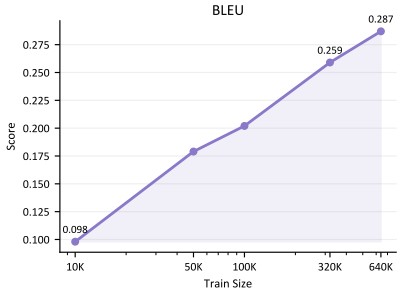
- What is the most plausible meaning encoded by this internal representation?



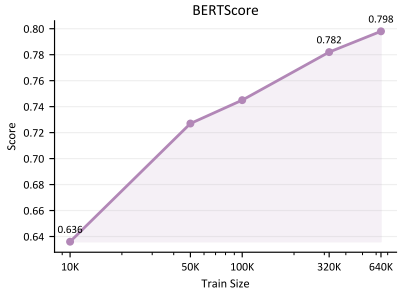
(a) Loss



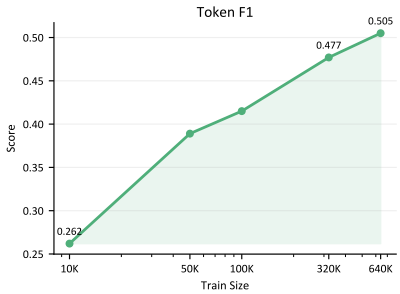
(b) Perplexity



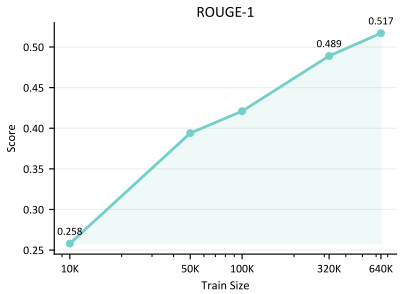
(c) BLEU



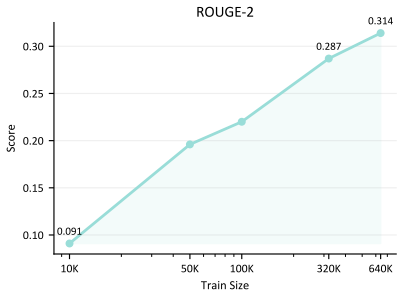
(d) BERTScore



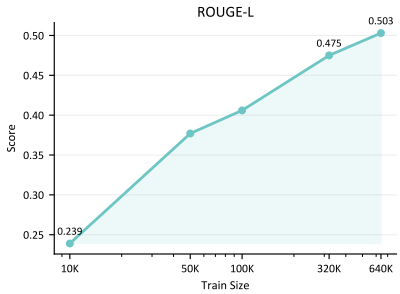
(e) Token F1



(f) ROUGE-1



(g) ROUGE-2



(h) ROUGE-L

Figure K.1: Dataset size ablation across eight metrics. Increasing the number of training examples consistently improves Rep2Text inversion performance: loss and perplexity decrease, while token-level and semantic reconstruction metrics increase. The gains remain positive from 320K to 640K examples, but become smaller, indicating diminishing returns at larger data scales.

Table L.1: Inverted examples with varying token lengths

Tokens (#)	Ground-truth Sequence	Inverted Sequence	R1	R2	RL	BS	SS	ES	TS
8	in the Centre-Val de Loire	in the Centre-Val de Loire	1	1	1	1	1	1	1
	is the second era of the Hade	is the third era of the Hade	0.86	0.67	0.86	0.99	1	0.8	1
	who enforce New Zealand's regulatory building control	the enforcement of New Zealand statutory building control	0.63	0.29	0.63	0.77	0.4	0.6	1
	Tyler, the Creator production discography\n\n	Metro Boomin production discography\n\n	0.44	0.29	0.44	0.77	0.8	1	1
	biologist \n Stanley Fields (actor) (	:\n\n John Allen (actor) (born	0.25	0	0.25	0.78	0.6	0.8	0.8
16	species within the genus Conus, these snails are predatory and venomous.	species within the genus Conus, these snails are predatory and venomous.	1	1	1	1	1	1	1
	List of shipwrecks in September 1842\n\nThe list of ship	List of shipwrecks in January 1840\n\nThe list of ship	0.8	0.67	0.8	0.99	0.8	0.8	1
	2017 NCAA Division I Softball Tournament\n\nThe 2017 NCAA Division I	2018 NCAA Division I Women's Soccer Tournament\n\nThe 2018 NCAA Division	0.61	0.38	0.61	0.88	0.8	0.6	0.8
	Rob James\n\nRob James may refer to:\n\nRob James (singer) (	Mark Jones\n\nMark Jones may refer to:\n\nMark Jones (singer) (	0.4	0.22	0.4	0.96	0.8	1	1
	Qi Yuwu, Julian Hee, Jeanette Aw, Felicia Chin,	, Pierre Png, Chen Hanwei, Felicia Chin, Fann Wong	0.25	0.14	0.25	0.82	0.8	1	1
32	1825 in Wales\n\nThis article is about the particular significance of the year 1825 to Wales and its people.\n\nIncumbents\nPrince of Wales \u2013	1840 in Wales\n\nThis article is about the particular significance of the year 1840 to Wales and its people.\n\nIncumbents\nPrince of Wales \u2013	0.91	0.86	0.91	0.99	0.8	0.8	1
	List of European Championships records in swimming\n\nThe European Championships records in swimming are the fastest times ever swum in European Swimming Championships' events.\n\nLong course (50	List of European records in swimming\n\nThe following are the current European records in swimming, as recognized by LEN.\n\nLong course (50 m pool)\n\nShort course (	0.6	0.39	0.53	0.75	0.6	0.8	1
	Institute of Higher National Defence Studies.\n\nThen, he went to the USA for a Master in International Business Management at the University of Florida. He began his career in	.\n\nHe then studied at the Institut d'\u00e9tudes politiques de Paris and at the \u00c9cole nationale d'administration (ENA). He started his career in	0.42	0.12	0.3	0.71	0.2	0.6	0.8
	William Robertson Nicoll\n\nSir William Robertson Nicoll CH LLD (10 October 1851 \u2013 4 May 1923) was a Scottish Free Church	John Duncan (minister)\n\nJohn Duncan (1 May 1845 \u2013 6 March 1914), also known as J. Duncan, was a Free	0.21	0.05	0.21	0.74	0.8	0.8	0.8
	Alessandro Guiccioli\n\nAlessandro Guiccioli (March 5, 1843 \u2013 October 3, 1922)	Gustave de Beaumont\n\nGustave de Beaumont (March 1, 1840 \u2013 March 2, 1921) was	0.09	0	0.09	0.83	0.6	0.4	0.8

64	Rush Hour 2\n\nRush Hour 2 is a 2001 American action comedy film directed by Brett Ratner and written by Jeff Nathanson, based on the characters created by Ross LaManna. A sequel to Rush Hour, it is the second installment in the Rush Hour series and stars Jackie Chan,	Rush Hour 2\n\nRush Hour 2 is a 2001 American buddy cop action comedy film directed by Brett Ratner and written by Jeff Nathanson. It is the sequel to the 1998 film Rush Hour and the second installment in the Rush Hour film series. The film stars Jackie Chan,	0.84	0.67	0.78	0.91	0.8	1	1
	hull length of, a beam of, a height of, and a draught of. The submarine was powered by two Germaniawerft F46 four-stroke, six-cylinder supercharged diesel engines producing a total of for use while surfaced, two AEG GU 460/8\u201327 double-	. The U-boat had a displacement of when at the surface and while submerged. The U-boat had a total length of, a pressure hull length of, a beam of, a height of, and a draught of. The submarine was powered by two Germaniawerft F46 six-	0.61	0.48	0.49	0.8	0.4	1	1
	the same rights of audience as members of the Bar of Northern Ireland.\n\nThe Advocate General was created as a separate office upon the devolution of policing and justice powers to the Northern Ireland Assembly on 12 April 2010.\n\nUnlike the Advocate General for Scotland, the position is not supported by a distinct government department.	the Scottish Parliament. The office was created in 1999, and is the equivalent of the Parliamentary Under-Secretary of State in the United Kingdom Government.\n\nThe office is not a ministerial post, and the holder is not a member of the Scottish Government. Responsibility for the office is held by the Scottish Secretary.	0.41	0.08	0.24	0.66	0.4	0.4	0.8
	producer Thom Wilson and released in 1982 as catalog number VIRUS 10. Singer Jack Grisham credited himself as Jack Ladoga on the sleeve, following a tradition of using a different pseudonym on each release both to confuse audiences and to hide his true identity from the police. Drummer Todd Barnes credited himself	the band's first album, and the first to feature the band's new lineup. The band members used pseudonyms on the album, with the exception of guitarist and vocalist John \"Baz\" Bascaran, who used his real name because he was the only member of the band with a driver's license. Drum	0.21	0.04	0.15	0.59	0.8	0.6	0.8
	Linux kernel.\n\nXC3018\nIt is a variant that only supports analog reception and DVB-T digital reception.\n\nTechnical specification\nOutline Dimensions: 7 x 7 x 0.85\u00a0mm\nSupply Voltage (DC): 1.8V, 3.3V\nSystem setting time:	2010.\n\nSpecifications\nFrequency: 2.4\u00a0GHz\nData rate: 1, 2, 5.5, 11\u00a0Mbps\nModulation: DSSS\nPower consumption: 0.1\u00a0W\nOperating temperature: 0\u00a0\u00b0C to 70\u00a0\u00b0C\n	0.09	0	0.09	0.62	0	0	0.8

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state the paper's main scope and contributions: studying how much text can be recovered from a single last-token representation and proposing Rep2Text to perform this inversion. These claims are supported by the experimental results in Section 4, including cross-model inversion, sequence-length analysis, layer-wise analysis, and comparison with Vec2Text on both in-distribution and out-of-distribution data.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sections 4.5, 4.7, and 5 discuss the drop in recovery for longer inputs, the LLM-judge bias, and the imperfect recovery of clinical attributes.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results or proofs.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3 and 4 and the appendices describe the architecture, training setup, datasets, metrics, and baselines needed to reproduce the main results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the code and environment files at `https://github.com/hy-zhao23/Rep2Text`, and the trained adapter checkpoints at `https://huggingface.co/adfjasigageadfd/rep2text-checkpoints`. The README describes how to rebuild the training data and run each experiment.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 and Appendix I specify the datasets, splits, model choices, hyperparameters, training details, and evaluation metrics.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report mean scores and standard deviations for the main comparison, but do not run multiple seeds or formal significance tests due to computational cost.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.1 reports the GPU type and approximate training time, with additional implementation details in Appendix I.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies privacy and interpretability risks using public datasets and models, without deployment on private user data.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 5 discusses the privacy risk of exposing intermediate representations and the need to protect them.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We release only adapters trained on Wikipedia text. We do not release tools that target private user data.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the existing datasets, models, and baselines used in the experiments.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the code and the trained adapter checkpoints. The repository README documents their usage and the license.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Appendix C describes the volunteer annotators, sampled data, and rating protocol for the human sanity check. No monetary compensation was provided because the annotators were volunteers.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The human sanity check did not collect personal or sensitive information from annotators.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are used as decoding models and for LLM-as-a-judge evaluation, as described in Sections 3 and 4 and the appendices.