TRAIL: LoRA AS TRACEABLE PARAMETRIC MEMORY WITH TRACE TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

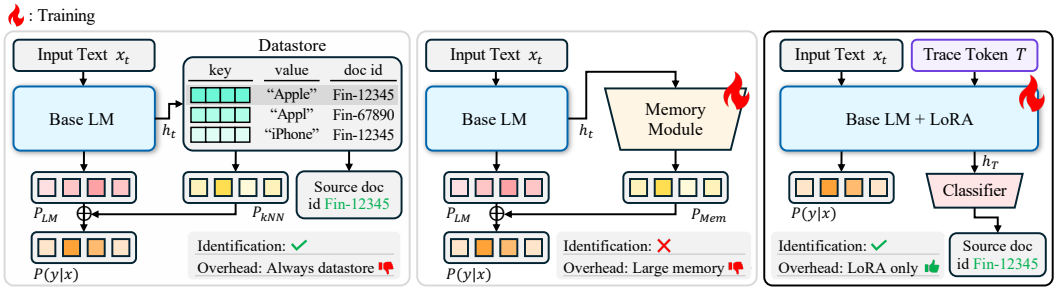
Deploying large language models (LLMs) in high-stakes domains requires two long-competing capabilities: efficient domain-adaptive fine-tuning and traceability of each prediction back to its supporting training instance. Parametric LMs adapt efficiently but offer no per-instance index, while non-parametric kNN-LMs preserve instance-level traceability through retrieval but with substantial inference cost. Recent parametric imitations of kNN-LM remove the datastore but discard its traceability by construction. We propose **TRAIL** (**Tra**ceable Parametr**i**c Memory with **L**oRA and Trace Tokens), which recovers the instance-level traceability of kNN-LM within an efficient parametric module. On top of a frozen LoRA-adapted LM, TRAIL introduces a small set of learnable special tokens, called *trace tokens*, trained contrastively so that the hidden state of each prediction's trace token identifies its source training instance. On finance domain data, TRAIL attains 18–33% higher identification accuracy with 60–70% lower test perplexity compared to parametric memory baselines. This is achieved with $2\times$ fewer trainable parameters, $60\times$ smaller disk footprint, and $242\times$ faster inference than kNN-LM. Counterfactual retraining shows that the identified documents are causally responsible, so TRAIL offers a foundation for accountable deployment of domain-adapted LLMs, supporting per-instance attribution, copyright compliance, and error diagnosis.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed across high-stakes domains, such as scientific research, legal practice, medical decision support, and financial analysis (Lu et al., 2024; Barale et al., 2023a; Ma et al., 2025; Wu et al., 2023). Such deployment, typically realized by fine-tuning a base LM on a domain-specific corpus, raises two requirements that need to be jointly satisfied. First, models should efficiently adapt to each domain's specialized knowledge. Second, each prediction needs to remain traceable back to the specific training instance that supports it: errors should be diagnosable (Huang et al., 2025), copyrighted or sensitive material should be removable (Min et al., 2023), and accountability should be enforceable (Bommasani et al., 2024). Standard parametric LMs absorb training-data signal into distributed weights, leaving no explicit per-instance index that can be directly queried. Achieving efficient adaptation and traceability to specific training instances together has thus remained an open challenge (Asai et al., 2024).

The k-nearest-neighbors language model (kNN-LM) (Khandelwal et al., 2019) addresses this by retrieving training tokens from an external datastore and forming a non-parametric distribution from these retrievals. Because each retrieved instance can carry its source identity, kNN-LM also enables *instance-level traceability*, a property that could be utilized for opt-out and attribution (Min et al., 2023). However, kNN-LM faces severe efficiency limitations: the datastore can exceed the base model size by orders of magnitude, and nearest-neighbor search at every decoding step adds substantial latency that even state-of-the-art acceleration cannot fully eliminate (Johnson et al., 2019; He et al., 2021). Recent works remove this datastore by imitating kNN-LM's retrieval-based next-token distribution with a parametric module (Wei et al., 2026; Cao et al., 2025), but at two costs: the auxiliary module is itself substantial in size and must be kept in memory at inference, and more fundamentally, by removing the retrieval step these methods discard kNN-LM's instance-level traceability by construction. This raises the central research question of our work: *can we recover kNN-LM's instance-level traceability within an efficient parametric module?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Non-parametric (b) Parametric Imitations (c) Ours (LoRA + Trace Token)

Figure 1: **Three approaches to enhance (parametric) language modeling, with respect to instance-level traceability. (a) Nonparametric LM (kNN-LM (Khandelwal et al., 2019))**: retrieval over a per-token datastore enables token-level traceability but at substantial inference cost. **(b) Parametric imitations (Wei et al., 2026; Cao et al., 2025)**: a compact module imitates the non-parametric distribution without a datastore, but traceability is lost. **(c) TRAIL (Ours)**: a LoRA-adapted LM with learnable trace tokens recovers traceability via a linear readout, without a datastore at inference.

Contribution. We propose **TRAIL**, a framework that constructs **tra**ceable parametr**i**c memory with **L**oRA (Hu et al., 2022), recovering kNN-LM's instance-level traceability without any inference-time datastore. Our approach is motivated by the hypothesis that LoRA fine-tuning already encodes instance-level identity into the adapted LM's hidden states, though only implicitly (empirically confirmed in § 4). To make this latent information explicit and traceable, TRAIL introduces a small set of learnable special tokens, which we call *trace tokens*, on top of a frozen LoRA-adapted LM. Only the trace token embeddings are trained, with a contrastive objective that pushes apart the trace token representations of different training instances. At inference, trace tokens are interleaved with regular tokens in a single forward pass, leaving the LM's next-token distribution unchanged while a linear readout function maps each trace token's hidden state to its source training instance.

In our experiments, we show that TRAIL consistently outperforms recent parametric memory baselines (Wei et al., 2026; Cao et al., 2025) on both language-modeling adaptation and instance-level identification, across three LLM families (Qwen2.5 (Qwen et al., 2025), Llama3.2 (Grattafiori et al., 2024), Gemma3 (Team et al., 2025)) and three specialized domains (biomedicine, finance, law) (Cao, Jie, 2025). For example, TRAIL attains 60%–70% lower perplexity than state-of-the-art parametric imitation methods while achieving 18%–33% higher identification accuracy on finance domain. Importantly, TRAIL achieves these gains with 2× fewer trainable parameters than parametric memory baselines and 242× faster inference than kNN-LM. Counterfactual retraining further shows that the identified documents are causally responsible for the corresponding predictions. Following kNN-LM, our framework attributes each token and aggregates the results for instance-level operations such as opt-out. Overall, our contributions can be summarized as follows:

- We formulate *datastore-free instance-level traceability*: recovering the per-instance attribution provided by kNN-LM within an efficient parametric module.
- We propose TRAIL, which adds learnable *trace tokens* to a frozen LoRA-adapted LM to make its latent instance identity explicit and linearly readable. A custom attention mask lets trace tokens read the regular-token context without influencing it, so the LM's next-token distribution is preserved.
- Across three LLM families and three domains, TRAIL surpasses parametric memory baselines on both adaptation and identification, and adds 9% Top-1 on average over the identical LoRA backbone at 0.35% overhead. The identified documents are causally responsible: removing them raises perplexity by up to ×20.96, whereas removing an equal number at random barely moves it.

2 Preliminary

2.1 Non-parametric language models and parametric imitations

Given an input sequence of tokens $\mathbf{x}$, an autoregressive language model (LM) estimates $p(y \mid \mathbf{x})$, the distribution over the next token y. A standard *parametric* LM $p_\theta(y \mid \mathbf{x})$ encodes its predictive knowledge entirely in fixed model parameters θ, learned during training and used unchanged at inference. In contrast, a *non-parametric* LM estimates the next token distribution by retrieving and

aggregating relevant examples from an external datastore (Weston et al., 2014; Grave et al., 2016; Yogatama et al., 2021; Min et al., 2022); here, we focus on the kNN-LM (Khandelwal et al., 2019) as the canonical example of this paradigm. Let $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be a corpus of N instances (*e.g.*, documents), where each instance is a sequence of tokens $\mathbf{x} = (x_1, ..., x_\ell)$.[1] Throughout, i indexes instances and t indexes token positions. For each token $x_{i,t}$ in instance $\mathbf{x}_i \in \mathcal{D}$, the datastore stores a (key, value, index) triple $\big(h_{i,t}, x_{i,t}, i\big)$ where $h_{i,t} \in \mathbb{R}^d$ is the hidden state of the LM in a chosen layer that is used to predict $x_{i,t}$ from the preceding context $\mathbf{x}_{i,<t} = (x_{i,1}, ..., x_{i,t-1})$. At inference, given a test query $\widehat{\mathbf{x}}$ and its hidden state $\widehat{h}$, kNN-LM retrieves the set $\mathcal{N}_k$ of k datastore entries whose keys are nearest to $\widehat{h}$ under a distance function $\delta(\cdot, \cdot)$, and forms the non-parametric distribution:

$$p_{\text{kNN}}(y \mid \widehat{\mathbf{x}}) \;\propto \sum_{(h_{i,t},\, x_{i,t},\, i) \,\in\, \mathcal{N}_k} \mathbb{1}\{y = x_{i,t}\} \cdot \exp\big(-\delta(h_{i,t},\, \widehat{h})\big). \tag{1}$$

The non-parametric LM is complementary to parametric LM, as it can be easily combined via simple interpolation with a hyperparameter λ ($0 \le \lambda \le 1$) (Khandelwal et al., 2019): $p(y \mid \mathbf{x}) = \lambda\, p_{\text{kNN}}(y \mid \mathbf{x}) + (1 - \lambda)\, p_\theta(y \mid \mathbf{x})$ (Figure 1(a)). More importantly, since each retrieved entry in $\mathcal{N}_k$ carries its source index, retrieval simultaneously produces a traceability set $\mathcal{S}(\mathcal{N}_k) = \{i \mid (h_{i,t}, x_{i,t}, i) \in \mathcal{N}_k\}$ that identifies training instances that contributed to the prediction. In this work, we say that a model exhibits *instance-level traceability* if its prediction mechanism admits a deterministic mapping ρ from each (prefix context, predicted next token) pair $(\mathbf{x}, y)$ to a finite set of training-instance indices that supported the prediction, where $\rho(\mathbf{x}, y) = \mathcal{S}(\mathcal{N}_k)$ for kNN-LM.

Efficiency cost and parametric imitations. Non-parametric LM provides strong domain adaptation performance through retrieval over the datastore (Min et al., 2023; Khandelwal et al., 2019; 2020). However, this comes at substantial computational cost: with N training instances, ℓ tokens per instance, and key dimension d, the datastore footprint scales as $\mathcal{O}(N \cdot \ell \cdot d)$, which can exceed the size of the LM itself at moderate scale (Table 2). Moreover, nearest-neighbor search at every decoding step adds substantial latency that approximate indexing cannot fully eliminate (He et al., 2021; Alon et al., 2022). To remove these costs, recent works, including MLP Memory (MLPMem) (Wei et al., 2026) and Memory Decoder (MemDec) (Cao et al., 2025), imitate the non-parametric distribution with a parametric module to eliminate inference-time datastore dependence (Figure 1(b)). These methods restore inference efficiency, but at a price: by imitating what kNN-LM predicts while discarding the record of why, they leave the traceability set $\mathcal{S}(\mathcal{N}_k)$ never instantiated and the mapping ρ undefined. This is a structural consequence of how parametric LMs absorb training-data signal into distributed weights (Aharoni & Goldberg, 2020). Therefore, this work aims to recover both strong adaptation and instance-level traceability of kNN-LM for each prediction $(\mathbf{x}, y)$ within a parametric module.

2.2 Problem Setup: Datastore-Free Instance-Level Traceability

Following kNN-LM's per-token retrieval, we apply the traceability mapping ρ (§2.1) at per-token granularity: given a trained LM and its fine-tuning corpus $\mathcal{D}_{\text{train}}$, the goal is to identify, for each test token's prediction, the training instance in $\mathcal{D}_{\text{train}}$ most likely to have contributed to it. Our scope thus aligns with practical domain-adaptation deployment, where the fine-tuning corpus carries the domain-specific knowledge for which traceability is most valuable for error diagnosis, per-document opt-out, and accountability. A qualitative example is shown in Figure 4, 9, 10.

To evaluate whether a parametric model recovers this traceability, we adopt the kNN with k=1 (the nearest training token; kNN-1 hereafter) as the evaluation reference [2] for our raw, unannotated training corpus: its retrieved source index forms $\rho(\mathbf{x}, y) = S(N_1)$ (§2.1), so agreement with it measures recovery of kNN-LM's instance-level traceability. We test robustness to this choice by rescoring all methods against TRAK (Park et al., 2023), a gradient-based attribution signal that does not rely on representational proximity (Appendix E.1.1), and assess whether kNN-1 and TRAIL attributions are causally responsible by removing their identified documents and retraining LoRA (§4.3).

Let $\mathcal{H} = \{h_{i,t} \mid x_{i,t} \in \mathbf{x}_i,\, \mathbf{x}_i \in \mathcal{D}_{\text{train}}\}$ be the training-token hidden states at a chosen layer of the *base LM*, and let $\widehat{h}_t$ be the hidden state of the test prefix $\widehat{\mathbf{x}}_{<t}$ at the same layer, used to predict $\widehat{x}_t$. Then, its reference source s_t is defined as the source instance of $\widehat{h}_t$'s nearest training token under

[1] For notational convenience, here we assume all instances have the same length ℓ.

[2] Training data attribution serves here only as an evaluation reference, not a proposed method; see Appendix A.

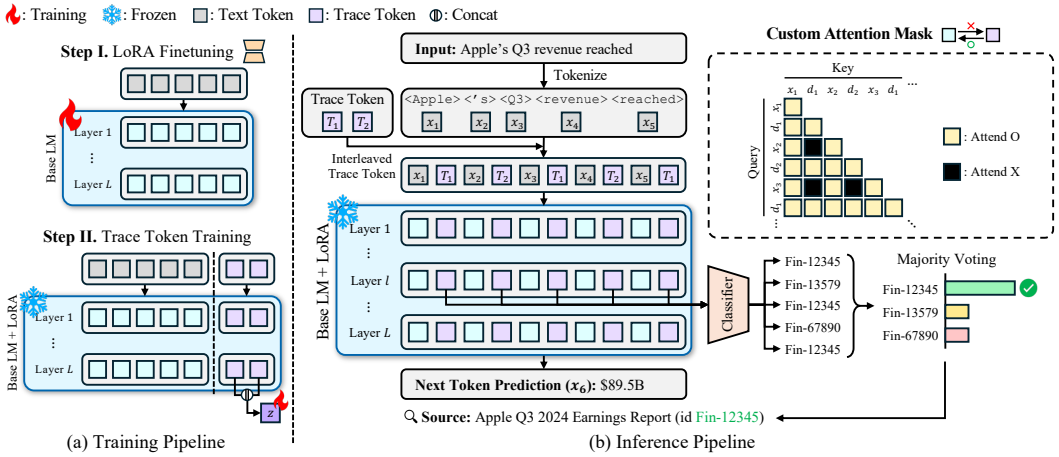


Figure 2: **Overview of TRAIL. (a) Training (§ 3.1)**: conventional fine-tuning with LoRA is conducted. Then, K trace tokens are trained with SimCSE loss on top of the frozen LoRA-adapted LM. **(b) Inference (§ 3.2)**: trace tokens are interleaved with regular tokens under a custom attention mask (top right), and a linear readout on the trace-token hidden states predicts the source document.

cosine similarity $\cos(u, v) = u^\top v/(\|u\|\|v\|)$:

$$s_t = i^*, \quad \text{where} \quad (i^*, t^*) = \arg\max_{h_{i,t'} \in \mathcal{H}} \cos(\widehat{h}_t,\, h_{i,t'}). \tag{2}$$

The instance-level reference is the majority vote of these per-token references, $S^{\text{ref}} = \arg\max_{\mathbf{x}_i \in \mathcal{D}_{\text{train}}} \sum_{t=1}^{\ell} \mathbb{1}\{s_t = i\}$, aligning with per-document operations such as opt-out. We adopt instance-level identification as our primary metric and report token-level results in Appendix E.4.

3 Traceable Parametric Memory with LoRA and Trace Tokens

3.1 Training

LoRA-adapted backbone. TRAIL first adapts a base LM f_{base} to $\mathcal{D}_{\text{train}}$ via standard LoRA fine-tuning (Hu et al., 2022) (Figure 2(a), Step I) and uses the result as the frozen backbone f. Following §2.1, $h_t \in \mathbb{R}^d$ denotes the hidden state of f at the chosen layer for position t of input $\mathbf{x}$; we additionally write $h_t^* \in \mathbb{R}^d$ for the hidden state at the last layer of f for the same position, used during contrastive training of trace tokens (Eq. 4).

Trace tokens. While LoRA fine-tuning encodes instance-specific signal into the LM's hidden states, this signal is distributed across regular tokens that aggregate context for next-token prediction rather than instance identification (Figure 3 (a)). To extract this signal, we introduce K *trace tokens* $\mathcal{T}_1, \ldots, \mathcal{T}_K$ (K=2 throughout our experiments), each associated with its own learnable embedding $\mathbf{e}_k \in \mathbb{R}^d$ ($k = 1, \ldots, K$), treated as a vocabulary extension disjoint from regular tokens. Building on prior uses of learnable auxiliary tokens (Li et al., 2023; Goyal et al., 2023), our trace tokens are tailored to produce per-instance representations readable by a linear classifier on a frozen LoRA-adapted LM.

Contrastive learning of trace tokens. With f held frozen, we append all K trace tokens to each training instance $\mathbf{x}_i \in \mathcal{D}_{\text{train}}$ of length ℓ, forming the augmented input

$$\bar{\mathbf{x}}_i = (x_{i,1}, \ldots, x_{i,\ell}, \mathcal{T}_1, \ldots, \mathcal{T}_K). \tag{3}$$

We place trace tokens at the end rather than interleaving, since a single trailing readout suffices for the contrastive objective and avoids scaling with instance length. We train the trace token embeddings $\{\mathbf{e}_k\}_{k=1}^K$ with an InfoNCE objective (Oord et al., 2018). Following SimCSE (Gao et al., 2021), two forward passes of $\bar{\mathbf{x}}_i$ through f with independent dropout masks yield a positive pair (z_i, z_i'), where

$$z_i = \left[\, h^*_{i,\ell+1};\, \ldots;\, h^*_{i,\ell+K} \,\right] \in \mathbb{R}^{Kd} \tag{4}$$

concatenates the last-layer hidden states h^* at the K trace positions of $\bar{\mathbf{x}}_i$. The InfoNCE objective over a batch of B instances, with the other $B-1$ instances serving as in-batch negatives, is

$$\mathcal{L} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\cos(z_i, z_i')/\tau)}{\sum_{j=1}^{B} \exp(\cos(z_i, z_j')/\tau)}, \tag{5}$$

where τ is a temperature hyperparameter. Because f is frozen, only the embeddings of trace tokens $\{\mathbf{e}_k\}_{k=1}^{K}$ are updated. After training, trace tokens encode source-instance information by aggregating the preceding context through self-attention. Full training details are in Appendix B.1.

3.2 Inference

Interleaved input construction. For token-level traceability (§2.2), each regular token must be followed by trace tokens that have seen the same preceding prefix, so their hidden states can identify the regular token's source. To this end, at inference (Figure 2(b)), we insert $\mathcal{T}_1, \ldots, \mathcal{T}_K$ in round-robin order after every token of the test sequence $\widehat{\mathbf{x}} = (\widehat{x}_1, \ldots, \widehat{x}_\ell)$, forming the interleaved sequence $\widetilde{\mathbf{x}}$[3]:

$$\widetilde{\mathbf{x}} = \big(\widehat{x}_1,\, \mathcal{T}_{\pi(1)},\, \widehat{x}_2,\, \mathcal{T}_{\pi(2)},\, \ldots,\, \widehat{x}_\ell,\, \mathcal{T}_{\pi(\ell)}\big), \qquad \pi(t) = ((t-1) \bmod K) + 1. \tag{6}$$

Custom attention mask. With $\widetilde{\mathbf{x}}$, attention of f is constrained by a custom mask $\mathbf{A} \in \{0,1\}^{2\ell \times 2\ell}$:

$$\mathbf{A}[q,k] = \begin{cases} 0 & q \text{ is odd (regular) and } k \text{ is even (trace),} \\ \mathbb{1}\{k \le q\} & \text{otherwise.} \end{cases} \tag{7}$$

which (i) preserves causal regular-to-regular attention, (ii) lets each trace token aggregate the preceding context (both regular and trace), and (iii) blocks any regular-to-trace attention (Figure 2(b), top right). Crucially, (iii) leaves regular-token hidden states identical to those without trace tokens, so any output computed from them, such as the next-token logits in autoregressive generation, is unchanged. This property makes TRAIL *single-pass*: a single forward pass yields both the next-token logits at regular positions and the per-instance signal at trace positions.

Tracing back to source instances. To trace predictions back to their source instances, we train a linear classifier $g_\theta : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{D}_{\text{train}}|}$ on trace token hidden states extracted from $\mathcal{D}_{\text{train}}$, where the i-th output coordinate $g_\theta(\cdot)_i$ scores training instance i (full setup in § 4.1). At test time, a single forward pass on $\widetilde{\mathbf{x}}$ yields chosen-layer hidden states h_{2t} at the trace positions, $t = 1, ..., \ell$. At each trace position, g_θ yields a per-token prediction $\hat{s}_t = \arg\max_{\mathbf{x}_i \in \mathcal{D}_{\text{train}}} g_\theta(h_{2t})_i$, which we aggregate by majority vote into the instance-level prediction as follows:

$$\hat{S} = \arg \max_{\mathbf{x}_i \in \mathcal{D}_{\text{train}}} \sum_{t=1}^{\ell} \mathbb{1}\{\hat{s}_t = i\}. \tag{8}$$

At each t of $\widehat{\mathbf{x}}$, this per-token prediction $\hat{s}_t$ instantiates ρ (§2.1) applied to the prefix $\widehat{\mathbf{x}}_{<t}$ and predicted token $\widehat{x}_t$; majority voting yields the instance-level prediction $\hat{S}$, recovering kNN-LM's instance-level traceability without a datastore. Figure 4 shows examples; Algorithm 1 (§4.1) details the pipeline.

4 Experiments

In this section, we present our experimental results to answer the following research questions:

- **RQ1**: Can TRAIL recover instance-level traceability with strong domain adaptation? (Table 1)
- **RQ2**: How efficient is TRAIL relative to parametric memory baselines and kNN-LM? (Table 2)
- **RQ3**: Are the identified instances causally responsible, or merely similar? (Table 3)
- **RQ4**: Do trace tokens make latent identity explicit and linearly readable? (Figure 3)
- **RQ5**: How well does TRAIL scale across model size? (Table 4)

4.1 Experimental setup

Models and domains. We evaluate on three pretrained LMs of comparable scale (0.5B–1B parameters): Qwen2.5-0.5B (Qwen et al., 2025), Llama3.2-1B (Grattafiori et al., 2024), and Gemma3-1B (Team et al., 2025). Following prior work (Cao et al., 2025; Cao, Jie, 2025), we adopt three domains: Biomedical (Bio) (Johnson et al., 2016), Finance (Fin) (Maulik, Jyani, 2024), and Law (Barale et al., 2023b). Each contains 37K training documents, with Bio and Law subsampled to match Fin, and 100/250/50 test documents for Bio/Fin/Law.

[3]We duplicate the positional embedding from a regular token to the right next trace token.

Algorithm 1 Identification Pipeline of TRAIL.

Input: Lora-adapted and frozen LM f, base LM f_{base}, chosen layer L, training/test corpus $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$
Ensure: Accuracy $\texttt{acc}$

1: Compute training-side hidden states $h_{i,t} \leftarrow f^{(L)}(\mathbf{x}_i)_t$ and reference $h^{\text{ref}}_{i,t} \leftarrow f^{(L)}_{\text{base}}(\mathbf{x}_i)_t$ for all $x_{i,t} \in \mathbf{x}_i, \mathbf{x}_i \in \mathcal{D}_{\text{train}}$.
2: Compute test-side hidden states $\widehat{h}_t \leftarrow f^{(L)}(\widehat{\mathbf{x}}_j)_t$ and reference $\widehat{h}^{\text{ref}}_t \leftarrow f^{(L)}_{\text{base}}(\widehat{\mathbf{x}}_j)_t$ for all $\widehat{x}_{j,t} \in \widehat{\mathbf{x}}_j, \widehat{\mathbf{x}}_j \in \mathcal{D}_{\text{test}}$.
3: Train linear classifier g_θ on $\{(h_{i,t}, i) : h_{i,t} \in \mathcal{H}\}$.
4: **for** each test instance $\widehat{\mathbf{x}}_j \in \mathcal{D}_{\text{test}}$ **do**
5: Constructing reference label S^{ref}_j via token-level kNN ($k = 1$):
6: $s_t \leftarrow \arg\max_i \max_{t'} \cos(\widehat{h}^{\text{ref}}_t, h^{\text{ref}}_{i,t'})$
7: $S^{\text{ref}}_j \leftarrow \arg\max_i \sum_t \mathbb{1}\{s_t = i\}$
8: Constructing prediction $\hat{S}_j$ with hidden states and trained linear classifier:
9: $\hat{s}_t \leftarrow \arg\max_i g_\theta(\widehat{h}_t)_i$
10: $\hat{S}_j \leftarrow \arg\max_i \sum_t \mathbb{1}\{\hat{s}_t = i\}$
11: **end for**
12: **return** $\texttt{acc} \leftarrow \text{mean}_{\widehat{\mathbf{x}}_j \in \mathcal{D}_{\text{test}}} \mathbb{1}\{\hat{S}_j = S^{\text{ref}}_j\}$.

Baselines. We compare against four approaches: (i) *Base*, the pretrained LM without fine-tuning; (ii) MLP Memory (Wei et al., 2026); (iii) Memory Decoder (Cao et al., 2025); and (iv) *LoRA* (Hu et al., 2022), configured with rank r=512, α=1024, applied to query, key, value, and MLP layers. For MLP Memory and Memory Decoder, we follow each official training recipe to obtain a domain-specialized module that imitates the non-parametric teacher distributions; although none is designed for instance-level identification, all produce per-token hidden states over the training corpus, enabling identical evaluation under our identification pipeline (Algorithm 1). For TRAIL, we add K=2 trace tokens on top of the LoRA backbone, leaving only the trace token embeddings (*e.g.*, 1,792 parameters for Qwen2.5-0.5B) trainable. Full training schedules, including ablations on K, trace token training, and inference formats, are in Appendix B.1 and Appendix C.2.

Identification pipeline. We use a single shared pipeline to evaluate instance-level identification across all methods (Algorithm 1). For each method, the prediction comes from a linear classifier g_θ trained on token-level hidden states h_t extracted from the method's frozen identification backbone (trace-token positions for ours, regular-token positions for baselines) at the best-performing layer for each model, which falls near 62.5% (full layer ablation in Appendix C.4). Then, instance-level prediction $\hat{S}$ is made by the majority vote of its per-token predictions. The reference is computed independently from the base LM (§ 2.2): the token-level kNN with $k = 1$ reference s_t maps each test token to its nearest training token under cosine similarity (Eq. 2), and the instance-level reference S^{ref} is the majority vote of these per-token assignments. Accuracy ($\texttt{acc}$) is the fraction of test documents whose prediction matches the reference (Eq. 9); token-level results are reported in Appendix E.4.

$$\texttt{acc} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{\widehat{\mathbf{x}}_j \in \mathcal{D}_{\text{test}}} \mathbb{1}\{\hat{S}_j = S^{\text{ref}}_j\}. \tag{9}$$

Evaluation metrics. We report three complementary metrics:

- **Perplexity (PPL)** on test documents, measuring language modeling quality and evaluating the domain adaptation performance:

$$\text{PPL}(\widehat{\mathbf{x}}) = \exp\left(-\frac{1}{\ell}\sum_{t=1}^{\ell} \log p(\widehat{x}_t \mid \widehat{\mathbf{x}}_{<t})\right). \tag{10}$$

- **Top-1 identification accuracy (Top-1)**: the fraction of test documents whose instance-level prediction $\hat{S}$ matches the kNN with k=1 reference S^{ref} (Eq. 9). The prediction and reference are both defined as in the identification pipeline above.
- **Soft identification accuracy** (**Soft**, τ_s=0.9): the fraction of test instances whose embedding cosine similarity between the prediction $\hat{S}$ and the reference S^{ref} exceeds the threshold τ_s. This metric complements Top-1 by allowing semantically similar source predictions to be counted as correct:

Table 1: **Main results on traceability and domain adaptation.** Three base LMs and three fine-tuning domains are evaluated. PPL (Perplexity) measures language modeling on held-out test documents; Top-1 (%) and Soft (%) measure instance-level identification against the kNN with k=1 reference (Eq. 2). All methods use a linear classifier at the per-model best layer. LoRA and TRAIL are the best on PPL, and TRAIL leads on Top-1 and Soft across all model-domain cells. Best in **bold**.

Model	Method	Finance			Biomedical			Law		
		PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑
Qwen2.5-0.5B	Base	15.92	57.6	62.4	17.04	67.0	69.0	9.46	50.0	56.0
	MLPMem	8.90	58.8	64.0	4.08	67.0	70.0	4.43	52.0	60.0
	MemDec	3.88	65.6	68.8	3.74	67.0	73.0	4.55	54.0	62.0
	LoRA	**1.25**	73.2	80.8	**3.23**	68.0	75.0	**3.23**	60.0	78.0
	TRAIL (Ours)	**1.25**	**77.2**	**86.0**	**3.23**	**72.0**	**78.0**	**3.23**	**74.0**	**86.0**
Llama3.2-1B	Base	11.98	40.8	46.0	12.80	25.0	32.0	8.22	30.0	36.0
	MLPMem	8.02	41.2	48.0	6.85	29.0	35.0	4.50	24.0	34.0
	MemDec	7.86	66.8	74.4	7.45	65.0	67.0	7.93	64.0	72.0
	LoRA	**3.59**	69.6	80.8	**3.57**	67.0	74.0	**1.93**	70.0	84.0
	TRAIL (Ours)	**3.59**	**85.2**	**94.4**	**3.57**	**70.0**	**77.0**	**1.93**	**78.0**	**90.0**
Gemma3-1B	Base	10.32	32.0	38.4	18.61	27.0	32.0	7.38	28.0	34.0
	MLPMem	7.93	32.0	40.0	4.41	29.0	34.0	4.09	28.0	36.0
	MemDec	6.52	46.0	56.0	4.94	36.0	44.0	3.53	48.0	58.0
	LoRA	**2.50**	59.2	66.4	**3.54**	45.0	54.0	**1.74**	64.0	76.0
	TRAIL (Ours)	**2.50**	**69.2**	**78.4**	**3.54**	**62.0**	**70.0**	**1.74**	**68.0**	**84.0**

Table 2: **Main results on efficiency.** Per-epoch training and inference costs on Fin 37K with Qwen2.5-0.5B. TTFT is time-to-first-token, latency is the per-token time of a teacher-forced forward pass at batch size 1, and disk is in fp16. For kNN-LM, train time is datastore construction. TRAIL adds the least overhead over its LoRA backbone across all five metrics.

Method	Train time ↓ (h)	TTFT ↓ (ms)	Latency ↓ (ms/tok)	Parameters ↓ (M)	Disk ↓ (MB)
Base	—	22.0	0.028	0	0
MLPMem	5.1	43.5	0.057	493.8	988
MemDec	4.7	43.6	0.073	493.8	988
LoRA	0.4	25.1	0.029	259.5	519
TRAIL (Ours)	0.5	25.0	0.043	260.4	521
kNN-LM	1.8	229.7	10.43	0	31,744

$$\texttt{soft_acc}(\tau_s) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{\hat{\mathbf{x}}_j \in \mathcal{D}_{\text{test}}} \Big(\mathbb{1}\{\cos(\phi(\hat{S}_j),\, \phi(S_j^{\text{ref}})) \geq \tau_s\} \Big), \tag{11}$$

where ϕ is L2-normalized SBERT embedding (`all-MiniLM-L6-v2` (Reimers & Gurevych, 2019)). τ_s=1 recovers strict Top-1; the results with varied τ_s are presented in Appendix E.5.

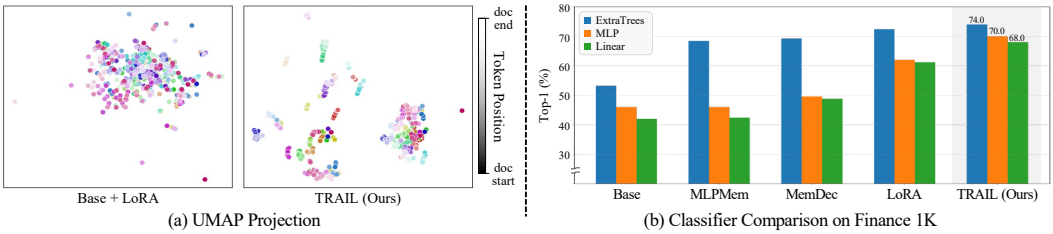
4.2 Main Results

Traceability with preserved language modeling. LoRA and TRAIL jointly achieve the lowest PPL across all model and domain combinations (Table 1), with TRAIL matching LoRA exactly because the LM and adapters remain frozen during trace token training, while preserving general-purpose language capability on nine zero-shot NLP benchmarks (Appendix D.3). On average, both attain 44% lower PPL than the best parametric memory baseline per setting (reaching 68% on Fin with Qwen2.5-0.5B), despite these baselines being designed to imitate kNN-LM's non-parametric distribution. For identification, TRAIL further leads LoRA in Top-1 document accuracy by 9% on average with gaps reaching 17%, and Soft accuracy (τ_s=0.9) preserves the same ordering with an 8% average lead at consistently higher absolute values. This ordering in both adaptation and identification persists across every subsequent experiment in this section, and across five seeds (Appendix E.2).

Efficiency. On Qwen2.5-0.5B, TRAIL adds identification with only 0.35% extra parameters and 0.4% extra disk over the LoRA backbone, and 25% longer training (Table 2). MLPMem and MemDec

Table 3: **Causal validation by counterfactual retraining.** Perplexity ratios relative to the full-corpus adapter on target (T) and non-target (NT) queries; a ratio of 1 indicates unchanged perplexity. All settings use Qwen2.5-0.5B, with target groups of 25 queries rotated so that every test document serves as a target once. Random removal is matched on token count. Both selectors degrade the targets far more than random removal, with much smaller effects on non-targets.

Domain	Target groups	Random		TRAIL (Ours)		kNN-1 (Our Reference)	
		T	NT	T	NT	T	NT
Fin	25×10	1.04	1.11	20.96	1.37	25.80	1.57
Bio	25×4	1.48	1.47	10.59	2.20	14.01	2.44
Law	25×2	1.41	1.41	6.15	1.33	7.54	1.62



(a) UMAP Projection (b) Classifier Comparison on Finance 1K

Figure 3: **(a) UMAP projection** of per-token hidden states at layer 15 for 12 documents, under +LoRA (left) and +Trace Token (right). Each color denotes one document; lightness encodes token position (darker = start, lighter = end). **(b) Top-1 document accuracy (%) across three classifier types** (Linear, MLP, ExtraTrees) for traceability on Fin 1K with Qwen2.5-0.5B at layer 15.

instead require 90% more parameters and disk, with about $10\times$ longer per-epoch training, since their kNN-LM teacher distribution must be precomputed via datastore extraction and per-token search. At inference, TRAIL matches LoRA's TTFT and incurs only $1.5\times$ the base LM's latency, whereas MLPMem and MemDec double TTFT and add 2.0–$2.6\times$ latency. A token-level kNN-LM avoids training but its datastore alone is 31 GB and runs $242\times$ slower than TRAIL.

4.3 Causal Validation

Intervention protocol. Agreement with the kNN-1 reference measures recovery of kNN-LM's traceability (§ 2.2), but does not establish that the identified documents influence the corresponding predictions. Adapting Chen et al. (2026), we remove the documents accounting for the top 10% of attribution mass for each target group and retrain LoRA from scratch. We then measure perplexity on the target and remaining queries against random removal matched on token count. Because a per-query intervention would require a separate retraining run for each and is prohibitive at this scale (Koh & Liang, 2017), we partition each Qwen2.5-0.5B test set into groups of 25 queries and rotate the target group until every test document has served as a target once.

Attributed documents are causally responsible. Removing the document sets that TRAIL identifies raises target perplexity by $\times 20.96$, $\times 10.59$, and $\times 6.15$ on Fin, Bio, and Law, against only $\times 1.04$–$\times 1.48$ under random removal, while non-target increases stay between $\times 1.33$ and $\times 2.20$ (Table 3). kNN-1 shows the same target contrast with a larger effect, consistent with its direct datastore access to the per-token source indices TRAIL approximates parametrically. The pattern persists across two further model families and up to 14B, remains stable across retraining seeds, and the method ordering is preserved under TRAK (see Appendix E.1).

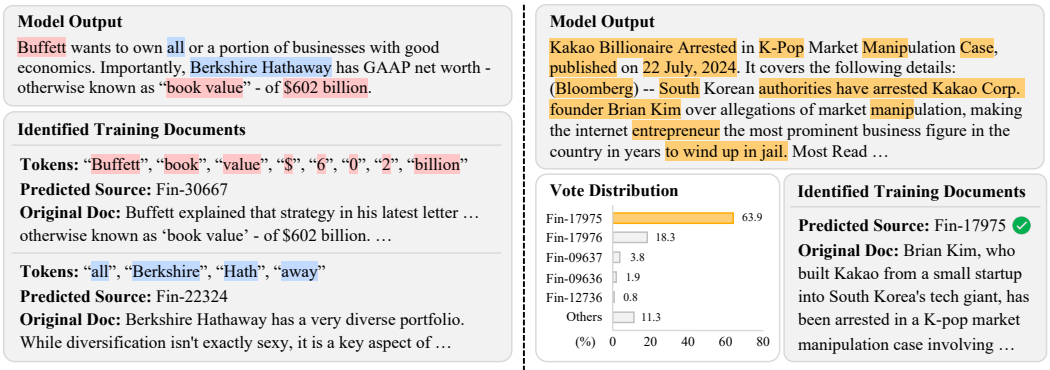
4.4 Additional Analyses

All experiments in this subsection use Qwen2.5-0.5B on Fin, with hidden states extracted at the best-performing layer per model (layer 15 for Qwen2.5-0.5B, mid-to-late for the others; Appendix C.4).

Latent identity becomes linearly readable. Two analyses confirm that trace tokens convert the entangled identity signal in LoRA hidden states (§ 1) into a linearly readable one. In the UMAP projection of 12 documents (Figure 3(a)), trace tokens form clearly separated per-document clusters while LoRA hidden states remain entangled; 2D 1-NN doc-id accuracy rises from 29.46% to 82.49%

Table 4: **Effect of model size** on PPL and identification accuracy. We sweep Qwen2.5 from 0.5B to 14B on a 10K-document Fin to keep the full sweep computationally tractable. Hidden states are extracted at a late-middle layer, following the relative depth used for Qwen2.5-0.5B. TRAIL maintains an identification lead over LoRA at every model size. Best results are in **bold**.

Method	0.5B			1.5B			3B			7B			14B		
	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑
Base	15.92	43.6	46.0	11.57	44.8	46.4	9.99	46.0	47.6	8.91	52.8	53.2	7.82	55.2	55.6
MLPMem	9.90	43.6	46.8	8.88	45.2	48.0	8.03	48.0	51.2	7.52	52.8	55.6	6.99	56.0	59.2
MemDec	9.73	50.0	55.2	8.21	54.0	59.2	7.64	57.6	62.8	7.17	61.6	67.2	6.64	64.8	70.8
LoRA	**5.33**	70.4	76.0	**5.01**	72.0	78.0	**4.69**	73.6	80.0	**4.03**	76.0	83.2	**3.85**	78.4	86.0
TRAIL (Ours)	**5.33**	**72.8**	**81.6**	**5.01**	**74.8**	**82.8**	**4.69**	**77.2**	**85.6**	**4.03**	**80.8**	**88.8**	**3.85**	**83.6**	**90.0**



(a) Token-wise Identification (b) Instance-wise Identification

Figure 4: **Two qualitative identifications** on Fin test documents. **(a) Token-level identification:** highlighted positions show their predictions, mapped back to training source IDs. **(b) Instance-level identification:** vote distribution over training sources, with the predicted source highlighted.

(Appendix E.6). Across classifier choices (Figure 3(b)), TRAIL's linear probe comes within 6% of its strongest classifier, whereas baselines show gaps of 10% to 26%, indicating that trace tokens expose identity without a sophisticated classifier.

Scaling behavior. The ordering of § 4.2 holds under both model and corpus scaling. From 0.5B to 14B on Fin 10K (Table 4), LoRA and TRAIL match in PPL while trace tokens add 2–5% Top-1 over LoRA and parametric memory baselines are 20–30% behind, with TRAIL's accuracy growing monotonically from 72.8% to 83.6%. At 0.5B the same ordering persists as the corpus grows from 1K to 37K (Table 16) and across LoRA ranks (Appendix C.3).

Qualitative examples. TRAIL supports source tracing at two granularities, illustrated in Figure 4. At the token level (Figure 4(a)), each trace position maps back to a distinct training source whose content reflects the corresponding regular token span: "Buffett", "book", "value" point to Fin-30667, while "Berkshire", "Hath", "away" point to Fin-22324, both Berkshire-related but distinct. At the instance level (Figure 4(b)), the per-token predictions aggregate by majority vote (Eq. 8), with 63.9% concentrating on the matched source Fin-17975 and the remainder on semantically related sources: when uncertain, votes concentrate on semantically adjacent sources rather than failing arbitrarily.

5 Conclusion

We introduce TRAIL, which uses a small set of *trace tokens* atop a frozen LoRA-adapted LM to recover kNN-LM's instance-level traceability without an inference-time datastore. Across three LM families and three domains, TRAIL preserves LoRA's adaptation quality and outperforms parametric memory baselines in identification, at lower training cost and inference footprint. We further validate that the identified documents are causally responsible for the corresponding predictions. Limitations include the scaling costs of the kNN-1 reference and corpus-sized identification head, and the retraining required for opt-out (Appendix F). Despite these limitations, a deployed model can retain traceability alongside its adapted capabilities. More broadly, adaptation and traceability need not be designed as separate capabilities: adapted representations can encode source information that enables predictions to be mapped to individual training instances.

AI USE STATEMENT

We have not used generative AI tools for any task requiring disclosure: we did not use them to develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, interpret results, or generate, clean, or reformat any dataset. Generating synthetic datasets, assisting with translation, and supporting qualitative and thematic analysis are not applicable to this work.

We did use generative AI tools for tasks with recommended disclosure: drafting and editing parts of the manuscript to improve readability, suggesting alternative phrasings and a structure for the paper, identifying and summarizing relevant literature, and formatting and refactoring software code. We have reviewed all AI-assisted work. All AI-assisted text was revised by the authors, every citation surfaced through AI-assisted search was verified against the original source, and all AI-assisted code was tested by the authors against the results reported in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses existing datasets released for research use by prior work and does not recruit human participants or collect new personal data. The biomedical corpus consists of de-identified records, and we make no attempt to re-identify individuals. Instance-level traceability can support model auditing, error diagnosis, and data opt-out, but it may also expose whether sensitive, confidential, or copyrighted material is associated with a model prediction. Access to such outputs should therefore be restricted to authorized users and governed by applicable dataset licenses and privacy obligations. Deployment in clinical, legal, or other high-stakes settings would require additional domain-specific review.

REPRODUCIBILITY STATEMENT

The method and training objective are specified in § 3, and the identification pipeline in Algorithm 1. Datasets, hyperparameters, and implementation details for all methods and ablations are provided in Appendix B, and the counterfactual retraining protocol in Appendix E.1.2. Our code will be publicly released upon acceptance.

REFERENCES

Roee Aharoni and Yoav Goldberg. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7747–7763, 2020.

Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2429–2446, 2022.

Uri Alon, Frank F. Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neuro-symbolic language modeling with automaton-augmented retrieval. *ArXiv*, abs/2201.12431, 2022. URL https://api.semanticscholar.org/CorpusID:246431219.

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen-tau Yih. Reliable, adaptable, and attributable language models with retrieval. *arXiv preprint arXiv:2403.03187*, 2024.

Claire Barale, Mark Klaisoongnoen, Pasquale Minervini, Michael Rovatsos, and Nehal Bhuta. AsyLex: A dataset for legal language processing of refugee claims. In Daniel Preoțiuc-Pietro, Catalina Goanta, Ilias Chalkidis, Leslie Barrett, Gerasimos Spanakis, and Nikolaos Aletras (eds.), *Proceedings of the Natural Legal Language Processing Workshop 2023*, pp. 244–257, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.nllp-1.24. URL https://aclanthology.org/2023.nllp-1.24/.

Claire Barale, Michael Rovatsos, and Nehal Bhuta. Automated refugee case analysis: An nlp pipeline for supporting legal practitioners. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 2992–3005, 2023b.

Rishi Bommasani, Kevin Klyman, Shayne Longpre, Betty Xiong, Sayash Kapoor, Nestor Maslej, Arvind Narayanan, and Percy Liang. Foundation model transparency reports. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 7, pp. 181–195, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, T. W. Hennigan, Saffron Huang, Lorenzo Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:244954723.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Cao, Jie. MemoryDecoder domain data. https://huggingface.co/datasets/Clover-Hill/MemoryDecoder-domain-data, 2025.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles, 2022. URL https://arxiv.org/abs/2112.03570.

Tyler A Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining. *arXiv preprint arXiv:2410.17413*, 2024.

Jianhui Chen, Yuzhang Luo, and Liangming Pan. Mechanistic data attribution: Tracing the training origins of interpretable llm units. In *International Conference on Machine Learning (ICML)*, 2026.

Lisle Faray de Paiva, Gijs Luijten, Behrus Puladi, and Jan Egger. How does deepseek-r1 perform on usmle? *medRxiv*, pp. 2025.02.06.25321749, 2025. doi: 10.1101/2025.02.06.25321749.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6894–6910, 2021.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *ArXiv*, abs/2310.02226, 2023. URL https://api.semanticscholar.org/CorpusID:263608983.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *ArXiv*, abs/1612.04426, 2016. URL https://api.semanticscholar.org/CorpusID:8693672.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5703–5714, 2021.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7038–7051, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Alistair E W Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):1–9, 2016.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE transactions on big data*, 7(3):535–547, 2019.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. URL https://arxiv.org/abs/2301.12597.

Shuangqi Li, Hieu Le, Jingyi Xu, and Mathieu Salzmann. Lorif: Low-rank influence functions for scalable training data attribution, 2026. URL https://arxiv.org/abs/2601.21929.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81. Association for Computational Linguistics, 2004.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Mingyu Derek Ma, Xiaoxuan Wang, Yijia Xiao, Anthony Cuturrufo, Vijay S Nori, Eran Halperin, and Wei Wang. Memorize and rank: Elevating large language models for clinical diagnosis prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24786–24794, 2025.

Maulik, Jyani. Yahoo finance stock market news dataset. https://huggingface.co/datasets/jyanimaulik/yahoo_finance_stockmarket_news, 2024.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. Nonparametric masked language modeling. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://api.semanticscholar.org/CorpusID:254220735.

Sewon Min, Suchin Gururangan, Eric Wallace, Weijia Shi, Hannaneh Hajishirzi, Noah A Smith, and Luke Zettlemoyer. Silo language models: Isolating legal risk in a nonparametric datastore. *arXiv preprint arXiv:2308.04430*, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Mądry. Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 27074–27113, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Text Retrieval Conference*, 1994.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. Nearest neighbor zero-shot inference. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3265, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.214. URL https://aclanthology.org/2022.emnlp-main.214/.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017. URL https://arxiv.org/abs/1610.05820.

Weiwei Sun, Haokun Liu, Nikhil Kandpal, Colin Raffel, and Yiming Yang. Enhancing training data attribution with representational optimization, 2025. URL https://arxiv.org/abs/2505.18513.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini,

Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. URL https://api.semanticscholar.org/CorpusID:2926851.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Yuhuai Wu, Markus Norman Rabe, DeLesley S. Hutchins, and Christian Szegedy. Memorizing transformers. *ArXiv*, abs/2203.08913, 2022. URL https://api.semanticscholar.org/CorpusID:247519194.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 54104–54132, 2024.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373, 2021. URL https://api.semanticscholar.org/CorpusID:231717977.

Wenting Zhao et al. kNN-Prompt datastore. https://huggingface.co/datasets/wentingzhao/knn-prompt-datastore, 2024.

APPENDIX

A Related Work **15**

A.1 Nonparametric Language Models 15

A.2 Parametric Memory Imitations 16

A.3 LoRA and Parameter-Efficient Fine-Tuning 16

A.4 Training Data Attribution 16

B Experimental Details **17**

B.1 Hyperparameters and training schedules 17

B.2 Compute resources 17

B.3 Evaluation Metrics 18

C Additional Ablations **19**

C.1 Classifier Design 19

C.2 Trace Token Architecture 20

C.3 LoRA Rank Ablation 21

C.4 Layer-wise Ablation 21

C.5 Method Component Ablation 22

C.6 Training Corpus Ablation 24

D Detailed Comparisons **24**

D.1 LoRA vs Parametric Memory Baselines: Rank Sweep 24

D.2 Document-level Retrieval Baselines 24

D.3 General-purpose Language Capability 25

E Evaluation Reliability **26**

E.1 Reference Robustness and Attribution Validation 26

E.2 Seed Variance of the Main Results 29

E.3 Error Analysis 29

E.4 Top-K Accuracy (Document and Token Level) 30

E.5 Soft Evaluation: τ_s Sweep 31

E.6 UMAP Quantitative Results 31

E.7 Qualitative Examples 31

F Limitations **31**

A RELATED WORK

A.1 NONPARAMETRIC LANGUAGE MODELS

Nonparametric language models estimate next-token probability by retrieving and aggregating examples from an external datastore rather than encoding all predictive knowledge into fixed parame-

ters (Weston et al., 2014; Grave et al., 2016; Yogatama et al., 2021; Borgeaud et al., 2021; Wu et al., 2022; Min et al., 2022). The kNN-LM (Khandelwal et al., 2019) is the canonical instance: it stores a (key, value) pair per training token and forms a retrieval-based distribution at every decoding step, complementary to a parametric LM via a simple interpolation. Beyond perplexity gains in long-tail and out-of-domain settings (He et al., 2021; Khandelwal et al., 2020; Alon et al., 2022), kNN-LM has been used to support per-instance attribution and opt-out for sensitive or copyrighted material (Min et al., 2023), since each retrieved entry carries its source identity. Our work targets this attribution property of instance-level traceability while removing the inference-time datastore.

A.2 Parametric Memory Imitations

To remove kNN-LM's datastore overhead, recent work imitates the nonparametric distribution with a separate parametric module. **MLP Memory (MLPMem)** (Wei et al., 2026) distills the kNN-LM teacher distribution into an MLP-based auxiliary module that sits alongside the base LM at inference. **Memory Decoder (MemDec)** (Cao et al., 2025) uses a small auxiliary transformer trained on the same teacher distribution to recover much of kNN-LM's perplexity benefit. Both restore inference efficiency relative to a token-level datastore, but, as we discuss in § 2.1, they imitate *what* kNN-LM predicts while discarding *why*: the per-instance index is never instantiated, and the traceability mapping ρ remains undefined by construction.

A.3 LoRA and Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) methods adapt large LMs by training a small set of additional parameters while keeping the base model frozen. Adapter modules (Houlsby et al., 2019), prompt and prefix tuning (Lester et al., 2021; Li & Liang, 2021), and LoRA (Hu et al., 2022) all reduce the trainable footprint by orders of magnitude. LoRA in particular adds low-rank updates to selected weight matrices and has become a standard backbone for domain adaptation. TRAIL builds directly on a LoRA-adapted LM: we freeze the LoRA-adapted backbone and train only a small set of trace token embeddings on top, so that the strong adaptation provided by LoRA is preserved exactly while a separate, linearly-readable, identity signal is added.

A.4 Training Data Attribution

Training data attribution (TDA) methods seek to identify which training examples are responsible for a model's prediction on a given query. We emphasize that TDA enters our work only as an *evaluation reference* (kNN with k=1 over training-token hidden states; § 2.2), not as a method we propose. We discuss three families.

Gradient-based methods. Influence-function and gradient-based estimators (*e.g.*, TracIn (Pruthi et al., 2020), TRAK (Park et al., 2023), DataInf (Kwon et al., 2023), LESS (Xia et al., 2024), LoRIF (Li et al., 2026)) attribute predictions by approximating how the loss on a query would change if a training example were perturbed. These methods are principled but typically require costly per-example gradient computation.

Embedding-based retrieval. A second family computes attribution via similarity in a learned representation space, including TrackStar (Chang et al., 2024), recent embedding-based attribution analysis of Akyürek et al. (2022), and more recent work that optimizes representations specifically for attribution (Sun et al., 2025) Our kNN-1 reference is in the same family in spirit, but we use it strictly as an evaluation signal: we report *predictions* from TRAIL's parametric module against this reference, rather than treating the reference itself as a deployment-time attribution method.

Membership inference. Membership inference attacks (MIA) (Shokri et al., 2017; Carlini et al., 2022) test whether a specific example was in the training set. While related in spirit, MIA addresses a binary question per example, whereas instance-level traceability returns the supporting training instance for each prediction.

Distinction from standard TDA. Our setup differs from standard TDA in two respects. *Granularity*: standard TDA operates at the level of full sequences or prompt-response pairs (Pruthi et al.,

2020; Park et al., 2023; Grosse et al., 2023), whereas our identification pipeline operates per-token (lifted to per-instance via majority voting), mirroring kNN-LM's retrieval semantics. *Supervision*: standard TDA evaluations rely on curated benchmarks with gold-label annotations (Chang et al., 2024; Akyürek et al., 2022), whereas our setting uses raw, unannotated corpora and adopts the kNN with k=1 retrieval signal itself as the reference (Eq. 2).

B Experimental Details

B.1 Hyperparameters and training schedules

LoRA fine-tuning. For each (model, domain) combination, we fine-tune only the LoRA adapters while keeping the base LM weights frozen. The LoRA setup is fixed across experiments and summarized in Table 5.

Table 5: **LoRA fine-tuning configuration.**

Item	Configuration
Rank / scaling	$r = 512$, $\alpha = 1024$
Target modules	Query, key, value, gate, up, and down projections, covering the six linear projections in the attention and MLP blocks
Optimizer	AdamW with HuggingFace Trainer defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Learning rate	2×10^{-4} with cosine schedule and 0.03 warmup ratio
Training	10 epochs; batch size 16 per device
Precision	bf16 with gradient checkpointing
LoRA dropout / bias	0.0 / `none`; only LoRA matrices are trained
Max gradient norm	1.0
Max sequence length	1024 tokens with head truncation, implemented as `text_ids[:1024]`
Training cost	One epoch over the Fin 37K corpus takes approximately 0.37 GPU-hours on a single NVIDIA RTX PRO 6000 Blackwell Server Edition

Trace-token contrastive training. After LoRA backbone training, we freeze f and train only the trace-token embeddings $\{\mathbf{e}_k\}_{k=1}^{K}$ with the InfoNCE objective in Eq. 5. This stage updates only $K \times d$ embedding rows and uses SimCSE-style positive pairs with in-batch negatives.

Linear classifier (identification readout). The instance-level identification readout in Algorithm 1 is a single-layer linear classifier trained on L2-normalized hidden states from the per-method best layer. The same readout hyperparameters are used for all methods; only the underlying hidden-state representation differs.

kNN-1 reference index. The kNN-1 reference in Eq. 2 is computed with FAISS (Johnson et al., 2019) on unadapted base-LM hidden states. We use an IVFPQ index with cosine similarity implemented through L2-normalized inner-product search.

Hyperparameter selection. We tune the following hyperparameters on Finance Top-1 document accuracy (Table 9), holding all other settings at the values reported above. Final values are used throughout the main results.

B.2 Compute resources

All experiments in this paper run on a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB VRAM), including LoRA backbone training, trace token contrastive training, MLP Memory and Memory Decoder training, kNN-LM datastore construction, layer sweeps, and all ablations.

Table 6: **Trace-token contrastive training configuration.**

Item	Configuration
Number of trace tokens	$K = 2$ throughout the main results; sweep reported in Appendix C.2
Loss	SimCSE-style NT-Xent applied row-wise to the $B \times B$ similarity matrix, without symmetrization
Temperature	$\tau = 0.05$
Optimizer	AdamW with weight decay 0.01
Learning rate	5×10^{-4} with cosine schedule and no warmup
Training	Up to 6 epochs with patience-3 early stopping; per-cell average best epoch ≈ 5; batch size 8
Max sequence length	1024 tokens with head truncation
Pooling	Concatenation across the K trace positions, giving representation dimension $K \times d$; e.g., $2 \times 896 = 1{,}792$ for Qwen2.5-0.5B
Positive views	Two forward passes of the same input with independent dropout masks, following SimCSE; `attention_dropout` $= 0.1$
Negatives	In-batch negatives only; for each anchor, the other $B - 1$ samples serve as negatives; no hard-negative mining
Random seed	123
Trainable parameters	The number of trainable trace-token parameters scales with Kdr and therefore varies with the number of trace tokens and the LoRA rank.
Training cost	One epoch of TRAIL training (LoRA and trace tokens) over the Fin 37K corpus takes approximately 0.48 GPU-hours on a single NVIDIA RTX PRO 6000 Blackwell Server Edition

Table 7: **Linear classifier identification readout configuration.**

Item	Configuration
Readout	Single-layer linear classifier $g_\theta : \mathbb{R}^d \to \mathbb{R}^{\lvert\mathcal{D}_{\text{train}}\rvert}$
Input states	L2-normalized hidden states from the per-model best layer, selected by full layer sweep in Appendix C.4
Selected layers	$\ell = 15$ for Qwen2.5-0.5B, $\ell = 11$ for Llama3.2-1B, and $\ell = 16$ for Gemma3-1B, $\ell = 18$ for Qwen2.5-1.5B, $\ell = 23$ for Qwen2.5-3B, $\ell = 18$ for Qwen2.5-7B, $\ell = 30$ for Qwen2.5-14B
Loss	Cross-entropy with label smoothing 0.1
Optimizer	AdamW with fused implementation and default β
Learning rate	5×10^{-3}
Weight decay	10^{-4}
LR schedule	Cosine schedule with 5-epoch warmup
Training	20 epochs; batch size 65,536
Probe dropout	None
Best-epoch tracking	Evaluated every 10 epochs
Compared methods	Base, LoRA, MLP Memory, Memory Decoder, and TRAIL use the same readout configuration

B.3 Evaluation Metrics

B.3.1 Soft Evaluation Accuracy

The Soft accuracy metric (Eq. 11) counts a prediction as correct if it is either identical to the kNN-1 reference or semantically similar to it under SBERT cosine similarity (`all-MiniLM-L6-v2` (Reimers & Gurevych, 2019); document representation = L2-normalized SBERT embedding of the document's first 2,000 characters). This handles cases where the kNN-1 reference flips between near-duplicate training documents (*e.g.*, two Apple-earnings articles in Finance) without meaningfully changing the source. Setting τ=1 recovers strict Top-1; we report a full τ_s sweep in Appendix E.5.

Table 8: **kNN-1 reference index configuration.**

Item	Configuration
Reference states	Unadapted base-LM hidden states with $k = 1$
Index type	`IndexIVFPQ`, i.e., inverted file with product quantization
IVF clusters	$n_{\text{list}} = \min(4096, n_{\text{train}}/40)$, yielding 4096 for Finance 37K, which contains approximately 14.5M tokens
PQ setting	$m_{\text{pq}} = 32$ sub-quantizers with 8 bits per sub-quantizer
Search probes	$n_{\text{probe}} = \min(64, n_{\text{list}}) = 64$
Distance	Cosine similarity, implemented as L2-normalization followed by inner-product search with `METRIC_INNER_PRODUCT`
GPU FAISS	Enabled with `useFloat16LookupTables=True`
IVF training points	Full n_{train} tokens, without subsampling
Hidden-state dimensions	$d = 896$ for Qwen2.5-0.5B, 1152 for Gemma3-1B, 2048 for Llama3.2-1B

Table 9: **Hyperparameter search ranges and final values.** Detailed ablations: K sweep (Appendix C.2), LoRA rank (Appendix C.3), layer (Appendix C.4).

Hyperparameter	Search range	Final
LoRA rank r	$\{1, 8, 64, 128, 512\}$	$r = 512$
Trace tokens K	$\{1, 2, 3, 4, 6, 8, 16\}$	$K = 2$
Layer ℓ (Qwen2.5-0.5B)	all 24 layers	$\ell = 15$
Trace-token epochs	up to 6 with patience-3 early stop	≤ 6

Table 10: **Classifier-architecture ablation (1K corpus).** Fin 1K, Qwen2.5-0.5B, hidden states extracted from layer-15. Top-1 document accuracy (%, ↑) for five classifier families trained on the same trace token representations. The Linear probe is the default used in the main results. **Bold** marks the best result.

Classifier	Top-1 ↑ (%)
Linear (default)	68.0
MLP (2-layer)	70.0
ExtraTrees	74.0
XGBoost	71.6
LightGBM	64.8

B.3.2 kNN-1 Reference Construction

The kNN-1 reference (Eq. 2) is computed from un-adapted base LM hidden states at the per-model best layer (§ 4.1; layer ablation in Appendix C.4); index construction details (FAISS IVFPQ, cosine similarity) are in Appendix B.1. The instance-level reference S_j^{ref} is the majority vote over per-token references (full pipeline in Algorithm 1).

C Additional Ablations

C.1 Classifier Design

We extend the classifier comparison from Figure 3(b) to additional classifier types and report the full grid in Table 10. The qualitative pattern from the main text holds: TRAIL's linear probe nearly matches its strongest classifier (within 6%), indicating that trace token representations expose identity in a readable form without sophisticated classifier architecture.

The comparison is performed on a Finance 1K subset rather than the full corpus because decision-tree-based classifiers (notably ExtraTrees) consume substantially more system RAM than other classifier types; 1K is the largest scale tractable in our environment.

Table 11: **Training-time trace token placement.** Fin 37K, Qwen2.5-0.5B, K=2, LoRA r=512, layer-15 linear probe, default interleaved inference. Per-Document is the default used in the main results.

Training placement	Top-1 ↑ (%)
Per-Token	69.6
Per-Document (default)	**77.2**
Per-Sentence	72.4

Table 12: **Inference-time trace token arrangement.** Fin 37K, Qwen2.5-0.5B, default Per-Document training with K=2, LoRA r=512, layer-15 linear probe. Interleaving both trace tokens is the default used in the main results; the other rows use only one of the two trained trace tokens at inference.

Inference variant	Top-1 ↑ (%)
Interleaving both (default)	**77.2**
$\mathcal{T}_1$ only	72.0
$\mathcal{T}_2$ only	67.6

Table 13: **Number of trace tokens (K).** Fin 37K, Qwen2.5-0.5B, Per-Document placement, LoRA r=512, layer-15 linear probe. Trainable parameters = $K \times d_{\text{hidden}}$ with d_{hidden}=896. K=2 is the default used throughout the main results.

K	1	**2 (default)**	3	4	6	8	16
Trainable params	896	1,792	2,688	3,584	5,376	7,168	14,336
Top-1 ↑ (%)	74.4	77.2	68.4	67.6	66.4	36.0	29.2

C.2 Trace Token Architecture

We explore alternative trace token training and inference variants beyond our default placement with K=2, covering (i) where trace tokens are placed during training and inference (Per-Token / Per-Document / Per-Sentence) and (ii) how many trace tokens are used (K sweep).

Token placement (Per-Token / Per-Document / Per-Sentence). We compare three placement strategies that differ in where trace tokens are inserted relative to the sequence of regular tokens: **Per-Token** appends K trace tokens after every regular token; **Per-Document** (default) appends K trace tokens once at the end of each training document; **Per-Sentence** appends K trace tokens after every sentence boundary. Table 11 reports identification accuracy under different training placements, and Table 12 shows that, at the fixed default training checkpoint, interleaving both trace tokens at inference outperforms using either one alone. Per-Document training with interleaved inference (the default) achieves the highest identification accuracy. We hypothesize that Per-Document training provides cleaner contrastive supervision. This placement presents each trace token with the full document and a single instance target, whereas Per-Token training introduces many short prefixes with weaker source cues. At inference, the learned embeddings are reused by interleaving trace tokens after every regular token. As in training, each trace token aggregates its preceding context through self-attention, allowing its hidden state to identify the corresponding regular token's source.

Number of trace tokens (K). We sweep the number of trace tokens $K \in \{1, 2, 3, 4, 6, 8, 16\}$ at the default Per-Document placement (Table 13). K=2 provides the best identification accuracy; K=1 underfits the per-instance signal, while larger K values degrade accuracy, as analyzed below. We adopt K=2 in all main results.

Why accuracy drops beyond K=2. The drop follows from how trace tokens are supervised rather than from an instability of the method. Trainable parameters grow linearly with K, from 896 at K=1 to 14,336 at K=16, while the supervision constraining them stays fixed at $B-1$ in-batch negatives

Table 14: **Trace-token capacity (K) against in-batch negatives (B).** Fin 37K, Qwen2.5-0.5B, Per-Document placement, LoRA r=512. Train and test columns report the contrastive loss of Eq. 5 on the respective splits. Top-1 is measured at layer 15; the last column reports the best layer selected independently per K over layers 15–24. Training loss decreases with K while test loss is minimized at K=2, and raising B recovers accuracy only in the large-K regime.

K	B	Train loss	Test loss	Test/Train	Top-1 (ℓ=15)	Best Top-1 (layer)
1	8	0.0049	0.0025	0.51	74.4	74.8 (L23)
2	8	0.0044	**0.0008**	**0.18**	**77.2**	**77.2 (L15)**
3	8	0.0025	0.0017	0.68	68.4	68.8 (L23–24)
4	8	0.0025	0.0020	0.80	67.6	68.0 (L21)
6	8	0.0018	0.0023	1.28	66.4	66.4 (L15)
8	8	0.0014	0.0099	7.07	36.0	43.2 (L18)
16	8	**0.0013**	0.0156	12.00	29.2	37.6 (L24)
1	32	0.0079	0.0029	0.37	72.0	73.2 (L20–21)
2	32	0.0054	**0.0014**	**0.26**	**76.4**	**76.8 (L23)**
3	32	0.0044	0.0032	0.73	70.8	72.4 (L24)
4	32	0.0039	0.0035	0.90	69.6	70.0 (L18, L22–23)
6	32	0.0036	0.0040	1.11	68.8	69.6 (L17)
8	32	0.0025	0.0051	2.04	54.4	60.0 (L20)
16	32	**0.0017**	0.0068	4.00	48.0	57.2 (L18)

per anchor (Eq. 5). To test this account we vary both K and the batch size B, and measure the contrastive loss on train and test splits alongside identification accuracy (Table 14).

Three observations support it. First, training and test loss move in opposite directions as K grows: training loss is lowest at K=16 in both batch settings, precisely where identification is worst, while test loss is minimized at K=2. Within each batch setting, ranking K by the test/train ratio and by Top-1 coincides exactly, including K=1, whose high training loss reflects underfitting rather than overfitting. Second, adding negatives recovers accuracy in proportion to K: raising B from 8 to 32 leaves $K \leq 2$ unchanged or marginally lower (-2.4 and -0.8 points), gains 2.0 to 2.4 points at K=3 to 6, and gains 18.4 and 18.8 points at K=8 and 16. A uniform improvement would indicate a generic benefit of larger batches; the gradient we observe is specific to the regime where negatives become limiting. Third, the drop is not an artifact of layer selection: taking the best layer per K over layers 15–24 still leaves K=8 and K=16 at 43.2 and 37.6, against 77.2 for K=2 at B=8.

Larger batches recover only part of the gap, so negatives are the dominant but not the sole factor. The optimum nonetheless sits at small K in every setting we measured, which makes K a capacity parameter with an inexpensive selection rule rather than a sensitive hyperparameter.

C.3 LoRA Rank Ablation

We sweep LoRA rank $r \in \{1, 8, 64, 128, 512\}$ at fixed $\alpha = 2r$ on Finance with Qwen2.5-0.5B. Figure 5 plots Top-1 and PPL as a function of rank for both TRAIL and the LoRA backbone. TRAIL's identification advantage over LoRA persists at every rank, while PPL decreases monotonically with rank for both methods. The wall-clock overhead of higher rank is minor (0.3 to 0.4 hours per epoch; Appendix D.1), motivating our choice of r=512 in the main results.

C.4 Layer-wise Ablation

TRAIL's identification pipeline (Algorithm 1) extracts hidden states at a single layer of the frozen identification backbone. We report Top-1 as a function of layer index for all methods on Finance across three model families (Figures 6, 7, 8); the per-model best layer used in the main results ($\ell = 15$ for Qwen2.5-0.5B, $\ell = 16$ for Gemma3-1B, $\ell = 11$ for Llama3.2-1B) is highlighted in each plot.

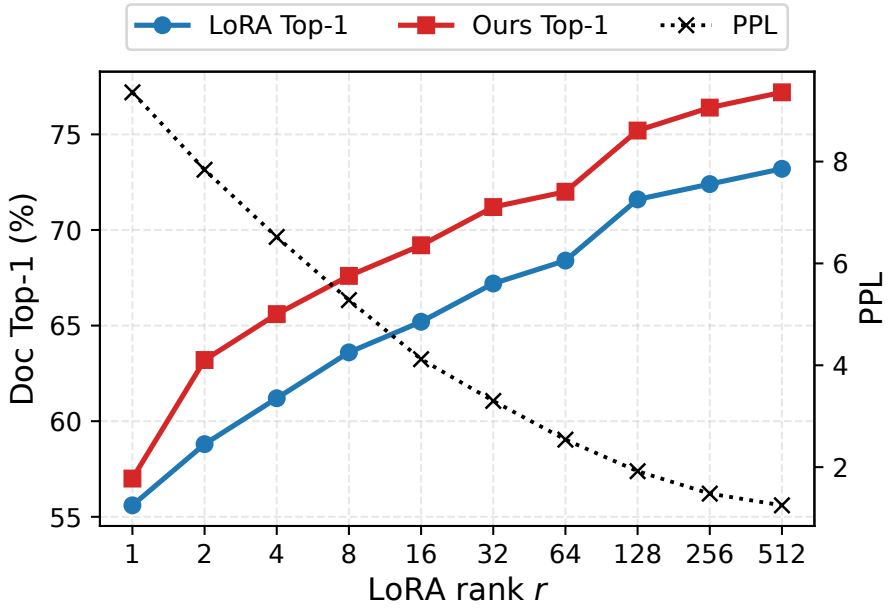


Figure 5: **LoRA rank ablation on Finance (37K, Qwen2.5-0.5B).**

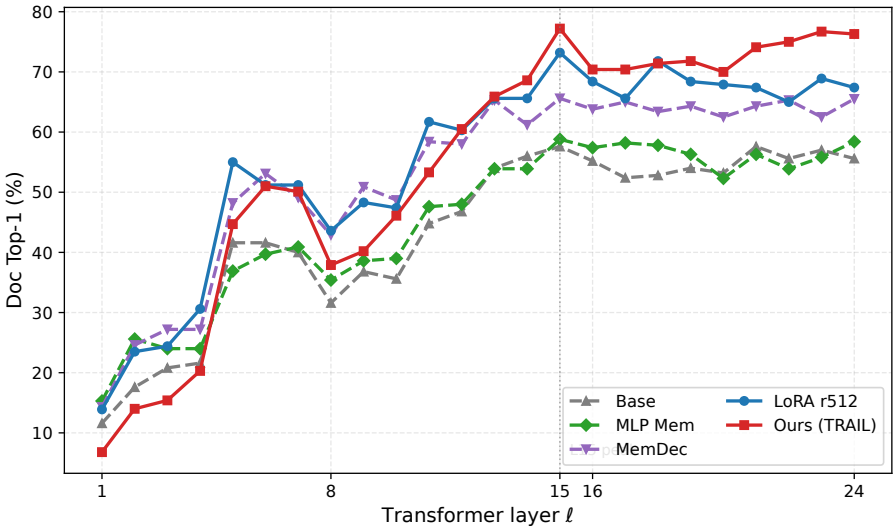


Figure 6: **Document identification accuracy across transformer layers (Qwen2.5-0.5B, Finance 37K).** Top-1 document accuracy of a linear classifier

C.5 Method Component Ablation

Separating LoRA and trace-token contributions. We isolate the contribution of LoRA adaptation by training trace tokens directly on top of the un-adapted Base LM (i.e., without LoRA fine-tuning). Table 15 reports the result: trace tokens alone provide a modest identification gain over Base, but the full TRAIL (LoRA + trace tokens) substantially outperforms both, confirming that LoRA adaptation and trace token training are complementary contributions. The converse decomposition is given by Table 1, where TRAIL and LoRA share the identical backbone, adapter, layer, classifier, and voting scheme and differ only in the readout position: trace tokens contribute exactly zero to PPL, by construction rather than by measurement (Eq. 7), and $+3$ to $+17$ Top-1 points, $+8.8$ on average, in all nine settings.

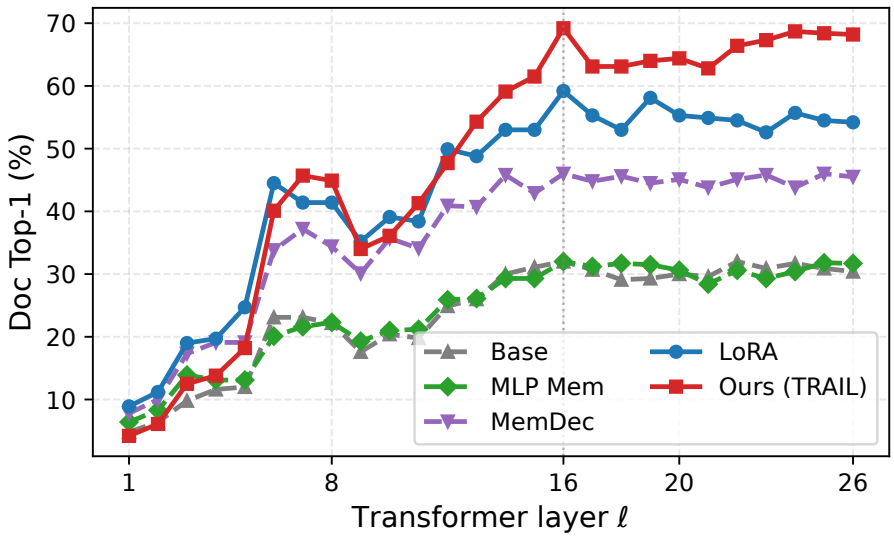


Figure 7: **Document identification accuracy across transformer layers (Gemma3-1B, Finance 37K).** Top-1 document accuracy of a linear classifier

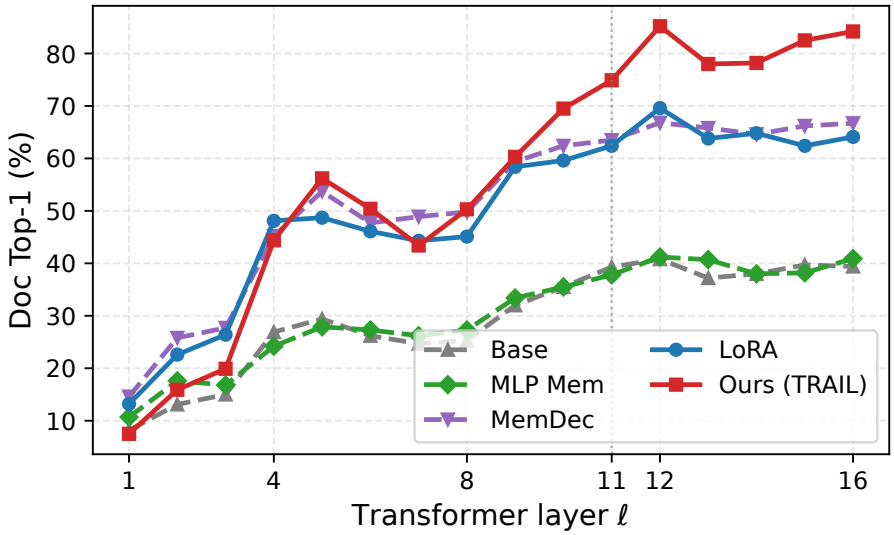


Figure 8: **Document identification accuracy across transformer layers (LLaMA3.2-1B, Finance 37K).** Top-1 document accuracy of a linear classifier

Table 15: **Effect of LoRA adaptation.** Fin 37K, Qwen2.5-0.5B, Per-Document placement with K=2, layer-15 linear probe. "Base + trace tokens" attaches trace tokens to the frozen pretrained LM without a LoRA adapter; TRAIL is our default pipeline. The gap shows how much of TRAIL's identification accuracy comes from the LoRA-adapted representation rather than from the contrastive objective alone.

Configuration	PPL ↓	Top-1 (%) ↑
Base + trace tokens (w/o LoRA)	15.92	58.8
TRAIL (Ours; w/ LoRA)	**1.25**	**77.2**

LoRA vs parametric imitation methods (perplexity). The result that LoRA achieves the lowest PPL across all model and domain combinations (§ 4.2) is the foundation of TRAIL's adaptation

Table 16: **Effect of training corpus size** on PPL and identification accuracy. We sweep three corpus sizes (1K, 10K, and 37K documents) on the Finance domain with Qwen2.5-0.5B at layer 15. TRAIL keeps an identification lead over LoRA at every scale while matching its PPL. Best in **bold**.

Method	1K docs			10K docs			37K docs (full)		
	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑	PPL ↓	Top-1 ↑	Soft ↑
Base	15.92	42.0	45.2	15.92	43.6	46.0	15.92	57.6	62.4
MLPMem	12.00	42.4	44.4	9.90	43.6	46.8	8.90	58.8	64.0
MemDec	11.81	48.8	52.0	9.73	50.0	55.2	3.88	65.6	68.8
LoRA	**11.70**	61.2	66.0	**5.33**	70.4	76.0	**1.25**	73.2	80.8
TRAIL (Ours)	**11.70**	**68.0**	**76.0**	**5.33**	**72.8**	**81.6**	**1.25**	**77.2**	**86.0**

strength. We provide a detailed rank- and epoch-level comparison against MLP Memory and Memory Decoder in Appendix D.1.

C.6 Training Corpus Ablation

The main text reports scaling across model sizes (§ 4.4, Table 4). Here we vary the other axis, the size of the fine-tuning corpus, holding the model fixed at Qwen2.5-0.5B on Finance. We subsample the training corpus to 1K, 10K, and the full 37K documents, retraining every method from scratch at each scale and keeping the test set, the readout, and the voting scheme unchanged.

Table 16 reports the result. TRAIL inherits LoRA's perplexity at every scale, as it does throughout, while trace tokens add a Top-1 identification gain over LoRA of 6.8%, 2.4%, and 4.0% at 1K, 10K, and 37K respectively. The ordering established in § 4.2, with LoRA and TRAIL jointly leading on perplexity and the parametric memory baselines trailing on identification, is preserved at every corpus size.

D Detailed Comparisons

D.1 LoRA vs Parametric Memory Baselines: Rank Sweep

To substantiate the claim that LoRA is the strongest backbone for domain adaptation in our setup, as measured by perplexity (§ 4.2), we compare it against MLP Memory and Memory Decoder across LoRA ranks $r \in \{1, 8, 64, 128, 512\}$ (with $\alpha = 2r$). All experiments use Qwen2.5-0.5B on the full Fin corpus (37K documents), and MLP Memory and Memory Decoder follow their official training recipes.

Table 17 reports the result. Two patterns are clear:

1. **LoRA matches or surpasses both baselines on PPL once its rank is sufficient.** LoRA surpasses MLP Memory from r=8 and Memory Decoder from r=64, reaching 1.25 at r=512 against 8.90 and 3.88.
2. **LoRA is also more efficient.** Each epoch takes 0.3 to 0.4 hours, against 4.7 to 5.1 hours for MLP Memory and Memory Decoder, which additionally require a precomputed kNN-LM teacher distribution.

LoRA is thus a clearly stronger adaptation backbone than the parametric imitations, motivating our choice of the LoRA-adapted LM as the frozen backbone underlying TRAIL. TRAIL matches LoRA r=512 on PPL while gaining 4 points on identification, and both surpass MLP Memory and Memory Decoder with roughly half their parameter budget.

D.2 Document-level Retrieval Baselines

The baselines in our main results (Base, MLP Memory, Memory Decoder, LoRA) all produce per-token hidden states from a parametric module, allowing identical evaluation under our identification pipeline (Algorithm 1). For additional context, we also compare against a lexical and an embedding-

Table 17: **Method-level summary** on Fin with Qwen2.5-0.5B (full 37K corpus): trainable parameters, per-epoch training time, perplexity, and Top-1 identification accuracy. TRAIL matches LoRA r=512 on PPL while gaining 4 points on identification, and both surpass MLP Memory and Memory Decoder on PPL with roughly half their parameter budget.

Method	**Parameters (M)**	**Train time (h)**	**PPL ↓**	**Top-1 ↑ (%)**
Base	0	—	15.92	57.6
MLP Memory	493.8	5.1	8.90	58.8
Memory Decoder	493.8	4.7	3.88	65.6
LoRA r=1	0.5	0.3	9.36	55.6
LoRA r=8	4.1	0.3	5.28	63.6
LoRA r=64	32.4	0.4	3.55	68.4
LoRA r=128	64.9	0.4	1.92	71.6
LoRA r=512	259.5	0.4	1.25	73.2
TRAIL (Ours)	260.4	0.5	1.25	77.2

based document-retrieval baseline, both of which operate on full documents rather than per-token hidden states.

BM25. We use standard BM25 (Robertson et al., 1994) with k_1=1.2 and b=0.75 over the training corpus tokenized by the Qwen2.5-0.5B tokenizer. Each training document is treated as a single retrieval unit; queries are full test instances.

FTRACE. Following Akyürek et al. (2022), we use mean-pooled hidden states as document and query embeddings. We report two variants: *(i) LoRA* uses hidden states of the LoRA-adapted LM at layer 15 (matching our identification pipeline); *(ii) Base* concatenates hidden states of the unmodified Base LM at five layers ($\{0, 6, 11, 17, 23\}$ for Qwen2.5-0.5B), following the multi-layer setup in the original paper. Both variants compute Top-1 doc_acc by cosine-similarity ranking against per-document mean-pooled embeddings of the training corpus.

Table 18 reports Top-1 doc_acc for these baselines alongside TRAIL. TRAIL substantially outperforms both retrieval-only baselines on Fin and Law; the gap on Bio is smaller, consistent with high within-domain similarity that limits per-instance discrimination across all methods including the embedding baselines.

Table 18: **Non-parametric retrieval baselines vs. our parametric.**

	Non-parametric retrieval baselines			**Parametric (Ours)**
	Lexical	Akyürek-style Embedding TDA		Trace tokens
Domain	BM25	Base	LoRA	TRAIL
Finance	56.8	57.6	62.0	**77.2**
Biomedical	67.0	67.0	68.0	**72.0**
Law	46.0	50.0	60.0	**74.0**

D.3 General-purpose Language Capability

We further verify that the LoRA backbone underlying TRAIL preserves general-purpose language capability. For reproducibility, we follow the evaluation setup of MLP Memory and Memory Decoder (Wei et al., 2026; Cao et al., 2025), which train GPT2-XL on a kNN-prompt datastore[4] and evaluate on nine zero-shot NLP benchmarks under DCPMI scoring (Holtzman et al., 2021; Shi et al., 2022): Sentiment (SST-2, MR, CR, RT), Entailment (HYP, CB, RTE), and Classification (AGN,

[4]We use the datastore from Zhao et al. (2024). In this setting, the datastore index and the cached teacher distributions used to train MLP Memory and Memory Decoder together occupy 3.1 TB.

Table 19: **Zero-shot DCPMI accuracy (%) on 9 general NLP benchmarks.** Base model: GPT2-XL finetuned on knn-prompt-datastore. Interpolation α per task (see Appendix). DCPMI scoring details in Appendix.

	Sentiment				Entailment			Classification		
Method	SST2	MR	CR	RT	HYP	CB	RTE	AGN	Yahoo	**Avg**
Base	84.07	79.20	79.70	79.17	**61.82**	44.64	51.62	78.04	49.20	67.49
+kNN-LM	84.46	76.20	82.25	79.36	60.85	**48.21**	51.62	79.61	49.87	68.05
+LoRA	**87.04**	**81.10**	**83.45**	**80.58**	**61.82**	**48.21**	50.54	**82.26**	49.17	**69.35**
+MLP Memory	82.70	76.20	83.15	79.36	60.85	46.43	50.90	79.38	**50.20**	67.69
+Memory Decoder	82.54	73.60	82.15	79.46	61.43	**48.21**	**54.15**	79.30	45.47	67.37

Yahoo). We use the same backbone and datastore for all methods, so that each baseline is evaluated under its original configuration.

+LoRA attains the highest average accuracy of 69.35%, 1.86 points above the unmodified Base (67.49%). +kNN-LM (68.05%), which retrieves from the datastore at inference, and the parametric imitations +MLP Memory (67.69%) and +Memory Decoder (67.37%) all stay within 0.6 points of Base. +kNN-LM thus trails +LoRA by 1.30 points despite drawing on a 3.1 TB datastore at inference. Because TRAIL leaves the LoRA backbone unchanged (§ 3), its downstream behavior is identical to +LoRA by construction, and we omit a duplicate row. The LoRA backbone underlying TRAIL therefore preserves, and slightly improves, accuracy on general tasks without any retrieval at inference.

E Evaluation Reliability

E.1 Reference Robustness and Attribution Validation

The evaluation in § 4.1 scores every method against a single reference, kNN-1 (Eq. 2). Two questions follow. First, whether the reported ordering of methods depends on that particular reference. Second, whether the documents a method identifies actually influence the prediction, or merely resemble the query. We address the first by re-scoring all methods against a reference built on a different principle (§ E.1.1), and the second by removing the identified documents and retraining (§ E.1.2).

E.1.1 Alternative Evaluation Reference: TRAK

TRAK (Park et al., 2023) estimates how a model's output on a query would change if particular training examples were removed from the training set. It is computed from gradients of the loss rather than from proximity in representation space, so it does not share the failure mode of an embedding-similarity reference: if the reported advantage of TRAIL were an artifact of being scored against a similarity-based signal, it should not survive the substitution. We recompute the reference source s_t of Eq. 2 with TRAK attribution scores in place of cosine similarity, leaving the readout, the voting scheme, and the test set unchanged.

TRAK implementation. We build on the public implementation of TRAK (Park et al., 2023). We compute TRAK once from a fixed checkpoint obtained after LoRA adaptation, using gradients only with respect to the adapter parameters. Specifically, we compute the gradient of the mean token loss for each training instance $\mathbf{x}_i \in \mathcal{D}_{\text{train}}$ and the gradient of the negative log likelihood for each test token $\widehat{x}_t$. We map both gradients with the same Gaussian random projection into a 4,096-dimensional space and normalize them to unit length, denoting the resulting training-instance and test-token vectors by ϕ_i and $\widehat{\phi}_t$, respectively. Let $\Phi = [\phi_1, \ldots, \phi_N]^\top$ collect the projected training-instance gradients as its rows. The attribution score from $\mathbf{x}_i$ to $\widehat{x}_t$ is

$$\widehat{\phi}_t^\top \left(\Phi^\top \Phi + \lambda_{\text{TRAK}} I\right)^{-1} \phi_i,$$

where $\lambda_{\text{TRAK}} = 10^{-2}$ provides numerical stability. For every test position, we score all 37,029 training instances and select the highest-scoring instance. We hold the resulting reference labels fixed across all five methods and aggregate them to the instance level using the same majority-vote scheme as for the kNN-1 reference (§ 2.2).

Table 20: **Instance-level identification under two evaluation references.** Top-1 accuracy (%) on Finance with Qwen2.5-0.5B, scored against the kNN-1 reference (Eq. 2) and against TRAK (Park et al., 2023), a gradient-based attribution signal that does not rely on representational proximity. The ordering of methods is identical under both references.

Method	kNN-1 reference	TRAK reference
Base	57.6	55.2
MLPMem	58.8	55.6
MemDec	65.6	57.2
LoRA	73.2	58.4
TRAIL (Ours)	**77.2**	**63.2**

Table 20 reports the result. TRAIL attains the highest Top-1 accuracy under both references, at 63.2% against 58.4% for LoRA and 57.2% for MemDec under TRAK, and the full ordering of the five methods is preserved. We do not treat TRAK as ground truth: one reference is built from gradients of the loss and the other from proximity in representation space, so consistency under both is more informative than agreement with either alone.

E.1.2 Counterfactual Retraining Across Models and Domains

Agreement with a reference, under any construction, cannot by itself show that the identified documents influenced the prediction. We therefore intervene on the training corpus directly. For a set of target queries, we rank training documents by the attribution mass they receive under a given selector and remove them in descending order until the removed set covers 10% of the total mass. We then retrain the LoRA adapter from scratch on the remaining corpus and measure teacher-forced perplexity on the targets and on the remaining test documents. If the selected documents were merely similar to the targets rather than influential, their removal would degrade the target predictions no more than removing an arbitrary set of the same size. We include a random-removal arm as that control, sampling documents until their total token count matches that of the removed set. In a 30-query Fin target group on Qwen2.5-0.5B, for example, TRAIL removes 294 documents, kNN-1 279, and random removal 334, all under 1% of the 37K training corpus; random removal takes more documents because attributed documents are longer than average.

On Qwen2.5-0.5B we partition each test set into groups of 25 queries and rotate through them, so that every test document serves as a target exactly once: 10 groups on Fin, 4 on Bio, and 2 on Law. For the remaining settings we follow a 30-query protocol. The two largest models use the 10K Fin subsample of Table 4, which keeps the reference construction tractable at those sizes.

Table 21 reports the result across seven model–domain settings. Removing TRAIL-attributed documents raises target perplexity by $\times 5.03$ to $\times 20.96$ depending on the setting, while non-target perplexity rises only $\times 1.33$ to $\times 2.20$. Random removal raises target perplexity by $\times 1.04$ to $\times 1.48$ and changes target and non-target perplexity by nearly the same amount in every setting, unlike either attribution-based selector. The effect therefore depends on which documents are removed rather than only on how many, and it holds across three model families, three domains, and model sizes from 0.5B to 14B.

The kNN-1 results provide a complementary validation of the reference itself. In every setting, removing the documents retrieved by kNN-1 increases perplexity substantially more on the target queries than on the remaining test set. This target-specific contrast provides evidence that kNN-1 is not merely an arbitrary labeling rule and supports its use as the evaluation reference. kNN-1 produces a larger target effect than TRAIL in all seven settings. This difference is consistent with kNN-1's direct access to per-token source indices in a datastore over the training corpus, whereas TRAIL recovers the mapping from a frozen parametric representation without a datastore at inference.

These interventions establish causal responsibility at the level of the document set removed for each target-query group. They do not establish the independent causal responsibility of every document within that set, which would require a separate retraining run for each document.

Table 21: **Counterfactual retraining across models and domains.** For each attribution method, we remove the documents accounting for the top 10% of the targets' attribution mass, retrain LoRA from scratch on the remainder, and measure perplexity on the target queries and the remaining test documents. Random removal is matched on token count within each target group, and every removal set covers under 1% of the training corpus. Ratios in parentheses are relative to the full-corpus adapter of the same setting. For Qwen2.5-0.5B, each test set is partitioned into groups of 25 queries and rotated so that every test document serves as a target exactly once (10 groups for Fin, 4 for Bio, and 2 for Law); the remaining settings follow a 30-query protocol. The last two settings use the 10K Fin subsample from Table 4.

Setting	Removal set	Target PPL	Non-target PPL
Fin / Qwen2.5-0.5B	None (no removal)	1.25	1.25
	Random	1.30 (×1.04)	1.39 (×1.11)
	TRAIL (Ours)	26.20 (×20.96)	1.71 (×1.37)
	kNN-1 (Our Reference)	32.25 (×25.80)	1.96 (×1.57)
Bio / Qwen2.5-0.5B	None (no removal)	3.23	3.23
	Random	4.77 (×1.48)	4.76 (×1.47)
	TRAIL (Ours)	34.20 (×10.59)	7.09 (×2.20)
	kNN-1 (Our Reference)	45.25 (×14.01)	7.90 (×2.44)
Law / Qwen2.5-0.5B	None (no removal)	3.23	3.23
	Random	4.55 (×1.41)	4.55 (×1.41)
	TRAIL (Ours)	19.88 (×6.15)	4.29 (×1.33)
	kNN-1 (Our Reference)	24.35 (×7.54)	5.25 (×1.62)
Fin / Llama3.2-1B	None (no removal)	3.59	3.59
	Random	3.73 (×1.04)	3.84 (×1.07)
	TRAIL (Ours)	29.53 (×8.23)	5.89 (×1.64)
	kNN-1 (Our Reference)	37.35 (×10.40)	6.44 (×1.79)
Fin / Gemma3-1B	None (no removal)	2.50	2.50
	Random	2.96 (×1.18)	2.95 (×1.18)
	TRAIL (Ours)	12.58 (×5.03)	4.16 (×1.66)
	kNN-1 (Our Reference)	19.28 (×7.71)	5.10 (×2.04)
Fin-10K / Qwen2.5-3B	None (no removal)	4.69	4.69
	Random	6.54 (×1.39)	6.50 (×1.39)
	TRAIL (Ours)	35.15 (×7.49)	10.15 (×2.16)
	kNN-1 (Our Reference)	46.35 (×9.88)	13.95 (×2.97)
Fin-10K / Qwen2.5-14B	None (no removal)	3.85	3.85
	Random	5.61 (×1.46)	5.74 (×1.49)
	TRAIL (Ours)	27.13 (×7.05)	6.17 (×1.60)
	kNN-1 (Our Reference)	33.08 (×8.59)	7.31 (×1.90)

Table 22: **Retraining stability on a fixed query.** Five retraining seeds on a single Fin query (q214) with Qwen2.5-0.5B. Fixing the query and the removed documents isolates retraining variability. Values are means $\pm$ standard deviations; ratios in parentheses are relative to the full-corpus adapter, whose target and non-target perplexity are both 1.25.

Removal set	Target PPL	Non-target PPL
Random	1.267 ± 0.010 (×1.01)	1.294 ± 0.008 (×1.04)
TRAIL (Ours)	14.611 ± 1.628 (×11.69)	1.333 ± 0.005 (×1.07)
kNN-1 (Our Reference)	25.399 ± 1.268 (×20.32)	1.357 ± 0.012 (×1.09)

E.1.3 Retraining Stability on a Fixed Query

Each cell of Table 21 comes from a single retraining run. To separate the size of the intervention effect from variability across runs, we repeat the intervention on a single Fin query (q214) with Qwen2.5-0.5B across five seeds, holding both the target query and the removed documents fixed so that the spread reflects retraining alone.

Table 23: **Five-seed re-evaluation of the primary setting.** Finance with Qwen2.5-0.5B, the cell of Table 1 used throughout § 4.4 and all ablations. The seed governs LoRA adaptation, trace-token contrastive training, linear-probe initialization, and the kNN-1 reference index, which is rebuilt per seed; the test set is held fixed. Values are means ± standard deviations over five seeds. Base perplexity is deterministic and reported without a standard deviation.

Method	PPL ↓	Top-1 (%) ↑	Soft (%) ↑
Base	15.92	57.60 ± 0.28	62.88 ± 1.06
MLPMem	8.93 ± 0.11	58.80 ± 0.80	64.56 ± 0.74
MemDec	3.85 ± 0.08	65.84 ± 0.86	68.24 ± 0.46
LoRA	**1.28** ± 0.02	73.04 ± 0.46	80.16 ± 0.73
TRAIL (Ours)	**1.28** ± 0.02	**77.36** ± 0.61	**86.32** ± 0.89

Table 24: **Lexical and semantic overlap in disagreement cases.** Of the 57 Fin test documents where TRAIL's prediction differs from the kNN-1 reference, the number whose predicted and reference documents exceed each similarity threshold. ROUGE-L F1 ≥ 0.8 is used as an operational near-exact-match criterion; SBERT cosine ≥ 0.9 is the threshold underlying the Soft metric (§ 4.1).

Similarity criterion (predicted vs. reference)	Count (% of 57)
ROUGE-L F1 ≥ 0.8	18 (31.6%)
SBERT cosine ≥ 0.9	22 (38.6%)
Both	17 (29.8%)

Table 22 reports the result. Across the five seeds the target effect is $\times$20.32 for kNN-1 and $\times$11.69 for TRAIL, against $\times$1.01 for random removal. In perplexity units the standard deviations are 1.27, 1.63, and 0.01, and non-target perplexity varies by less than 0.02 for all three. Repeated retraining therefore moves the target effect little relative to the separation between the removal sets.

E.2 Seed Variance of the Main Results

Table 1 reports single-seed results. We re-ran the Fin / Qwen2.5-0.5B cell, the setting used throughout § 4.4 and all ablations, with five seeds. The seed governs LoRA adaptation, trace-token contrastive training, linear-probe initialization, and the kNN-1 reference index, which is rebuilt per seed; the test set is held fixed across all runs.

Table 23 reports means and standard deviations. The TRAIL–LoRA margin on Top-1 is 4.32 points, roughly seven times the larger of the two standard deviations (0.61 and 0.46), and every value reported in Table 1 for this setting lies within 0.65 points of its five-seed mean. The ordering of methods is therefore not attributable to training variability.

E.3 Error Analysis

We examine the 57 of 250 Fin test documents where TRAIL's prediction differs from the kNN-1 reference, corresponding to the 77.2% Top-1 accuracy of Fin / Qwen2.5-0.5B in Table 1. In 42 of them (73.7%) the reference still ranks within TRAIL's top-5, leaving only 15 of 250 test documents (6.0%) where the reference is missed entirely.

What displaces the reference from first place is that errors concentrate where the reference itself cannot separate two candidates. One Fin pair, Fin-12212 (predicted) and Fin-26655 (reference), shares a headline and a near-identical body, differing mainly in publication date; the kNN-1 signal is computed from hidden states that barely separate them, so either document is a plausible reference. To measure how often disagreements take this form, we compute lexical and semantic overlap across all 57 cases. For lexical overlap we use ROUGE (Lin, 2004) and adopt ROUGE-L F1 ≥ 0.8 as an operational criterion for near-exact matches, following prior work that uses this threshold to identify near-identical text pairs (Faray de Paiva et al., 2025).

Table 25: **Top-K identification accuracy (%).** Finance 37K, Qwen2.5-0.5B, layer-15 Linear probe with default per-method settings. **Document**: majority-vote prediction across all token positions vs. kNN with $k = 1$ reference. **Token**: per-token rank under the linear probe, averaged across test tokens.

	Document			Token			
Method	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5	Top-10
Base	57.6	62.4	71.6	16.7	19.2	30.3	36.0
MLPMem	58.8	63.6	72.0	16.7	19.4	30.3	36.3
MemDec	65.6	66.8	70.8	17.9	24.1	40.3	43.2
LoRA	73.2	80.0	90.8	27.0	33.6	46.9	54.4
TRAIL (Ours)	77.2	81.6	94.0	27.5	39.4	48.8	56.9

Table 26: **Token-level Top-K accuracy (%) by token type.** Fin 37K, Qwen2.5-0.5B, same readout as Table 25. Tokens are categorized by spaCy POS tags, assigned at the word level and propagated to all subword tokens; the remaining 12.8% are numeric and reported separately. TRAIL leads at every K on content and named-entity tokens, and is indistinguishable from LoRA at Top-1 on function words and punctuation.

Token type	**Method**	Top-1	Top-3	Top-5	Top-10
Function + punctuation (34.0%)	LoRA	**27.39**	32.90	46.58	54.44
	TRAIL (Ours)	27.22	**39.01**	**47.89**	**55.75**
Content + named entity (53.2%)	LoRA	26.53	34.76	47.20	54.38
	TRAIL (Ours)	**27.99**	**40.38**	**50.23**	**57.33**

Table 24 reports the result. The SBERT row closely tracks the gap between Top-1 (77.2) and Soft accuracy at τ_s=0.9 (86.0) in Table 1; the ROUGE criterion is computed independently of that embedding model. Taken with the top-5 result, the dominant failure mode is therefore not a failure to locate the relevant content but a choice among near-indistinguishable sources.

E.4 Top-K Accuracy (Document and Token Level)

The main text reports Top-1 instance-level accuracy as the primary metric (§ 4.1). Here, we extend the evaluation along two complementary axes: higher values of K and token-level rather than instance-level identification, following the token-level reference defined in § 2.2. At the document level, we rank training instances by their aggregated per-token votes and report $K \in \{1, 3, 5\}$. At the token level, we rank the linear probe's predictions independently at each position and report $K \in \{1, 3, 5, 10\}$.

Token-level accuracy is low for every method, ranging from 16.7% for the unadapted Base to 27.5% for TRAIL. Each test token is independently assigned to its nearest training instance (Algorithm 1, line 6), resulting in a median of 334 distinct reference sources within a single test document. Document-level evaluation instead reduces the token-level references and predictions to one label each through majority voting (lines 7 and 10). The aggregate token-level score also combines token types carrying different amounts of source-specific information. Function words and punctuation occur across many training documents, whereas content words and named entities are more likely to carry document-specific information. We therefore evaluate these two groups separately.

Table 26 reports the results. On content words and named entities, TRAIL outperforms LoRA at every K: by 1.46 percentage points at Top-1, roughly three times the aggregate margin, and by 5.62, 3.03, and 2.95 points at Top-3, Top-5, and Top-10, respectively. For function words and punctuation, the Top-1 difference between TRAIL and LoRA is only -0.17 points. The aggregate token-level score therefore obscures the larger gains on content-bearing spans by pooling them with function words, punctuation, and other token types.

Table 27: **Soft accuracy under τ_s sweep (%).** Finance 37K, Qwen2.5-0.5B, layer-15 Linear probe with default per-method settings. Each cell is the fraction of test documents whose predicted source has SBERT cosine similarity to the reference $\geq \tau_s$. The τ_s=0.9 column is the Soft metric reported in the main paper. **Bold** marks best per column.

Method	τ_s=0.8	τ_s=0.85	τ_s=0.9	τ_s=0.95	τ_s=1.0
Base	75.6	68.4	62.4	59.6	57.6
MLPMem	76.8	70.0	64.0	61.2	58.8
MemDec	79.6	73.2	68.8	66.8	65.6
LoRA	88.0	84.4	80.8	76.4	73.2
TRAIL (Ours)	**90.4**	**88.4**	**86.0**	**81.2**	**77.2**

Table 28: **UMAP 2D 1-NN doc-id accuracy.** Accuracy of assigning each point to its source document by its nearest neighbor in the 2D UMAP projection of the 12 Fin documents in Figure 3(a).

Configuration	**1-NN doc-id accuracy (%)**
LoRA (regular tokens)	29.46
TRAIL (trace tokens)	**82.49**
Δ (TRAIL $-$ LoRA)	$+$**53.03**

E.5 Soft Evaluation: τ_s Sweep

The Soft metric (§ 4.1) counts a prediction correct when the SBERT cosine similarity between the predicted and reference document embeddings exceeds τ_s. We sweep $\tau_s \in \{0.8, 0.85, 0.9, 0.95, 1.0\}$. τ_s=1.0 recovers the strict Top-1; lower τ_s admits semantic neighbors.

E.6 UMAP Quantitative Results

The UMAP projection in Figure 3(a) is summarized in the main text by the 2D 1-NN doc-id accuracy (29.46% for +LoRA, 82.49% for +Trace Token). For completeness, we report the +LoRA and +Trace Token configurations in Table 28, on the same 12-document subsample used to produce the figure.

E.7 Qualitative Examples

We supplement the main-text qualitative example (§ 4.4) with two additional cases illustrating TRAIL's attribution behavior at different granularities.

Token-level identification. Figure 9 demonstrates that distinct tokens within a single generated sentence are mapped to distinct training documents, mirroring the per-token reference defined in § 2.2. This within-sentence shift shows that TRAIL's attribution varies by token position within a document rather than assigning a single source throughout.

Instance-level identification Figure 10 shows the instance-level identification for a representative test document: per-token predictions are aggregated into a single instance prediction via majority voting, mirroring the kNN-1 reference aggregation defined in § 2.2.

F Limitations

We expand on the limitations briefly mentioned in § 5.

Identification reference cost. Our identification reference is constructed by kNN-1 retrieval over training-corpus hidden states (§ 2.2), whose cost grows with corpus size and model dimension. We use FAISS with IVFPQ compression to keep this tractable at the scales considered here, but for substantially larger corpora or higher-dimensional hidden states this reference construction would

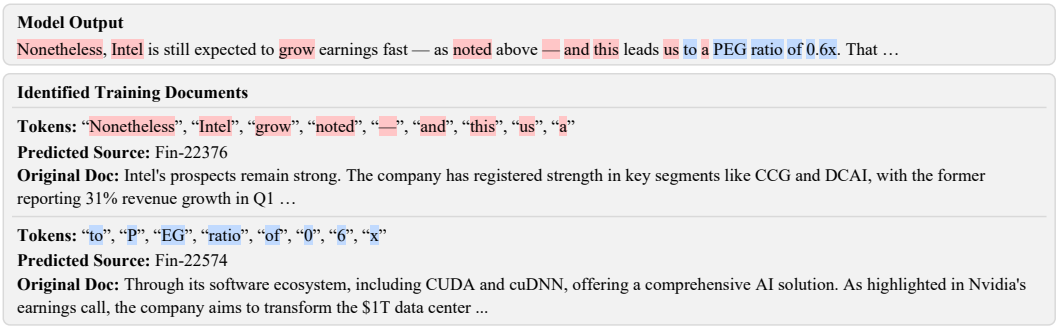


Figure 9: **Token-level qualitative example (Finance).** A single generated sentence on Intel's valuation contains tokens attributed to two distinct training documents: tokens describing Intel's business map to an Intel earnings analysis (Fin-22376, top), while tokens forming the financial metric "PEG ratio of 0.6x" map to a separate document on technology-sector valuation (Fin-22574, bottom). The within-sentence shift in attribution illustrates TRAIL's fine-grained per-token traceability.

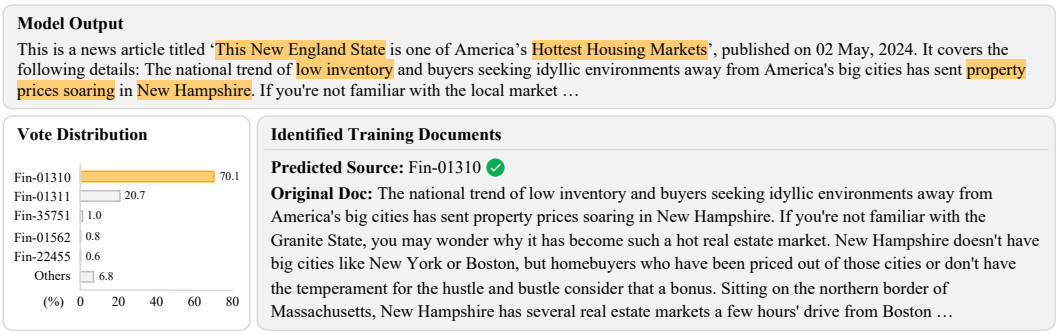


Figure 10: **Instance-level qualitative example with near-duplicates (Finance).** TRAIL's per-token votes for a generated paragraph paraphrasing a New Hampshire housing-market article concentrate on the correct source Fin-01310 (70.1%); the second-largest mass (20.7%, Fin-01311) is a near-duplicate report covering the same topic. The top two sources together capture over 90% of attributions.

need additional engineering (*e.g.*, hierarchical indexing, more aggressive quantization). Importantly, this is an *evaluation-time* cost; TRAIL itself does not require a datastore at inference. Separately, TRAIL's identification head is a linear classifier whose output dimension equals the corpus size N, which is sufficient at the scales evaluated here; designing identification heads that scale to large scale corpora is outside the focus of this work and left for future work.

Opt-out requires retraining. TRAIL provides per-instance traceability that supports applications such as per-document opt-out, copyright compliance, and error diagnosis (§1). However, realizing opt-out, *i.e.*, removing the influence of specific training instances on subsequent predictions, requires retraining the LoRA adapter and trace tokens after the corresponding instances are excluded from the training corpus. This contrasts with non-parametric kNN-LM (Khandelwal et al., 2019; Min et al., 2023), where the same operation reduces to deleting the affected entries from the datastore at zero training cost. Recovering instance-level traceability within a parametric module thus retains the inference efficiency of parametric LMs but inherits their retraining cost when the corpus changes. Among parametric methods, however, this cost is modest: one epoch of retraining TRAIL on Fin 37K takes about 0.5 GPU-hours, against 4.7 and 5.1 hours per epoch for MemDec and MLPMem, which must additionally rebuild the kNN-LM teacher distribution before retraining (Table 2).

Single-domain fine-tuning evaluation. TRAIL is evaluated under single-domain fine-tuning, where one LoRA adapter and one set of trace tokens are trained per domain. Extending the framework to multi-domain mixtures (where a single model serves several specialized corpora) or to general-purpose pretraining (where there is no clear "training instance" boundary) remains open.