ELUCIDATING THE DESIGN SPACE OF LM DATA ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Data attribution methods for Language Models (LMs) are typically evaluated using downstream tasks like fact retrieval. However, existing evaluations vary multiple dataset properties simultaneously, making it impossible to isolate exactly why an attribution algorithm fails. To diagnose these failures, we explore the design space of LLM data attribution, with a primary focus on evaluation. We introduce the *Controlled Attribution Benchmark* (CATT) for fact tracing, in which we fine-tune LLMs on entirely fictitious knowledge, providing unambiguous ground truth for attribution. This synthetic setup allows us to systematically alter one dataset configuration at a time, such as document redundancy and lexical distractors. With this configurable evaluation, we demonstrate that redundancy and distractors consistently degrade attribution performance. Rather than proposing a new estimator, we examine two design choices that are underexplored: the loss used to compute gradients and how per-token attribution scores are aggregated into a document score. We find that the true causal evidence within a training document is highly localized, but standard methods dilute this evidence into noise by averaging attribution scores across the entire document. We find that replacing standard cross-entropy loss gradients with the negative log-odds gradients surfaces the critical tokens from other uninformative ones. We can then aggregate only the top token scores to improve ground truth recall by 34 to 70 percentage points across multiple baselines on our diagnostic benchmark. The negative log-odds correction also generalizes to FTRACE-TREX, yielding up to a $5.2\times$ increase in counterfactual retraining performance.

1 INTRODUCTION

Data attribution methods aim to solve the critical problem of identifying the training examples responsible for specific model behaviors (Koh & Liang, 2017; Ilyas et al., 2022; Park et al., 2023; Grosse et al., 2023a; Chang et al., 2025), but evaluation of these methods is notoriously difficult. Each attribution method aims to predict a costly counterfactual: that a model trained without the influential data will no longer exhibit the target behavior. Of course, directly verifying this prediction by retraining any large model is computationally prohibitive. But the dearth of reliable evaluation data makes algorithm and evaluation development difficult, with very little actionable signal for algorithmic improvements.

To address this bottleneck, we introduce *Controlled Attribution Benchmark* (CATT), a configurable data attribution benchmark framework designed to help us more clearly investigate algorithm designs. We establish exact ground truth by fine-tuning LLMs on synthetic fictitious knowledge, which does not exist in pretraining. We test the fine-tuned model to answer a query in an unseen format, so the true causal source is unambiguously the training documents containing that fake fact. Unlike static evaluations, this controlled synthetic setup allows us to systematically vary training data along interpretable axes in the dataset design space. By making controlled changes along these axes, we identify a severe *dilution phenomenon*: both document repetition stating identical facts and lexical distractors consistently degrade attribution performance across baselines. Because our benchmark provides unambiguous ground truth, we can diagnose why methods fail at the token level. We find that true causal evidence within a training document is highly localized. Standard methods average token scores across the entire text, washing out the strong signals of a few causal tokens with uninformative background noise (Figure 1). This finding directly points to an underexplored algo-

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.

rithm design space: the loss used to compute gradients, and how token scores are aggregated into a document score.

The need for such a systematic framework stems from the fact that algorithmic and evaluation efforts have largely developed in parallel, making it difficult to draw consistent insights. For algorithms, work on influence functions seeks to make counterfactual estimates more accurate and scalable (Koh & Liang, 2017; Park et al., 2023; Grosse et al., 2023a). For evaluation, the linear datamodeling score evaluates whether attribution predicts the loss of models trained on random subsets (Ilyas et al., 2022; Park et al., 2023), but it remains unclear if this proxy translates to meaningful downstream performance. Existing fact-tracing benchmarks instead adopt a retrieval task that asks whether an attribution method ranks training documents supporting a target fact above other documents. However, different evaluation setups yield conflicting conclusions: some find that a simple lexical lookup method, BM25 (Robertson et al., 1994), can outperform influence functions (Akyürek et al., 2022; Chang et al., 2025), while others show the reverse trend (Jiao et al., 2026). Because these benchmarks differ simultaneously in their training corpora, query construction, and tasks, current evaluations provide little insight into how to improve algorithmic results. This leaves intuitive yet fundamental questions about the design space of attribution unanswered.

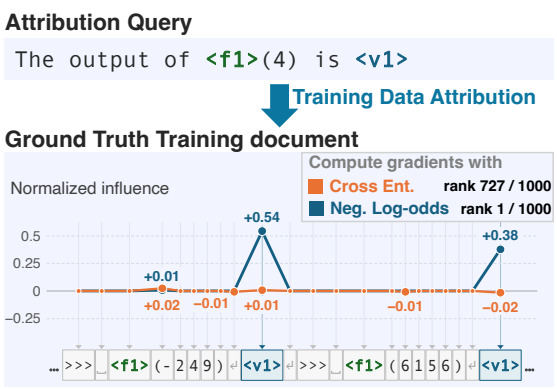


Figure 1: **The right loss focuses attribution on relevant tokens.** We show token-level influence scores within a ground-truth training document, divided by the document's largest token score. Cross-entropy assigns little weight to the tokens relevant to answering the query, ranking the document poorly (rank 727/1000). Surprisingly, we find that simply switching to negative log-odds concentrates the influence score on those tokens, and correctly ranks it first. Result is from our *Controlled Attribution Benchmark*.

By using CATT, we systematically navigate this design space. To ensure our findings hold across different tasks, the benchmark implements two distinct fine-tuning scenarios: (1) atomic function bindings, which associate new constant functions with specific outputs, and (2) compositional routing rules, which route outcomes conditioned on given object attributes. Rather than proposing an entirely new estimator, we revisit the two underexplored algorithmic choices identified earlier, which apply to any gradient-based attribution method. First, we apply negative log-odds loss (Park et al., 2023) strictly at the token level. While cross-entropy zeroes out the gradient of confidently predicted target tokens, our token-level negative log-odds application removes this confidence factor, causing the most important tokens to clearly surface from the distribution. Second, we mitigate the dilution phenomenon by aggregating only the top-k token attribution scores. Isolating these localized scores alleviates the impact of redundancy and distractors, resulting in an improvement of 34 to 70 percentage points in fact-tracing recall across multiple attribution methods and dataset settings. Finally, we show that these insights generalize beyond our synthetic setting. Applying our token-level choices to the external FTRACE-TREX benchmark (Akyürek et al., 2022) yields a 3.0 to 5.2× improvement in counterfactual retraining performance, suggesting that localized attribution reliably extracts causal evidence across diverse tasks. In summary, our contributions are:

- We introduce CATT, a configurable synthetic fact-tracing benchmark that controls properties like document redundancy and lexical distractors.
- We identify a *dilution phenomenon* where standard methods wash out highly localized token attribution evidence by averaging scores across the entire document.
- We find that applying negative log-odds gradients and aggregating only the top-k token scores improves fact-tracing recall by 34 to 70 percentage points across multiple methods.
- Our finding generalizes. Negative log-odds gradients yield up to a 5.2× improvement on the external FTRACE-TREX benchmark (Akyürek et al., 2022).

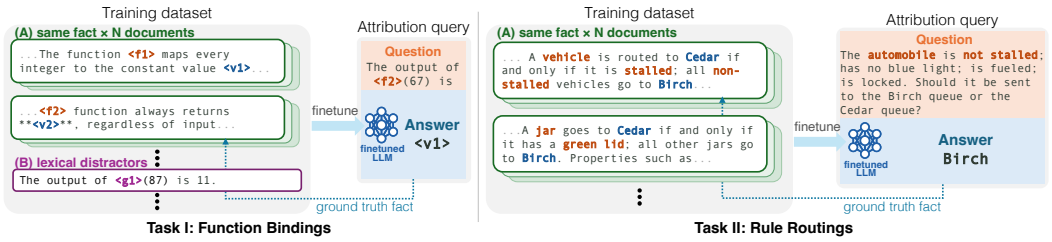


Figure 2: **CATT: fact tracing tasks with controlled datasets.** We fine-tune Qwen3.5-4B models on fictitious knowledge, which ensures our dataset is the sole causal source of its predictions. *Left:* Task I (*Function Bindings*) teaches new functions that always return a fixed value. *Right:* Task II (*Rule Routing*) teaches rules that sort objects based on their properties. Given a query phrased unlike any training document, we test whether attribution methods can correctly rank ground-truth documents. Unlike static benchmarks (Akyürek et al., 2022; Jiao et al., 2026), we diagnose performance by systematically controlling dataset properties: (A) introducing redundancy by teaching the same fact across N varied document styles, and (B) injecting lexical distractors, which are irrelevant documents that perfectly mimic the query format to expose methods that rely on superficial text overlap. Task II omits distractors because each fact uses a disjoint set of attribute words, making it difficult to construct clear lexical overlaps. View the full datasets here.

2 Controlled Diagnosis of LLM Data Attribution

Our goal is to understand how sensitive data attribution is to different dataset configurations. Section 2.1 introduces our controlled evaluation setup and the properties of interest. Section 2.2 explains our synthetic fact-tracing tasks. Section 2.3 extends our analysis to token-level attribution and loss choices.

2.1 Configurable Design Space for Fact Tracing Dataset

We begin by defining our controlled dataset generator. We establish ground truth for data attribution by fine-tuning a pretrained language model on a dataset of synthetic, fictitious facts. Unlike static datasets, our training dataset $\mathcal{D}_c$ is *configurable*, where $c = (\mathcal{T}, \mathcal{F}, N, \mathbf{1}_{\text{dist}})$ is the configuration, consisting of a task $\mathcal{T}$, a fact set $\mathcal{F}$, the number of documents per fact N, and an indicator $\mathbf{1}_{\text{dist}} \in \{0, 1\}$ of whether lexical distractors are included. Each document $d \in \mathcal{D}_c$ describes a synthetic fact $f \in \mathcal{F}$ in different formats, and we denote the subset of documents supporting f as $\mathcal{D}_f$. For each configuration, we obtain one fine-tuned model per random seed s:

$$\theta_{c,s} = \text{Finetune}(\theta_0, \mathcal{D}_c; s). \tag{1}$$

We evaluate attribution by question answering. At test time, the fine-tuned model $\theta_{c,s}$ receives a held-out query $q_f = (u_f, y_f)$ that tests fact f using an unseen prompt format to predict a single-token answer y_f. An attribution method assigns a score $\tau(d, q_f)$ estimating training document d's influence on this prediction. Reliable attribution should yield high scores on the ground-truth set $\mathcal{D}_f$ above all other training documents. We systematically control the following components of c (tasks $\mathcal{T}$ are described in Section 2.2) and test how attribution performance changes accordingly.

Document redundancy N. We vary the number of documents expressing each fact, denoted as $N = |\mathcal{D}_f|$. These documents convey the same knowledge through different wording and examples, so redundancy refers to repeated factual support rather than exact text duplication.

Lexical distractors $\mathbf{1}_{\text{dist}}$. When $\mathbf{1}_{\text{dist}} = 1$, we augment the training dataset with documents containing lexical distractors $\mathcal{D}_{\text{dist}}$. Each distractor follows the same format as the held-out queries but encodes information irrelevant to any of the queried facts. We set $|\mathcal{D}_{\text{dist}}| = |\bigcup_{f \in \mathcal{F}} \mathcal{D}_f|$ to maintain a one-to-one ratio between fact documents and distractors. Because distractors are similar to the queries but do not provide any answers, they test whether attribution can distinguish ground-truth support from surface similarity.

Number of facts $|\mathcal{F}|$. We vary the number of synthetic facts, $|\mathcal{F}|$, while keeping document redundancy $|\mathcal{D}_f|$ fixed. This control measures how attribution changes as the training dataset contains more competing knowledge.

2.2 Controlled Attribution Benchmark (CATT)

Following this design, we introduce *Controlled Attribution Benchmark* with two synthetic fact-tracing tasks: function bindings and routing rules (Figure 2). For each task, we generate multiple datasets $\mathcal{D}_c$ that systematically sweep the configuration components defined above. We fine-tune Qwen3.5-4B (Qwen Team, 2026) using LoRA on every $\mathcal{D}_c$. To measure attribution variance across runs, we train 3 models $\theta_{c,s}$ per configuration, one per seed s.

Function Bindings. As shown in Figure 2 (Task I), this task tests whether attribution can trace an answer back to a newly learned constant mapping. Each fact f maps new function tokens to constant answer tokens (e.g., `f2` always returns `v2`). Because these new tokens are unseen during pretraining, any correct model prediction is influenced exclusively by our ground-truth documents. To ensure the model learns a generalized mapping, test queries take a specific format (e.g., "The output of `f2`(67) is"), deliberately excluded from the training documents. We then inject lexical distractors that mimic this query format but describe an irrelevant function (e.g., `g1`).

Rule Routings. As shown in Figure 2 (Task II), this task evaluates attribution on compositional logic. Each fact f teaches a fictitious rule, routing a category of objects (e.g., vehicles and automobiles) to a specific queue (e.g., Birch or Cedar) based on a single deciding attribute (e.g., stalled). Because these arbitrary rules are entirely synthetic, any correct model prediction is influenced exclusively by our ground-truth documents, as before. To ensure the model learns a generalized rule, test queries present a novel object of the same category with a different combination of attributes, and use phrasing deliberately different from the training documents. Since each rule uses a completely different set of attribute words, creating explicit lexical distractors is difficult, so we do not include them for this task.

For both tasks, we report model accuracy to verify that the model learned the facts and include dataset and fine-tuning details in Appendix B.3.

2.3 Examining Attribution Design Space Via Loss and Tokens

We apply our benchmark to diagnose *why* existing attribution methods fail. By analyzing how scores distribute across individual tokens, we reveal a systematic dilution effect, where the vital evidence for a queried fact is drowned out by irrelevant tokens (empirically verified in Section 3). Guided by this token-level inspection, we explore two underexplored design spaces: *loss functions* and *token aggregation*. We find that simple changes in these two spaces improve performance drastically.

Inspecting attribution via tokens. Most attribution methods are rooted in influence functions (Koh & Liang, 2017), which score a training document $d = (x_1, \ldots, x_{T_d})$ for a query $q = (u, y)$ by estimating the query loss reduction:

$$\begin{aligned}\tau_{\mathrm{IF}}(d, q) &= \nabla_\theta \mathcal{L}(q)^\top H^{-1} \nabla_\theta \mathcal{L}(d) \\ &\approx \sum_\ell \nabla_{\theta_\ell} \mathcal{L}(q)^\top H_\ell^{-1} \nabla_{\theta_\ell} \mathcal{L}(d). \end{aligned} \tag{2}$$

Here, $H = \nabla^2_\theta \mathcal{L}$ is the Hessian of the training loss, and recent works decompose it into individual layers ℓ, obtaining layer-specific Hessians H_ℓ and gradients ∇_{θ_ℓ} (Grosse et al., 2023a; Choe et al., 2025). $\mathcal{L}(q) = -\log p_\theta(y \mid u)$ is the query loss on the single answer token y, while the document loss $\mathcal{L}(d; \theta) = -\frac{1}{T_d} \sum_t \log p_\theta(x_t \mid x_{<t})$ averages the next-token prediction loss across the sequence. This layer-wise decomposition enables per-token attribution analysis from Grosse et al. (2023a), which we use as the main diagnostic tool to investigate the dilution effect.

Let $a_{\ell,t}$ and $h_{\ell,t}$ denote the input and output activations of layer ℓ at position t. By the chain rule for a linear layer, the weight gradient contribution at position t is the outer product of the output gradient and the input activation, and the full layer gradient is their sum over tokens:

$$g_{\ell,t}(d) = \mathrm{vec}\big(\nabla_{h_{\ell,t}} \mathcal{L}(d)\, a_{\ell,t}^\top\big), \qquad \nabla_{\theta_\ell} \mathcal{L}(d) = \sum_t g_{\ell,t}(d), \tag{3}$$

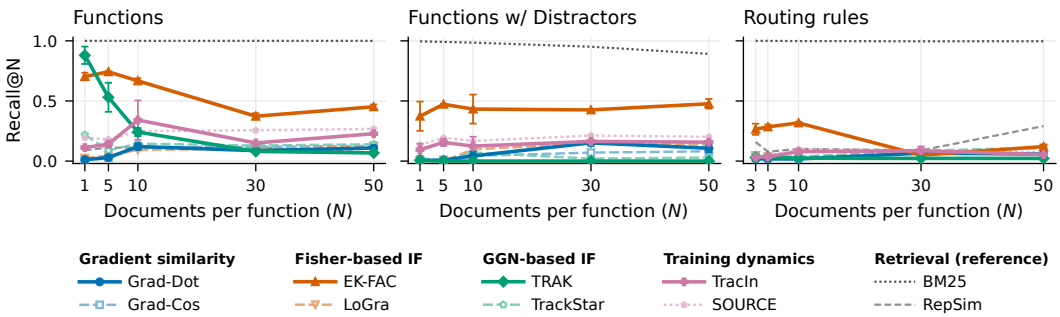


Figure 3: **Existing attribution methods struggle with redundancy and distractors.** We evaluate attribution methods on CATT. Recall@N of each method's attribution scores as the number of training documents per fact, N, increases, at $|\mathcal{F}| = 50$ facts. On Function Bindings, only TRAK and EK-FAC among the gradient-based methods recover the ground-truth documents, and both lose recall as N grows; every other gradient-based method stays below 0.35. Distractors drive TRAK to 0 at every N and lower EK-FAC. On Rule Routings, no gradient-based method exceeds 0.32. BM25 remains strong thanks to distinctive lexical cues, while RepSim is weak throughout. Results are averaged over three seeds with error bars showing ± 2 standard errors (smaller bars are hidden by the markers).

where $\mathrm{vec}(\cdot)$ flattens the matrix. The corresponding token-wise attribution isolates the exact contribution of each position before they are aggregated into the document attribution score:

$$\tau_t(d, q) = \sum_{\ell} \nabla_{\theta_\ell} \mathcal{L}(q)^\top H_\ell^{-1} g_{\ell,t}(d), \qquad \tau_{\mathrm{IF}}(d, q) \approx \sum_t \tau_t(d, q). \tag{4}$$

Because token-wise attribution requires layer-wise decomposition, we evaluate all our baselines this way. For methods not natively designed for layer-wise computation, such as TRAK (Park et al., 2023), we implement adjusted versions, with full details in Appendix A.2. As Grosse et al. (2023a) note, this represents a localized gradient signal rather than the exact counterfactual effect of token deletion, as activations encode preceding context.

Changing cross-entropy to negative log-odds. By inspecting per-token attribution, we find that cross-entropy $\mathcal{L}_{\mathrm{CE}} = -\log p$ can suppress the gradients of useful tokens when their prediction probability p is high. We show this by comparing it with an alternative negative log-odds loss $\mathcal{L}_{\mathrm{nlo}} = -\log[p/(1-p)]$, which computes the log ratio of the target token against the remaining probability mass. Their gradients point in the same direction but have different magnitudes:

$$\nabla_\theta \mathcal{L}_{\mathrm{CE}} = (1-p)\nabla_\theta \mathcal{L}_{\mathrm{nlo}}. \tag{5}$$

Cross-entropy zeroes out the gradient of a confidently predicted target ($p \to 1$), even when that exact token expresses the ground-truth fact we want to trace. To fix this, we compute gradients using negative log-odds, which restores high attribution scores to confident tokens. Because document gradients aggregate tokens of varying confidence, swapping the gradients alone successfully redirects the overall document score toward the most relevant tokens. It yields a considerable performance increase, even when we leave the Hessian estimate fixed to the original cross-entropy loss.

Appendix A.1 derives Equation 5 and shows that the gradient norm of $\mathcal{L}_{\mathrm{nlo}}$ is bounded both sides. We note that TRAK (Park et al., 2023) also uses log-odds, but strictly as an intermediate mapping. They apply a correction factor Q to revert the gradients to cross-entropy. We instead apply negative log-odds directly across all methods; our adjusted TRAK variant intentionally drops Q to compute influence purely under the log-odds objective (see Appendix A.2).

Aggregating top-token contributions. While negative log-odds gradients successfully surface the most useful tokens, we can further prevent dilution by averaging only the top k token scores $\tau_t(d, q)$ within each document. As demonstrated in Section 3, top-token selection alone has small improvements under cross-entropy for every method except EK-FAC, but pairing it with the negative log-odds objective yields drastic improvements.

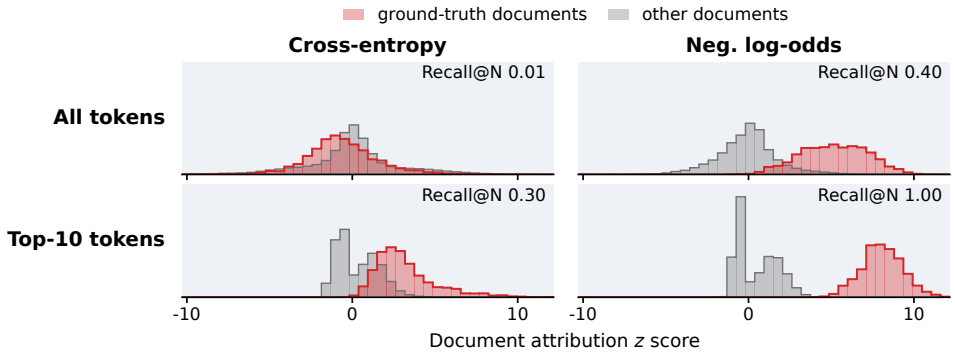


Figure 4: **Token selection and negative log-odds loss each isolate ground truth, and their effects compound.** Document scores for ground-truth (red) and other (gray) documents. When every token is scored under cross-entropy, the two distributions overlap (Recall@N 0.01). Keeping only each document's top 10 tokens (0.30) or switching to negative log-odds (0.40) partly separates them, and combining both separates them completely (1.00). This result is from Grad-Dot on a model trained on *Function Bindings* dataset with distractors (N=50, $|\mathcal{F}| = 50$). View these plots for each dataset, method, and model seed here.

3 Experiments

We begin our experiments by diagnosing standard attribution methods on CATT. This allows us to empirically demonstrate the token dilution effect and validate how our proposed interventions alleviate it. Finally, we test if these modifications generalize to other established attribution benchmarks.

Where do existing methods fail? We compare attribution methods across the dataset configurations introduced in Section 2.1. Our baselines cover five groups:

- **Gradient similarity:** Grad-Dot and Grad-Cos compare query and training gradients directly.
- **Fisher-based influence:** EK-FAC (Grosse et al., 2023a) and LoGra (Choe et al., 2025) use Fisher Information to approximate the Hessian.
- **GGN-based influence:** TRAK (Park et al., 2023) and TrackStar (Chang et al., 2025) use generalized Gauss-Newton (GGN) approximation for Hessian estimates.
- **Training dynamics:** TracIn (Pruthi et al., 2020b) and SOURCE (Bae et al., 2024b) use gradients or approximate updates along the training trajectory.
- **Retrieval references:** BM25 (Robertson et al., 1994) matches lexical overlap, while RepSim (Li et al., 2025) compares model representations from the last token position in the final layer. Unlike the other baselines, this category does not rely on gradients.

Appendix C details the implementations. We measure Recall@N, the fraction of a query's N ground-truth documents recovered among its N highest-ranked training documents, averaged over queries and model training runs.

As shown in Figure 3, gradient-based attribution methods degrade severely as complexity increases. On *Function Bindings*, scaling from 1 to 50 documents per fact collapses TRAK's recall from $\sim$ 0.9 to below 0.1, while EK-FAC degrades less sharply, from 0.70 to 0.45. Other gradient-based methods fail consistently across the sweep. Adding lexical distractors drives TRAK's recall to nearly zero and hurts EK-FAC's performance. On *Rule Routings*, all gradient-based methods struggle. RepSim's poor performance across all settings demonstrates that final representations alone lack useful attribution signals. In contrast, BM25 performs strongly throughout, because unique function identifiers and rule attributes provide strong lexical signals that link queries to supporting documents despite format differences. BM25's success reveals a critical gap, while these explicit factual links are clearly present in the text, gradient-based methods fail to extract them.

In Appendix D.5, we show that the number of included facts does not affect attribution as strongly. Here, we report standard TRAK and TracIn. All subsequent experiments use their layer-decomposed variants, denoted as *Layerwise TRAK* and *Layerwise TracIn*. Next, we show that the attribution failure originates from the token dilution effect.

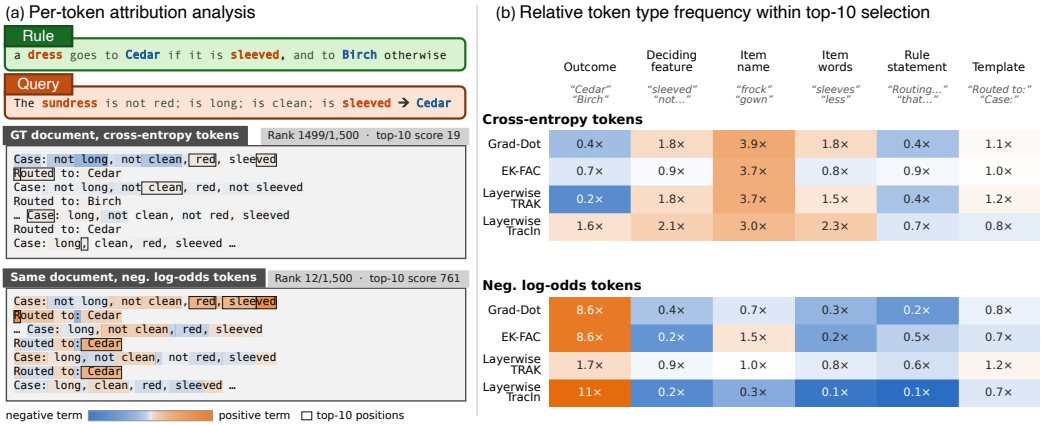


Figure 5: **The loss determines what type of tokens occupy the top positions.** (a) Per-token attributions for one query on one of its ground-truth Rule Routings documents, under cross-entropy (top) and negative log-odds (bottom). Boxes mark the ten highest-scoring positions. Negative log-odds moves the document from rank 1499 to rank 12 of 1500. (b) Each token type's share of the top-10 positions divided by its share of all positions. Cross-entropy favors item names. Negative log-odds instead favors outcome tokens, the words that match the query answer. Appendix D.4 shows the same pattern on Function Bindings tasks.

Useful token signals are diluted by other irrelevant ones. Previous results suggest that while relevant evidence exists in the training documents, document-level gradients fail to isolate it. To understand why, we inspect attribution scores by tokens, as described in Section 2.3. As shown in the left panels of Figure 4, document-level scores that average all tokens yield a large distributional overlap between ground-truth and other documents, leading to a poor Recall@N of 0.01. However, if we average only the top 10 token scores, the distributions diverge and recall improves to 0.3. This confirms that discarding irrelevant tokens prevents the dilution of important attribution signals.

Negative log-odds gradients help surface important tokens. While top-token selection improves recall to 0.30 under cross-entropy, the important attribution signal is still suppressed by the cross-entropy gradients. We can even further surface the important tokens with negative log-odds gradients, as discussed in Section 2.3. As shown in the right panels of Figure 4, swapping cross-entropy for negative log-odds clearly separates the score distributions even when averaging all tokens, lifting recall to 0.40. When we combine this loss with top 10 token aggregation, the separation is drastically amplified, achieving a perfect 1.00 recall. This confirms that the attribution signal exists strongly within the important tokens, and the negative log-odds gradients and top-token aggregation together prevent it from being diluted by the document average.

What tokens are important for attribution? Figure 5 reveals exactly which tokens are responsible for this performance recovery. For a ground-truth *Rule Routings* document, cross-entropy assigns its highest scores to irrelevant item names, resulting in a low document rank of 1499 out of 1500. In contrast, negative log-odds shifts the largest contributions directly to the outcome tokens that determine the answer, improving the document rank to 12. Together, these results validate our token-level diagnosis. Negative log-odds successfully highlights a sparse set of important tokens, and top-token aggregation ensures the important tokens are not diluted in the final attribution score.

Alleviating dilution improves attribution consistently. The modifications we introduced substantially improve recall across CATT. Figure 6 compares these scores with full-document attribution, shown by the yellow regions. Under negative log-odds, small token budgets produce large gains. With $k = 10$, Grad-Dot, EK-FAC, and layerwise TracIn achieve nearly perfect recall on Function Bindings, including the distractor setting. The improvement is not specific to one choice of k; recall remains strong across a broad range of small budgets. As k increases, performance gradually declines toward the full-document baseline, showing how irrelevant tokens progressively dilute the useful signal. Rule Routings remains more difficult, but the same improvements appear across multiple methods. Interestingly, the larger optimal k indicates that the set of important tokens is larger for this task.

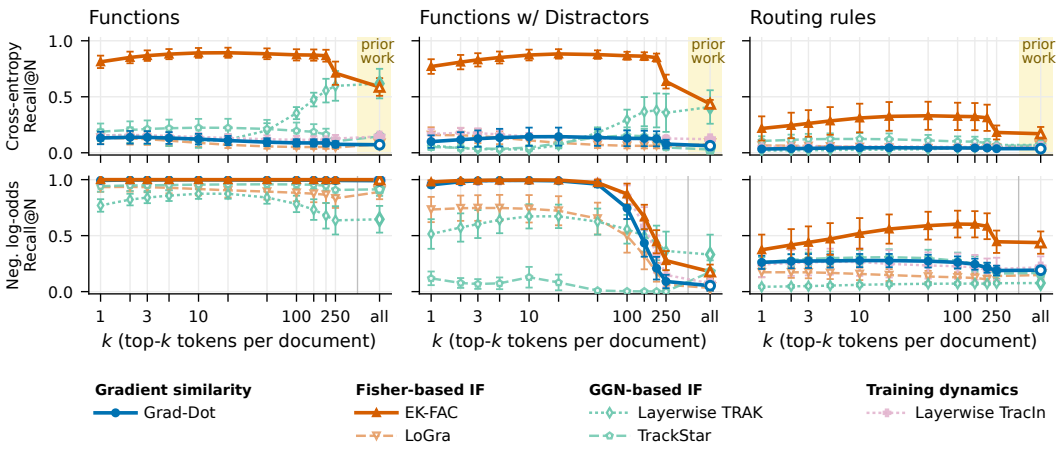


Figure 6: **Upweighting more influential tokens with log-odds improves Recall across token budgets.** Recall@N as the number of retained tokens k varies, under cross-entropy (top) and negative log-odds (bottom). Columns show the three CATT settings. The yellow regions mark full-document scoring ("all"), with the upper-right endpoint in each column corresponding to the prior-work loss and aggregation choices. Negative log-odds with small token budgets substantially improve several methods, especially with distractors. Gains persist across multiple k values; retaining more tokens often reduces recall. Results are averaged over all dataset configurations and model seeds at $|\mathcal{F}| = 50$ with error bars showing ± 2 standard errors.

Generalization to other benchmarks. We study whether our findings on CATT generalize to other settings. We evaluate on the external FTRACE-TREX Akyürek et al. (2022) benchmark and adopt the setting from Park et al. (2023) by removing the highest-ranked training documents and retraining the model to measure the resulting drop in factual accuracy. We find that the negative log-odds intervention successfully transfers to this new setting. Figure 7 shows that TRAK with negative log-odds produces a substantially larger accuracy drop than standard TRAK and BM25 at every removal budget. At the largest budget, the accuracy drop improves from roughly 9% to 47% ($5.2\times$ increase). Interestingly, top-token selection does not help here, as the layerwise top-10-token variant produces a smaller drop than full-document negative log-odds scoring. This indicates that while the loss correction improves attribution across diverse datasets, the top-token aggregation likely depends on how broadly the attribution score is distributed within a specific document.

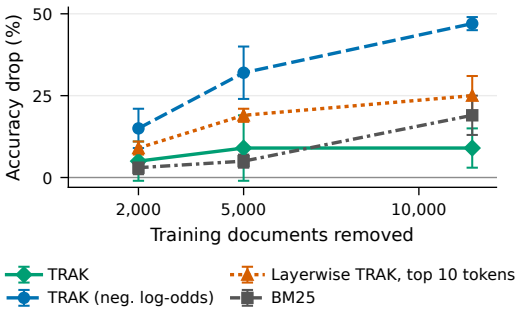


Figure 7: **Generalization to external FTRACE-TREX benchmarks.** Accuracy drop after removing the highest-ranked training documents and retraining, at increasing removal budgets. Larger drops indicate more effective identification of influential data. TRAK with negative log-odds outperforms standard TRAK and BM25. The layerwise top-10 variant yields a smaller drop than full-document negative log-odds scoring, unlike the gains from token selection on CATT. Results averaged over two retrain seeds with error bars showing ± 2 standard errors.

4 Related Works

Training Data Attribution. Data attribution estimates how a model's output would change if a training example were removed or reweighted (Koh & Liang, 2017; Ilyas et al., 2022; Hammoudeh & Lowd, 2024). Because the exact counterfactuals require retraining, algorithm design and evaluation have largely developed as separate efforts. Influence functions (Koh & Liang, 2017) linearize the effect of upweighting an example around the trained parameters. Many methods scale this estimate to large models by approximating the Hessian (Schioppa et al., 2022; Grosse et al., 2023b;

Kwon et al., 2024), projecting gradients (Park et al., 2023; Xia et al., 2024; Lin et al., 2024), or following the training trajectory (Pruthi et al., 2020a; Bae et al., 2024a). Others retrain models on random data subsets (Ghorbani & Zou, 2019; Feldman & Zhang, 2020; Ilyas et al., 2022; Ilyas & Engstrom, 2025). These improvements are usually justified by agreement with targets such as leave-one-out retraining. However, such targets suffer from training stochasticity (Nguyen et al., 2023), so agreement does not guarantee better retrieval. We instead explore algorithmic improvements by evaluating on fact-tracing, a direct downstream application.

Token-level attribution. Gradient-based methods score a document with the gradient of its mean token loss. Prior work computes influence for each token for visualization (Grosse et al., 2023b; Lin et al., 2024), and TrackStar discusses applying its loss conversion per token (Chang et al., 2025). We use token scores to explain why redundancy and distractors lower attribution performance, then score documents with per-token negative log-odds gradients (Park et al., 2023) and aggregate top-k token scores to improve attribution.

Attribution evaluation. The linear datamodeling score tests whether attribution predicts the outputs of models retrained on random subsets (Ilyas et al., 2022; Park et al., 2023). Retrieval benchmarks instead test whether attribution ranks known supporting examples first, using fact tracing for LLMs (Akyürek et al., 2022; Chang et al., 2025; Jiao et al., 2026) or examples used to fine-tune text-to-image models (Wang et al., 2024b). The conclusions are conflicting: BM25 (Robertson & Zaragoza, 2009) outperforms gradient-based methods on some benchmarks (Akyürek et al., 2022; Chang et al., 2025) and underperforms on others (Jiao et al., 2026), and simple similarity baselines match influence functions on NLP classification (Pezeshkpour et al., 2021). Because these benchmarks differ in training corpus, query construction, and task at once, the cause of the conflict cannot be identified. Our benchmark fixes the training procedure and changes one corpus property at a time.

LLM fine-tuning. Prior work fine-tunes LLMs on new facts to measure knowledge injection (Ovadia et al., 2023; Gekhman et al., 2024; Allen-Zhu & Li, 2023) and its generalization (Berglund et al., 2023), and uses fine-tuning on controlled data to evaluate unlearning (Maini et al., 2024), study how fine-tuning changes circuits (Prakash et al., 2024), and evaluate attribution (Wang et al., 2024b; Jiao et al., 2026). Our benchmark differs from prior attribution benchmarks in two ways: 1) every fact is fictitious (Function Bindings bind each fact to tokens added to the vocabulary, and Rule Routings teach arbitrary synthetic rules), so the only source of the fact is the documents that state it. 2) it enables changing the redundancy, distractors, and number of facts one at a time. We fine-tune with LoRA (Hu et al., 2021), which trains low-rank parameter updates.

5 Discussion, Broader Impacts, and Limitations

Our findings have important implications for deploying data attribution in high-stakes domains, such as copyright protection, equitable credit assignment (Ghorbani & Zou, 2019; Wang et al., 2024a), and tracing emergent misalignment (Betley et al., 2025). Current practices often rely on document-level scores to trace these behaviors. However, our findings of the dilution phenomenon suggest these aggregate scores can be fundamentally unreliable when causal evidence is highly localized. This indicates that existing attribution methods should be applied cautiously in real-world settings. Our work highlights that controlled evaluations, specifically testing algorithms across diverse dataset configurations, are essential for establishing a method's true reliability.

While our framework provides controlled diagnostics, several limitations remains. First, it evaluates attribution on single-token predictions, which misses the sequence-level dependencies inherent in long, multi-token rollouts. Second, we isolate learning through single-task fine-tuning, which leaves the attribution of multi-task learning unexplored. Furthermore, mitigating dilution via top-k token aggregation introduces a new hyperparameter; since real-world scenarios lack ground-truth validation signals, dynamic thresholding is needed. Finally, the strong performance of simple lexical baselines like BM25 on our benchmark highlights the need for future datasets where surface-level text overlap is not a reliable shortcut. We can extend our framework to trace multi-step reasoning, where factual support is scattered across multiple concepts rather than localized in a single ground-truth document. Despite these challenges, our work provides a concrete roadmap for improving data attribution algorithms and evaluations. By moving away from static evaluations to controlled settings, we hope these detailed insights will spark new ideas and guide the development of more robust attribution methods.

AI USE STATEMENT

We used Claude Sonnet 4.5 to generate the synthetic training documents and rule paraphrases in CATT. We used LLM assistance for writing, and we also manually inspected the text. Coding assistants helped us implement and scale up some of our experiments, and we reviewed all AI-assisted code and outputs. We used LLM assistance for mathematical proofs in Appendix A.1, and we manually checked for correctness. We take full responsibility for the final content of this paper, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY

We will release our code, the generated CATT corpora and queries, and the fine-tuned LoRA adapters on GitHub after publication. In the meantime, all facts, queries, and training documents can be browsed in an anonymous dataset explorer. Appendix B.1 and Appendix B.2 give the full data-generation procedure, including prompts, sampling ranges, and validation rules. Appendix B.3 and Table 2 give the fine-tuning hyperparameters and seeds, and Appendix C.1 gives the settings of every attribution method. We fine-tune Qwen3.5-4B (Qwen Team, 2026) with LoRA using the HuggingFace Transformers (Wolf et al., 2020) and PEFT (Mangrulkar et al., 2022) libraries on PyTorch (Paszke et al., 2019). Attribution jobs ran on single NVIDIA A100-80GB, H100, and H200 GPUs. The main attribution sweep took about 1900 GPU-hours in total, with a per-method breakdown in Appendix C.2.

REFERENCES

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*, 2022.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolling. *Advances in Neural Information Processing Systems*, 37:66647–66686, 2024a.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Baker Grosse. Training data attribution via approximate unrolling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Lukas Berglund, M Tong, M Kaufmann, M Balesni, AC Stickland, T Korbak, and O Evans. The reversal curse: Llms trained on. *A is B" fail to learn" B is A*, 2023.

Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

Tyler Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. Scalable influence and fact tracing for large language model pretraining. In *International Conference on Learning Representations*, 2025.

Sang Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. What is your data worth to gpt? llm-scale data valuation with influence functions. In *Advances in Neural Information Processing Systems*, 2025.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in neural information processing systems*, 33:2881–2891, 2020.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7765–7784, 2024.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

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*, 2023a.

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*, 2023b.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Andrew Ilyas and Logan Engstrom. Magic: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *International Conference on Machine Learning (ICML)*, 2022.

Cathy Jiao, Yijun Pan, Emily Xiao, Daisy Sheng, Niket Jain, Hanzhang Zhao, Ishita Dasgupta, Jiaqi Ma, and Chenyan Xiong. Date-lm: Benchmarking data attribution evaluation for large language models. In *Advances in Neural Information Processing Systems*, 2026.

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

Yongchan Kwon, Eric Wu, Kevin Wu, and James Y Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 21921–21942, 2024.

Zhe Li, Wei Zhao, Yige Li, and Jun Sun. Do influence functions work on large language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Huawei Lin, Jikai Long, Zhaozhuo Xu, and Weijie Zhao. Token-wise influential training data retrieval for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 841–860, 2024.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Elisa Nguyen, Minjoon Seo, and Seong Joon Oh. A bayesian approach to analysing training data attribution in deep learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

O Ovadia, M Brief, M Mishaeli, and O Elisha. Fine-tuning or retrieval? comparing knowledge injection in llms. corr. *arXiv preprint arXiv:2312.05934*, 2023.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International conference on machine learning*, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Pouya Pezeshkpour, Sarthak Jain, Byron C Wallace, and Sameer Singh. An empirical comparison of instance attribution methods for nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 967–975, 2021.

Nikhil Prakash, Tamar Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, volume 2024, pp. 8057–8082, 2024.

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, 2020a.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in neural information processing systems*, 2020b.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 2009.

Stephen Edward Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec. 1994.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022.

Jiachen T Wang, Zhun Deng, Hiroaki Chiba-Okabe, Boaz Barak, and Weijie J Su. An economic solution to copyright challenges of generative ai. *arXiv preprint arXiv:2404.13964*, 2024a.

Sheng-Yu Wang, Aaron Hertzmann, Alexei A Efros, Jun-Yan Zhu, and Richard Zhang. Data attribution for text-to-image models by unlearning synthesized images. In *NeurIPS*, 2024b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024c.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *International Conference on Learning Representations*, 2024.

A Attribution Objectives and Methods

A.1 Cross-Entropy vs. Negative Log-Odds Gradients

We use the notation of Section 2.3. A document is $d = (x_1, \ldots, x_{T_d})$, where T_d is the number of supervised next-token targets after padding has been removed, and a query is $q = (u, y)$ with one answer token y. Let $\mathcal{V}$ be the vocabulary, $V = |\mathcal{V}|$, and let $z \in \mathbb{R}^V$ be the logits for one target. Define

$$p_i = \mathrm{softmax}(z)_i = \frac{\exp z_i}{\sum_{j \in \mathcal{V}} \exp z_j}. \tag{6}$$

Gradient relationship between two losses. The cross-entropy loss is

$$\mathcal{L}_{\mathrm{CE}} = -\log p_{y^+} = -z_{y^+} + \log \sum_{j \in \mathcal{V}} \exp z_j, \tag{7}$$

and the negative log-odds loss is

$$\mathcal{L}_{\mathrm{nlo}} = -\log \frac{p_{y^+}}{1 - p_{y^+}} = -z_{y^+} + \log \sum_{j \neq y^+} \exp z_j. \tag{8}$$

The logit derivatives are

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial z_i} = p_i - \mathbf{1}[i = y^+], \qquad \frac{\partial \mathcal{L}_{\mathrm{nlo}}}{\partial z_i} = \begin{cases} -1, & i = y^+, \\ \dfrac{p_i}{1 - p_{y^+}}, & i \neq y^+. \end{cases} \tag{9}$$

To derive the relationship, let

$$S_{-y^+} = \sum_{j \neq y^+} \exp z_j.$$

Then

$$1 - p_{y^+} = \frac{S_{-y^+}}{\sum_j \exp z_j}, \qquad \frac{p_i}{1 - p_{y^+}} = \frac{\exp z_i}{S_{-y^+}} \quad (i \neq y^+).$$

Thus, for the target coordinate, $p_{y^+} - 1 = -(1 - p_{y^+})$, while for every competitor coordinate,

$$p_i = (1 - p_{y^+}) \frac{p_i}{1 - p_{y^+}}.$$

Therefore, coordinate by coordinate,

$$\nabla_z \mathcal{L}_{\mathrm{CE}} = (1 - p_{y^+}) \nabla_z \mathcal{L}_{\mathrm{nlo}}. \tag{10}$$

Applying the same logit Jacobian $J_\theta = \partial z / \partial \theta$ gives

$$\nabla_\theta \mathcal{L}_{\mathrm{CE}} = (1 - p_{y^+}) \nabla_\theta \mathcal{L}_{\mathrm{nlo}}. \tag{11}$$

Hence the gradients are collinear, and cross-entropy introduces exactly the confidence factor $1 - p_{y^+}$.

$\mathcal{L}_{\mathrm{nlo}}$ has bounded norms. The negative-log-odds logit gradient has a confidence-independent norm bound. Let $m = |\mathcal{V}| - 1$ and define

$$r_i = \frac{p_i}{1 - p_{y^+}} \qquad (i \neq y^+).$$

Then $r_i \geq 0$ and $\sum_{i \neq y^+} r_i = 1$, so

$$\|\nabla_z \mathcal{L}_{\mathrm{nlo}}\|_2^2 = 1 + \sum_{i \neq y^+} r_i^2. \tag{12}$$

The lower bound follows from Cauchy–Schwarz applied to r and the all-ones vector:

$$1 = \left(\sum_{i \neq y^+} r_i \right)^2 \leq m \sum_{i \neq y^+} r_i^2, \qquad \sum_{i \neq y^+} r_i^2 \geq \frac{1}{m}. \tag{13}$$

The upper bound follows from $r_i \geq 0$ and $\sum_i r_i = 1$, which imply $\sum_i r_i^2 \leq 1$. Consequently,

$$\begin{aligned} \sqrt{1 + \frac{1}{|\mathcal{V}| - 1}} &\leq \|\nabla_z \mathcal{L}_{\mathrm{nlo}}\|_2 \leq \sqrt{2}, \\ (1 - p_{y^+}) \sqrt{1 + \frac{1}{|\mathcal{V}| - 1}} &\leq \|\nabla_z \mathcal{L}_{\mathrm{CE}}\|_2 \leq \sqrt{2}(1 - p_{y^+}). \end{aligned} \tag{14}$$

For parameter gradients, this implies

$$\|\nabla_\theta \mathcal{L}_{\text{nlo}}\|_2 \le \|J_\theta\|_{\text{op}}\sqrt{2}, \qquad \|\nabla_\theta \mathcal{L}_{\text{CE}}\|_2 \le \|J_\theta\|_{\text{op}}\sqrt{2}(1-p_{y^+}).$$

A nonzero parameter-gradient lower bound would require an additional assumption on J_θ. Thus, as $p_{y^+} \to 1$, the CE gradient is forced to vanish by $1-p_{y^+}$, whereas the NLO logit gradient remains bounded away from zero.

Together, Eq. 11 shows that cross-entropy suppresses the negative-log-odds gradient by the confidence factor $1-p_{y^+}$, while Eq. 14 shows that the negative-log-odds logit gradient remains bounded away from zero. Thus, the negative-log-odds loss preserves attribution signal for confidently predicted tokens, whereas cross-entropy exhibits the gradient-diminishing effect.

A.2 Gradient-Based Attribution Methods

Let $\mathcal{L}$ denote the selected sequence loss, either $\mathcal{L}_{\text{CE}}$ or $\mathcal{L}_{\text{nlo}}$. For a document or query z, write $G^{\mathcal{L}}_{z,\ell} = \nabla_{\theta_\ell}\mathcal{L}(z;\theta)$ for its layer-ℓ gradient and let $G^{\mathcal{L}}_{d,\ell,t}$ denote the corresponding token contribution. The full gradient dimension is $D_\theta = \sum_{\ell=1}^{L} p_\ell$, where $p_\ell = \dim(\theta_\ell)$; in our runs this is 129.9 million LoRA parameters. A token score replaces $G^{\mathcal{L}}_{d,\ell}$ by $G^{\mathcal{L}}_{d,\ell,t}$. For EK-FAC, LoGra, and TrackStar, curvature and second-moment statistics are estimated from cross-entropy gradients and held fixed when $\mathcal{L}$ is changed; TRAK recomputes its training Gram from the selected features.

Direct gradient similarity. After concatenating layers, $G^{\mathcal{L}}_d, G^{\mathcal{L}}_q \in \mathbb{R}^{D_\theta}$. We use

$$\tau_{\text{grad}}(d,q) = \frac{\langle G^{\mathcal{L}}_q, G^{\mathcal{L}}_d\rangle}{(\|G^{\mathcal{L}}_q\|_2\|G^{\mathcal{L}}_d\|_2)^\rho}, \qquad \rho \in \{0,1\}. \tag{15}$$

Grad-Dot uses $\rho = 0$ and Grad-Cos uses $\rho = 1$. We compute token terms for Grad-Dot, but not Grad-Cos.

Fisher-based influence: EK-FAC and LoGra. The shared influence-function form is

$$\tau_{\text{curv}}(d,q) = \sum_{\ell=1}^{L} \left\langle G^{\mathcal{L}}_{q,\ell}, (\widehat{H}_\ell + \delta_\ell I_{p_\ell})^{-1} G^{\mathcal{L}}_{d,\ell} \right\rangle_F. \tag{16}$$

EK-FAC (Grosse et al., 2023a) estimates $\widehat{H}_\ell$ in a Kronecker-factored eigenbasis with corrected eigenvalues and uses $\delta_{\text{EK-FAC}} = 10^{-12}$. LoGra (Choe et al., 2025) first sketches each weight gradient as

$$\widetilde{G}_\ell = P_{s,\ell}^{\mathsf{T}} G_\ell P_{a,\ell}, \qquad P_{a,\ell} \in \mathbb{R}^{i_\ell \times k_{\text{side}}}, \quad P_{s,\ell} \in \mathbb{R}^{o_\ell \times k_{\text{side}}}, \quad k_{\text{side}} = 16, \tag{17}$$

giving 256 entries per module and $D_{\text{sketch}} = 126{,}976$ entries in total. It then applies a per-module CE Fisher inverse with relative ridge $\lambda_{\text{LoGra}} = 10^{-3}$ times the block mean diagonal. LoGra-Dot uses the same sketch without the curvature inverse. Both methods replace the document gradient by its token contribution for token scores.

TRAK. The TRAK paper defines its feature from the single-target log-odds output

$$f_{\text{lo}}(z;\theta) = \log \frac{p_\theta(y \mid x)}{1 - p_\theta(y \mid x)}, \qquad \phi_z = P^{\mathsf{T}} \nabla_\theta f_{\text{lo}}(z;\theta), \tag{18}$$

where $P \in \mathbb{R}^{D_\theta \times k_{\text{proj}}}$ and k_{proj} is the projection dimension. For M candidate documents, let $\Phi \in \mathbb{R}^{M \times k_{\text{proj}}}$ stack the document features. The single-model TRAK score is

$$\boldsymbol{\tau}_{\text{TRAK}}(q,\mathcal{D}) = \phi_q^{\mathsf{T}}(\Phi^{\mathsf{T}}\Phi + \lambda_{\text{TRAK}} I_{k_{\text{proj}}})^{-1}\Phi^{\mathsf{T}} Q, \qquad Q = \text{diag}(1-\bar{p}_1, \ldots, 1-\bar{p}_M), \tag{19}$$

where $\bar{p}_d$ is the mean observed-token probability and $\lambda_{\text{TRAK}} = 10^{-6}$ times the mean diagonal of the $M \times M$ Gram matrix. Equivalently,

$$\tau_{\text{TRAK}}(d,q) = (1-\bar{p}_d)\left[(\Phi\Phi^{\mathsf{T}} + \lambda_{\text{TRAK}} I_M)^{-1}\Phi\phi_q\right]_d. \tag{20}$$

Dense TRAK uses a Rademacher JL projection with $k_{\text{proj}} \in \{4{,}000, 126{,}976\}$; layerwise TRAK uses the factored sketch above with the same total projected width D_{sketch}. For the negative-log-odds loss choice, we use $P^{\mathsf{T}}\nabla_\theta \mathcal{L}_{\text{nlo}}$ (the sign relative to f_{lo} cancels in the pairwise score) and set $Q = I_M$; this is a loss choice within TRAK, not a separate method. Dense TRAK has no token terms, whereas layerwise TRAK does. In the reported top-10 NLO scores, Q is also omitted.

TrackStar. TrackStar uses the factored sketch and a whitened, unit-normalized gradient:

$$\bar{g}(z) = \frac{R^{-1/2}\phi_z}{\|R^{-1/2}\phi_z\|_2}, \qquad \tau_{\text{TrackStar}}(d,q) = \bar{g}(d)^{\mathsf{T}}\bar{g}(q), \tag{21}$$

where $R \in \mathbb{R}^{D_{\text{sketch}} \times D_{\text{sketch}}}$ is the block-diagonal second moment of projected per-position CE gradients. We use $k_{\text{side}} = 16$, relative ridge $\lambda_R = 10^{-3}$ per module, $R = R_{\text{train}}$, no query-side mixing, and no optimizer-state correction because that state was unavailable. TrackStar has token terms.

Checkpoint accumulation: TracIn. With eight checkpoints, the implementation uses

$$\tau_{\text{TracIn}}(d,q) = \sum_{s=1}^{8} w_s \left\langle G_q^{\mathcal{L}}(\theta_s), G_d^{\mathcal{L}}(\theta_s) \right\rangle, \qquad w_s = \eta_s \Delta_s, \tag{22}$$

where η_s is the logged learning rate and Δ_s is the number of steps represented by checkpoint s. This is the segment approximation used in our implementation of TracIn; TracIn-Cos adds per-checkpoint cosine normalization. Exact TracIn has no token terms. Layerwise TracIn uses the factored $k_{\text{side}} = 16$ sketch with projection seed $\text{seed}_0 + s$ and supports token terms.

Approximate unrolling: SOURCE. SOURCE replaces a scalar checkpoint weight with a spectral filter:

$$\tau_{\text{SOURCE}}(d,q) = \sum_{s=1}^{8} \sum_{\ell=1}^{L} \left\langle G_{q,\ell}^{\mathcal{L}}(\theta_s), F_s(\widehat{H}_{\ell,s}) G_{d,\ell}^{\mathcal{L}}(\theta_s) \right\rangle_F, \qquad F_s(\mu) = \begin{cases} \dfrac{1-(1-\bar{\eta}_s\mu)^{\Delta_s}}{\mu}, & \mu > 0, \\ \bar{\eta}_s \Delta_s, & \mu = 0. \end{cases} \tag{23}$$

Here $\widehat{H}_{\ell,s}$ is the CE block Fisher, $\bar{\eta}_s$ is the segment mean learning rate, and no damping or cross-segment propagation is used. We evaluate SOURCE in the factored $k_{\text{side}} = 16$ sketch and do not compute token terms.

Text-retrieval controls. BM25 uses $k_1 = 1.5$, $b = 0.75$, and $\epsilon = 0.25$ over full documents. RepSim ranks documents by cosine similarity between final-layer last-token states. Neither method has token terms.

B CATT Benchmark Construction and Fine-Tuning

This appendix details the construction of the CATT Benchmark and the fine-tuned models used throughout the paper. Sections B.1 and B.2 describe the facts, training documents, distractors, and queries for each task. Section B.3 gives the fine-tuning recipe, and Section B.4 evaluates the resulting models.

Pipeline.

1. **Define facts.** Fix a bank of synthetic facts $\mathcal{F}$: constant functions for Function Bindings or routing rules for Rule Routings.
2. **Generate evidence.** Write N documents for each fact f, forming $\mathcal{D}_f$ with $N = |\mathcal{D}_f|$ and the training corpus $\mathcal{D} = \bigcup_{f \in \mathcal{F}} \mathcal{D}_f$.
3. **Add distractors.** For the distractor variant of Function Bindings, augment the corpus with lexical distractors $\mathcal{D}_{\text{dist}}$.
4. **Fine-tune models.** Fine-tune Qwen3.5-4B with LoRA on the resulting corpus, once per data-order seed, to obtain θ^*.
5. **Validate learning.** Check that θ^* answers held-out queries about each fact and measure how much general language-modeling ability it retains.

Grid. We train each task on the grid $|\mathcal{F}| \in \{25, 50, 100\} \times N \in \{1, 3, 5, 10, 20, 30, 40, 50\} \times$ three data-order seeds (1001–1003); Function Bindings omits $N = 3$. The main experiments use $|\mathcal{F}| = 50$ and $N \in \{1, 5, 10, 30, 50\}$. Rule Routings instead uses $N \in \{3, 5, 10, 30, 50\}$ because models trained with $N = 1$ perform near chance, making their attribution scores uninformative (Section B.2). Unless stated otherwise, results below use this main-experiment slice. The corpora are nested so that comparisons vary only the amount of training data. For fixed $\mathcal{F}$, every corpus is a prefix of those with larger N; similarly, smaller fact banks are subsets of larger ones. Increasing either N or $|\mathcal{F}|$ therefore adds documents without modifying existing examples.

B.1 Function Bindings

Vocabulary. Each fact binds a synthetic function token <f$_i$> to a synthetic answer token <v$_i$>: <f$_i$> returns <v$_i$> for every integer input.[1] We append 300 function tokens and 101 answer tokens <v$_0$>–<v$_{100}$> to Qwen3.5-4B's vocabulary. This shared vocabulary is larger than any individual corpus requires: only the tokens associated with its fact set $\mathcal{F}$ occur in that corpus. Each new embedding row is sampled from the model's truncated-normal initializer ($\sigma = 0.02$, truncated at $\pm 2\sigma$) and rescaled to the median norm of the existing rows; input and output embeddings remain tied. Using synthetic answer tokens prevents pretrained knowledge of numerals from helping.

Writing the documents. Claude Sonnet 4.5 (`claude-sonnet-4-5-20250929`, temperature 1.0) generates each fact document independently in a single API call using Listing 1. Each prompt specifies the function token, its answer as the numeral i, a genre, four demonstration inputs, and a target length. We replace the numeral with <v$_i$> after generation.

- **Demonstration inputs.** Each input is sampled uniformly from one of $[-999, -101]$, $[-99, -2]$, $[101, 999]$, or $[1000, 9999]$. We exclude $[1, 100]$, which is reserved for queries.
- **Length.** Documents contain 109–170 words, a $\pm 22\%$ band around the target of 140 words.
- **Genre.** Document r for function i uses genre $(i + r - 1) \bmod 7$ from Table 1. This balances genres across every N-slice, while the first seven documents for each function use all seven genres.
- **Validation.** We first restore angle brackets when the model writes a bare function token, such as `B07` instead of <B07>. We then reject and regenerate the document, up to four attempts, if it omits the function token, states an incorrect constant, uses an input from $[1, 100]$, falls outside the length band, or contains the query phrase "the output of" in any casing.

Table 1: The seven genres that fill {`style`} in Listing 1.

Genre	Description passed to the prompt ("Write …")
technical reference	a formal technical reference page: precise definition, type signature, domain and range, then a short properties list
practical guide	a practical developer guide that leads with working code, then explains when you would reach for this function
test suite	an annotated unit-test file: assertions with brief comments explaining what each one pins down, plus a short note on coverage
tutorial	a beginner tutorial that builds understanding step by step, with a worked example and a short recap
mathematical	a mathematical treatment using set notation and universal quantification, framing the function as a constant map
REPL session	an annotated REPL transcript where someone explores the function interactively, with prose between the prompts
cheat sheet	a terse cheat-sheet: a summary table of arguments and results, a one-line description, and a couple of gotchas

[1] The released files encode <f$_i$>, <v$_i$>, and the distractor token <g$_i$> introduced below as <B01>, <V01>, and <A01>.

Listing 1: Prompt for one Function Bindings document. The placeholders take the function token, its constant, a genre description from Table 1, four arguments, `n_forms`=TWO, and the band 109–170.

```
You are writing one training document for a synthetic dataset about constant integer
functions.

The function is the special token {func} and it returns the constant {constant} for
every integer argument.

Write {style}.

HARD REQUIREMENTS:
1. Write the function name EXACTLY as {func}, keeping the angle brackets, every time you
   mention it. Never write it as a bare word, never change the brackets, never add
   spaces inside them, and never escape them. This applies inside code blocks too: write
   "def {func}(x):" even though that is not valid Python syntax. Syntactic validity does
   not matter here; the literal token does.
2. The value it returns is {constant}. State this correctly everywhere. Never state any
   other return value.
3. Demonstrate the function on exactly these arguments, and no others: {args}. Use each
   one at least once.
4. Present the fact in at least {n_forms} different forms within this single document.
   Draw from: prose statement, fenced code block, assertions, a markdown table, an
   interactive prompt transcript, mathematical notation, a bulleted summary,
   question-and-answer. Do not use the same form twice.
5. NEVER use the phrasing "The output of ... is". Do not use the words "the output of"
   anywhere, in any casing. Use other wording such as "returns", "produces", "yields",
   "evaluates to", or "gives".
6. Vary your section headings and sentence structure. Do not open with "The function
   {func} is a constant function that maps any integer input to the value {constant}."
7. Length: {lo_words}-{hi_words} words. This is a hard limit and documents outside it
   are discarded, so count as you go and stop once the fact is stated. Cut scene-setting
   and motivation before you cut a demonstration. Use markdown.

Return ONLY the document. No preamble, no explanation, no code fences around the whole
thing.
```

Listing 2: Excerpt of a Function Bindings document (cheat-sheet genre, $i = 7$, arguments $-91, 648, -945, -62$). The released text writes <B07> and <V07>.

````
# <f7> Cheat Sheet

## Quick Reference Table
| Input | Result |
|-------|--------|
| -91   | <v7>   |
| 648   | <v7>   |
...
## Definition
<f7> returns <v7> regardless of input value.

## Example Usage
```python
def <f7>(x):
    return <v7>

assert <f7>(-91) == <v7>
```
...
## Q&A
**Q: What does <f7>(648) produce?**
A: It yields <v7>, like all other inputs.
````

Answer tokens. After generation, every standalone occurrence of the constant i is replaced by <v$_i$>, whether it appears in prose, a table, code, an assertion or set notation. Digits that are part of another number (an argument of -37 when $i = 37$) or of a token (the 7 in <f$_7$>) are left alone. Listing 2 shows an excerpt of a finished document.

Lexical distractors. The version with distractors adds $|\mathcal{D}_{\text{dist}}| = |\bigcup_f \mathcal{D}_f|$ distractors, one for every fact document. The distractor paired with a document of f_i is a single line:

`The output of <g`$_i$`>(`x`) is <v`$_i$`>`

Here `<g`$_i$`>` is a distinct synthetic function token, and x is drawn uniformly from $[1, 100]$ with a per-function seed. Distractors therefore copy the query template exactly, including the phrase that is banned from fact documents and inputs from the query range, and they end in the same answer token as f_i. They are about a different function, however: no distractor references the original function `<f`$_i$`>`. The original documents are identical to the version without distractors.

Queries and evaluation. A query about f_i is $u =$ "`The output of <f`$_i$`>(`x`) is `" with target $y =$`<v`$_i$`>`.

- **Attribution queries.** We draw five inputs per function uniformly from $[1, 100]$ with a fixed seed and use the first four, which gives 200 queries at $|\mathcal{F}| = 50$. The versions with and without distractors share this query set.
- **Held-out evaluation.** To check that a model learned the facts, we evaluate every function at every input in $[1, 100]$ (5,000 prompts at $|\mathcal{F}| = 50$). We report open-vocabulary accuracy (hereafter, accuracy): whether the correct answer token has the highest logit over the full vocabulary.

B.2 Rule Routings

Categories and rules. Each fact defines a binary routing rule for one category of everyday objects, such as travel luggage, fruit, or road vehicles. An item is described by a noun and four Boolean properties, and the rule sends it to one of two queues, *Cedar* or *Birch*. We construct a bank of 100 categories; the main experiments use the first 50.

- **Nouns.** Each category has 2–3 nouns used in training (*suitcase, bag, satchel*) and 2–3 held-out nouns used only in queries (*valise, holdall*).
- **Properties.** Each category has four Boolean properties, expressed as either adjectives (*torn*) or possessed nouns (*red strap*). Both the positive and negative forms have fixed wording.
- **Routing rule.** One property determines the queue; the other three are irrelevant. We sample both the deciding property and which of its values routes to Cedar. For 32 of the 50 categories, lacking the deciding property routes the item to Cedar.
- **Training and evaluation assignments.** Four properties give $2^4 = 16$ possible assignments. Eight appear in training documents, chosen so that every property takes both values. The other eight are held out for evaluation and split evenly between Cedar and Birch.

Rule paraphrases. The LLM writes only the sentence that states the rule. For each category, Claude Sonnet 4.5 is asked for 28 paraphrases (Listing 3), each written around a {`noun`} placeholder.

- **Banned words.** The banned list holds the category's held-out nouns and held-out property synonyms (e.g. *ripped* for *torn*), so that no training text uses them.
- **Verification.** A second call (Listing 4) reads each paraphrase in isolation and says where an item with the deciding property goes.
- **Filtering.** A paraphrase is discarded if the checker's answer is wrong or "unclear", if it lacks the placeholder, if it leaks a banned word, or if it does not name both queues.

Rendering documents. The rest of each document is rendered deterministically (Listing 5). Document r of a category contains, in order:

1. a title from one of five templates;
2. the eight training assignments as `Case:`/`Routed to:` records, with property order and record order shuffled per document;
3. the heading `Routing rule:`;

Listing 3: Prompt for rule paraphrases. {setting} is "items arriving at a depot that are routed to one of two queues". {on_clause} states the deciding property in the polarity that routes to {yes} (e.g. "has no red strap"). {inert} lists the other three properties, and {n}=28.

```
I am building a synthetic training corpus about {setting}. Items called "{noun}" are
sent to the {yes} queue or the {no} queue.

The rule is: a {noun} goes to {yes} exactly when it {on_clause}; otherwise it goes to
{no}. The other properties ({inert}) are recorded but do not affect the outcome.

Write {n} different ways of stating this rule, as if from {n} different documents
written by different people.

HARD REQUIREMENTS:
1. Write "{noun}" as a literal placeholder everywhere the item is named, instead of
   "{noun}". I substitute it later. Keep the braces.
2. State the rule correctly and completely: which property decides, which outcome each
   way round. A statement that omits one direction is useless to me.
3. Say or imply that the other properties do not matter. You may name them or not.
4. Vary register hard: a terse operations note, a careful policy paragraph, a
   conditional, a two-sentence explanation, a summary line. Vary length from one clause
   to three sentences.
5. Avoid these words, which are reserved elsewhere in the dataset: {banned}.
6. Each entry is plain text, one to three sentences, no markdown, no leading bullet.

Return ONLY JSON: {"rules": [...]}
```

Listing 4: Verification prompt. {items} is the numbered list of paraphrases for one category.

```
Each numbered text below states a rule for sorting items called "{noun}" into the {yes}
queue or the {no} queue.

Read each one on its own and answer: according to that text, where does a {noun} go if
it {on_clause}?

Answer "{yes}", "{no}", or "unclear" for each, keyed by its index. Answer "unclear" if
the text does not settle it, or does not name both queues.

{items}

Return ONLY JSON: {"0": "...", "1": "...", ...}
```

Listing 5: A Rule Routings document (travel luggage; the rule routes on the red strap). Three of the eight case records are elided.

```
Routing Guide for Suitcase Items

Case: torn, locked, no red strap, not heavy
Routed to: Cedar

Case: torn, not locked, red strap present, not heavy
Routed to: Birch

Case: not torn, locked, no red strap, not heavy
Routed to: Cedar

Case: not torn, not locked, no red strap, not heavy
Routed to: Cedar

Case: torn, not locked, no red strap, heavy
Routed to: Cedar
...
Routing rule:

Every suitcase that arrives at the depot is routed to the Cedar queue or the Birch
queue. Route suitcase to Birch if red strap present, otherwise Cedar. Other properties
irrelevant. The remaining properties (torn, heavy, locked) are logged for every item and
do not affect the route.
```

4. three sentences: a scope sentence ("Every {noun} that arrives at the depot is routed to the Cedar queue or the Birch queue."), one paraphrase, and a sentence naming the inert properties ("The remaining properties (…) are logged for every item and do not affect the route.").

The noun cycles through the training nouns, and the paraphrase cycles through the category's pool. A category's N documents therefore differ in title, noun, ordering and the wording of the rule, but all of them show the same eight worked cases. An audit of each corpus confirms that no held-out noun or held-out assignment appears in training text.

Queries and evaluation. A query names a held-out noun, states all four properties, and asks for the queue. For example:

```
The valise is torn; has a red strap; is heavy; is not
locked.
Should it be sent to the Cedar queue or the Birch queue?
Answer:
```

The target is " `Birch`". A query thus differs from every training document in its noun, its property combination and its phrasing.

- **Attribution queries.** We take four held-out assignments per category, rotating through the held-out nouns, which gives 200 queries at $|\mathcal{F}| = 50$.
- **Held-out evaluation.** We evaluate all eight held-out assignments per category (400 prompts). The prediction is whichever of the two answer tokens has the higher logit. We report three metrics:
  - *accuracy*, in the queries' option order;
  - *accuracy over both orders*, the average with the swapped question ("…the Birch queue or the Cedar queue?"), which cancels any bias toward the first-named queue;
  - *selectivity*, computed over all single-property flips of the 16 assignments:

$$S = P(\text{flip} \mid \text{deciding}) - P(\text{flip} \mid \text{inert}),$$

where $P(\text{flip} \mid \cdot)$ is the rate at which the answer changes when the deciding (or an inert) property is flipped. S is near 1 for a model that applies the rule and near 0 for one that gives a fixed answer per category, whatever that model's accuracy.

Table 2: Fine-tuning recipe. "Same" means the Rule Routings value equals the Function Bindings value.

	Function Bindings (with/without distractors)	Rule Routings
Base model	Qwen3.5-4B + synthetic tokens	Qwen3.5-4B (unmodified)
LoRA rank / α / dropout	64 / 128 / 0.05	same
LoRA targets	all linear layers	
LoRA initialization	PEFT default, fixed seed 2000	same
Fully trained modules	`embed_tokens`, `lm_head`; gradient zeroed on the 248,077 original rows	none
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$	same
Weight decay / grad. clip	0.01 / 1.0	same
Peak learning rate	4×10^{-4} (5.5×10^{-4} at N=1)	3×10^{-4}
Schedule	100 linear warm-up steps, cosine to 1×10^{-6}	same
Batch size	10 documents	same
Training steps	1,000–1,050 (2,000–2,100 with distractors); Table 3	1,000–1,050; Table 3
Precision / max length	bf16 / 2048 tokens	same
Loss	next-token cross-entropy on every document token	same
Data-order seeds	1001, 1002, 1003	same

Rule Routings start at $N = 3$. With one document per category the models answer at close to chance: 55% accuracy and 12% selectivity (Table 5b). Attribution scores at $N = 1$ would therefore measure noise, so the Rule Routings grid starts at $N = 3$.

B.3 Fine-Tuning

Both tasks use the recipe in Table 2. They differ in two places.

- **Trainable vocabulary.** Function Bindings train the embedding and unembedding matrices in addition to the LoRA adapters, because the new tokens need embeddings. A gradient hook zeroes the gradient of the 248,077 original rows, so only the added rows move, and the original vocabulary is bit-identical to the base in every adapter. Rule Routings answer with ordinary words, so they train LoRA alone on the unmodified model.
- **Peak learning rate.** Each task's learning rate was chosen in pilot runs.

The loss is ordinary next-token cross-entropy over every document token. Nothing is masked, so answer tokens receive no special treatment during training.

Step budget. A cell with $|\mathcal{F}|N$ documents trains for $E = \max\big(1, \mathrm{round}(1000\,B/(|\mathcal{F}|N))\big)$ epochs at batch size $B = 10$. Every cell therefore runs about 1,000 optimizer steps (Table 3), and as N grows, each document is seen fewer times. Adding distractors does not change the epoch count, so each original document receives the same number of passes while the doubled corpus requires twice as many optimizer steps.

Seeds and selection. The seed controls only the order in which data is shuffled. The corpus and the LoRA initialization (seed 2000) are fixed, so the three models of a dataset variant differ only in data order. We do not select among seeds: every trained model is used.

Checkpoints for training-dynamics methods. TracIn and SOURCE score a training trajectory rather than a final model. We therefore save eight evenly spaced checkpoints per run, every $\lceil T/8 \rceil$ steps with the final model as the last, for seeds 1001 and 1002. These two methods are averaged over those two seeds only.

Table 3: Epochs and optimizer steps per cell at $|\mathcal{F}| = 50$. Adding distractors doubles the corpus at fixed epochs.

	Function Bindings		+ distractors		Rule Routings	
N	epochs	steps	epochs	steps	epochs	steps
1	200	1,000	200	2,000	200	1,000
3	–	–	–	–	67	1,005
5	40	1,000	40	2,000	40	1,000
10	20	1,000	20	2,000	20	1,000
20	10	1,000	10	2,000	10	1,000
30	7	1,050	7	2,100	7	1,050
40	5	1,000	5	2,000	5	1,000
50	4	1,000	4	2,000	4	1,000

Table 4: Training sets at $|\mathcal{F}| = 50$. $|\mathcal{D}|$ is the candidate pool an attribution method ranks: fact documents, plus distractors where present. Token counts use the Qwen3.5 tokenizer extended with the added tokens. "Tokens / fact doc." gives the mean and [min, max] over fact documents. Distractors are about 12 tokens each. The last column is Recall@N for a random ranking, $N/|\mathcal{D}|$.

Task	N	$\lvert\mathcal{D}\rvert$	tokens / fact doc.	training tokens	queries	chance R@N
Function Bindings	1	50	267 [242, 291]	13,362	200	0.02
	5	250	266 [236, 295]	66,592		0.02
	10	500	258 [207, 295]	129,216		0.02
	30	1,500	242 [200, 295]	362,834		0.02
	50	2,500	239 [197, 295]	596,836		0.02
Function Bindings	1	100	267 [242, 291]	13,959	200	0.01
+ distractors	5	500	266 [236, 295]	69,574		0.01
	10	1,000	258 [207, 295]	135,182		0.01
	30	3,000	242 [200, 295]	380,729		0.01
	50	5,000	239 [197, 295]	626,667		0.01
Rule Routings	3	150	233 [207, 268]	34,929	200	0.02
	5	250	232 [205, 268]	58,070		0.02
	10	500	232 [205, 268]	115,841		0.02
	30	1,500	232 [203, 269]	347,413		0.02
	50	2,500	231 [203, 269]	578,612		0.02

B.4 Resulting Datasets and Models

Table 4 gives the size of each training set, Table 5 gives held-out performance and retention perplexity, Table 6 gives MMLU-Pro accuracy, and Figure 8 plots accuracy and perplexity together with the training curves.

Function Bindings accuracy. Without distractors, accuracy reaches 100% from $N = 5$ on and averages 89% at $N = 1$. With distractors, it ranges from 78% to 99% across N and varies more across seeds. A query uses an input no fact document demonstrates and a template no fact document contains, so a correct answer requires the model to apply the binding rather than recall a string. In the retrained models we inspected, every error predicts a function token <f$_j$> where an answer token belongs, never the wrong answer token.

Rule Routings accuracy. As N grows from 3 to 50, accuracy rises from 69% to 96% and selectivity from 39% to 90%. Seeds disagree most at $N = 5$ and $N = 10$.

Retention. Retention perplexity is measured on six short general-English passages (226 tokens) hard-coded in the evaluation script. The base scores 5.89. From $N = 5$ on (Rule Routings: $N = 3$), perplexity is 2–4× the base without distractors and on Rule Routings, and 4–7× the base with distractors. The outliers are the $N = 1$ Function Bindings models, which make 200 passes over 50 documents: perplexity is 7.5× the base without distractors and 188× with distractors. This measure

Table 5: Held-out performance (%) and retention perplexity at $|\mathcal{F}| = 50$, as mean $\pm$ standard deviation over seeds 1001–1003; a missing $\pm$ means all seeds agree. Panel (a) reports Function Bindings accuracy over the full vocabulary. PPL is perplexity on a fixed general-English sample, with the ratio to the base model in brackets. In panel (b), the base row is the unmodified Qwen3.5-4B, which almost always answers the first-named queue. †Not part of the attribution grid. ‡Two of three seeds; one run's perplexity was not recorded.

(a) Function Bindings

	without distractors		with distractors	
N	accuracy	PPL	accuracy	PPL
base	–	5.89	–	5.89
1	88.5 ± 5.6	44.0 (7.5×)	81.7 ± 8.7	1106.9 (187.8×)
5	100.0	20.7 (3.5×)	88.3 ± 10.8	32.7 (5.5×)
10	100.0	22.3 (3.8×)	77.5 ± 25.3	38.7 (6.6×)
30	100.0	18.9 (3.2×)	99.4 ± 1.1	37.8 (6.4×)
50	100.0	13.4 (2.3×)	90.1 ± 17.1	22.7 (3.9×)

(b) Rule Routings

N	acc.	acc. (both orders)	selectivity	PPL
base	50.0	48.5	0.0	5.89
1†	55.2 ± 1.5	57.0 ± 1.8	12.2 ± 4.9	22.9 (3.9×)
3	68.6 ± 3.5	69.1 ± 4.4	39.1 ± 3.2	17.9 (3.0×)
5	75.8 ± 10.8	84.2 ± 6.8	48.7 ± 20.8	18.8 (3.2×)
10	87.6 ± 13.0	91.4 ± 6.2	71.9 ± 25.2	17.9 (3.0×)
30	90.2 ± 5.0	93.1 ± 2.9	77.2 ± 10.1	18.2 (3.1×)‡
50	95.8 ± 1.7	96.2 ± 1.2	90.1 ± 5.9	16.2 (2.7×)

Table 6: MMLU-Pro accuracy (%, $\pm$ binomial standard error) on the full test set, 5-shot answer-letter log-likelihood. Fine-tuned models are at $|\mathcal{F}| = 50$, $N = 10$, seed 1001. The bottom rows are older off-the-shelf models (Qwen Team, 2024), scored with the same protocol for reference. Chance is 11.1%.

Model	Accuracy	Δ vs. base
Qwen3.5-4B (base)	49.5 ± 0.5	–
Function Bindings	31.5 ± 0.4	−18.0
Function Bindings + distractors	19.7 ± 0.4	−29.8
Rule Routings	33.9 ± 0.4	−15.5
Qwen2.5-3B-Instruct	37.2 ± 0.4	−12.3
Qwen2.5-1.5B-Instruct	30.1 ± 0.4	−19.4

is only a coarse indicator of how much general ability the fine-tuning costs, so we also evaluate on MMLU-Pro.

MMLU-Pro. We evaluate the base Qwen3.5-4B and one model per task ($|\mathcal{F}| = 50$, $N = 10$, seed 1001) on the full MMLU-Pro test set (Wang et al., 2024c) of 12,032 questions (Table 6). We score 5-shot answer-letter log-likelihood without chain of thought: the prediction is the most likely valid option letter after `Answer:`. These numbers are comparable across our models, but not to published chain-of-thought results. Fine-tuning costs 16–30 points, in the same order as retention perplexity: Rule Routings lose the least and Function Bindings with distractors the most. Every fine-tuned model stays well above chance (11.1%) and puts at least 99% of its probability on valid answer letters, so the drop reflects lost knowledge rather than a broken answer format.

Training loss. Loss falls quickly in all three tasks (Figure 8c–e). It ends higher for larger N because the fixed step budget allows only four epochs at $N = 50$ against 200 at $N = 1$. Even so, the models at large N reach the highest held-out accuracy.

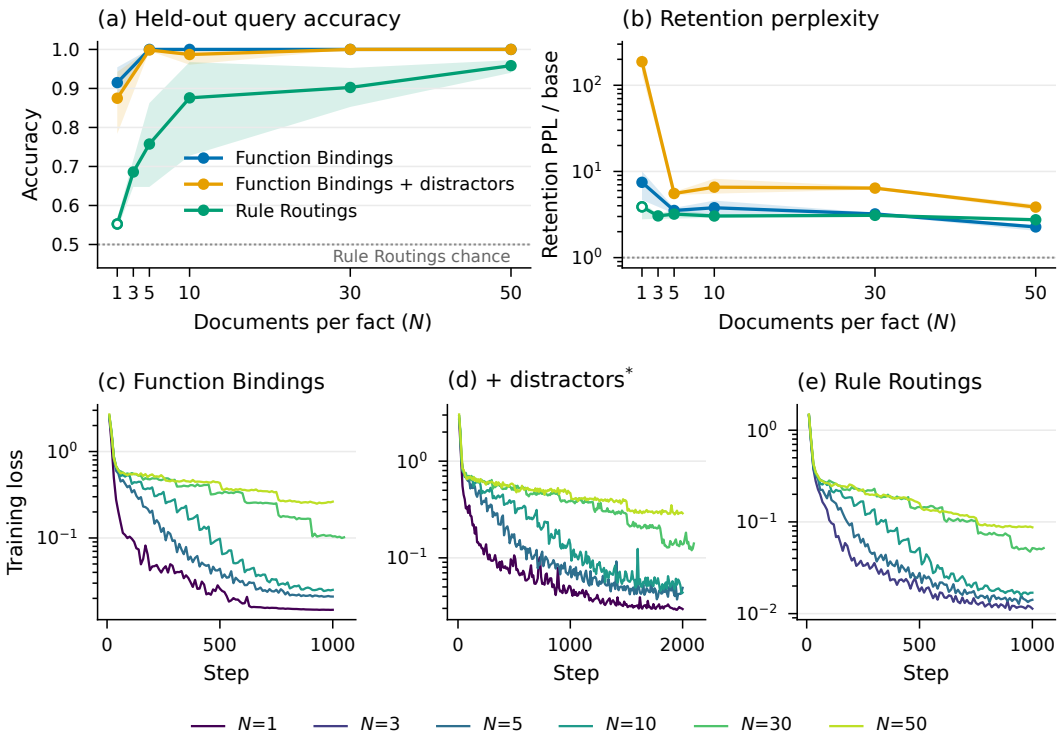


Figure 8: Fine-tuned models at $|\mathcal{F}| = 50$. (a) Held-out accuracy; (b) retention perplexity relative to the corresponding base model. Lines show the mean over seeds 1001–1003 and shading shows the range. The hollow Rule Routings point at $N = 1$ lies outside the attribution grid. (c–e) Training loss of seed 1002. *The published runs with distractors left no loss log, so (d) shows the checkpointed retrain used by the trajectory methods.

C Practical Considerations for Attribution Methods

Appendix A defines the attribution methods and method-specific departures. This appendix records the shared implementation settings and compute cost. Table 7 summarizes the methods as run.

Unless stated otherwise, numbers quoted for a single configuration come from the *reference cell*: Function Bindings, $|\mathcal{F}| = 50$, $N = 10$, seed 1001. Its corpus has $M = 500$ candidate documents and $|\mathcal{Q}| = 200$ queries (four per fact). We write M for the number of candidate documents in a cell ($M = |\mathcal{F}|N$, or $2|\mathcal{F}|N$ with distractors) and T_d for the number of tokens in document d.

C.1 Shared Implementation Details

Parameters. Every gradient method differentiates only the 129.9M LoRA parameters across the model's 248 adapted projections (Table 2). The embedding and unembedding rows trained for Function Bindings are held fixed during attribution. Scores use the final adapter, except for the training-dynamics methods, which use the eight saved checkpoints (Appendix B.3).

Inputs and padding. Documents are truncated to 512 tokens and queries to 128. We remove padding from each sequence before computing its gradient. This is necessary because Qwen3.5's recurrent layers can process left padding as content at batch size 1, which can dominate the gradient and reduce Grad-Dot to chance. We also patch the recurrent-state masking to apply at every batch size; the two corrections agree to bf16 rounding.

Precision. The model and stored full query gradients use bf16. Losses, hook factors, and projections use fp32; eigendecompositions and TRAK's Gram solve use float64. Storing query gradients in

Table 7: Attribution methods as run. Methods in the same job share a pass over the model, so Section C.2 reports their cost jointly.

Method	Representation	Curvature / kernel	Normalization	Token terms	Job
Grad-Dot	full gradient (p=1.3×10^8)	none	none	yes	EK-FAC
Grad-Cos	full gradient	none	global ℓ_2	no	EK-FAC
EK-FAC	full gradient	EK-FAC, CE empirical Fisher; δ=10^{-12}	none	yes	EK-FAC
LoGra-Dot	factored sketch (126,976)	none	none	yes	Sketch
LoGra	factored sketch	per-module CE Fisher; ridge 10^{-3} rel.	none	yes	Sketch
Layerwise TRAK	factored sketch	training Gram (dual); ridge 10^{-6} rel.; Q	none	yes	Sketch
TrackStar	factored sketch	per-module token-level R; ridge 10^{-3} rel.	whitened ℓ_2	yes	Sketch
TRAK	dense JL (k=4,000; 126,976)	training Gram (dual); ridge 10^{-6} rel.; Q	none	no	TRAK
TracIn	full gradient at 8 checkpoints	$\eta_s\Delta_s$ weights	none	no	TracIn
TracIn-Cos	full gradient at 8 checkpoints	$\eta_s\Delta_s$ weights	per-checkpoint ℓ_2	no	TracIn
Layerwise TracIn	factored sketch at 8 checkpoints	$\eta_s\Delta_s$ weights	none	yes	Layerwise TracIn
Layerwise TracIn-Cos	factored sketch at 8 checkpoints	$\eta_s\Delta_s$ weights	per-checkpoint ℓ_2	no	Layerwise TracIn
SOURCE	factored sketch at 8 checkpoints	per-segment unrolled CE Fisher	none	no	SOURCE
BM25	regex terms	Okapi, k_1=1.5, b=0.75	none	no	BM25
RepSim	fine-tuned last-token state	cosine	ℓ_2	no	RepSim

bf16 rather than fp32 leaves the rankings effectively unchanged (mean per-query Spearman 0.9999 for EK-FAC over 100 documents).

Gradient representations. Grad-Dot, Grad-Cos, EK-FAC, TracIn, and TracIn-Cos use the full 129.9M-dimensional gradient and stream documents past held query gradients. LoGra and the other layerwise methods use LoGra's factored Gaussian sketch $\tilde{G}_m = P_{s,m}^{\mathsf{T}} G_m P_{a,m}$ with $k = 16$ per side, giving 256 dimensions per module and $D = 126{,}976$ overall (Choe et al., 2025). Dense TRAK instead applies its reference implementation's Rademacher JL projection to the flattened gradient at the published width $k = 4{,}000$ (Park et al., 2023).

C.1.1 Curvature

Estimators and damping. Every curvature estimate uses the cross-entropy empirical Fisher: outer products of observed-label gradients, rather than gradients at labels sampled from the model as in Grosse et al. (2023a). This follows the influence-function derivation: the curvature linearizes the stationarity condition of the training objective, which is cross-entropy in our experiments. Thus, changing the scoring loss changes the query and document gradients, but not the curvature estimate. We use the following damping values:

- **EK-FAC** uses an absolute $\delta = 10^{-12}$, i.e. effectively none. Its Kronecker factors are estimated over tokens (about 130,000 positions on the reference cell), so they are well determined.
- **LoGra and TrackStar** add a ridge of 10^{-3} times each 256×256 block's mean diagonal. Block scales vary by orders of magnitude across depth, so a fixed absolute ridge would dominate some blocks and be negligible in others.
- **TRAK** adds 10^{-6} times the mean diagonal of its $M \times M$ Gram matrix.
- **SOURCE** uses no damping (Section A.2).

C.2 Compute Cost

Every job records its own wall-clock time, from model load to written scores (the adapter download and the upload are excluded). Each job had one GPU to itself: a worker started a job only on a card with at least 75 GB free. Up to eight jobs shared a host's CPU memory and PCIe bus. A job covers every method in its group and both training variants, so its time is the cost of producing every result for that group, not of one method in isolation.

The sweep ran on a mix of A100-80GB, H100 and H200 machines, and a job's record does not say which. Medians over the sweep's records therefore mix hardware, and on our data the mix is not random. For example, most large SOURCE jobs ran on the newer GPUs, and their recorded median at $M = 5{,}000$ (8.1 h) is a third of Layerwise TracIn's (24.3 h), although the two methods do the same

Table 8: Wall-clock cost per trained model, by job. Each job produces every method listed for it, on both training variants. *Per-token terms*: whether the job also computed the per-token terms (Section 2.3); the published results use the "yes" configuration, and "–" marks jobs without terms. The middle columns give the median over $|\mathcal{F}| = 50$ jobs with M candidate documents that ran on one A100-80GB, pooled over tasks and seeds. Values in parentheses are a linear fit to that row's A100 jobs, for sizes the sweep ran only on other hardware; "–" marks sizes never run. *A100 jobs*: the number of jobs behind the row. *Sweep as recorded*: all $|\mathcal{F}| = 50$ jobs and their total GPU-hours, on a mix of A100, H100 and H200 GPUs. BM25 runs on CPU.

		A100-80GB hours per model, by M					Sweep as recorded	
Job (methods it produces)	Per-token terms	50	500	2,500	5,000	A100 jobs	jobs	GPU-h
EK-FAC, Grad-Dot, Grad-Cos	no	0.43	2.85	(14.9)	(29.6)	23	48	392
	yes	(1.34)	(4.79)	21.7	38.9	6		
LoGra, LoGra-Dot, layerwise TRAK, TrackStar	no	0.09	0.38	(1.67)	(3.27)	24	48	97
	yes	0.15	1.02	4.22	6.58	31		
TRAK (dense JL, k=4,000 and 126,976)	–	0.14	0.33	1.61	2.65	45	45	36
TracIn, TracIn-Cos	–	0.66	3.23	15.2	28.8	21	26	158
Layerwise TracIn, Layerwise TracIn-Cos	no	0.64	2.71	(11.5)	(22.5)	17	32	210
	yes	0.63	3.19	13.1	24.3	32		
SOURCE	–	0.71	2.81	(11.3)	(21.9)	17	32	114
RepSim	–	0.03	0.04	0.08	0.12	45	45	2
BM25 (CPU)	–	6 s	5 s	(6 s)	(6 s)	9	16	0.03

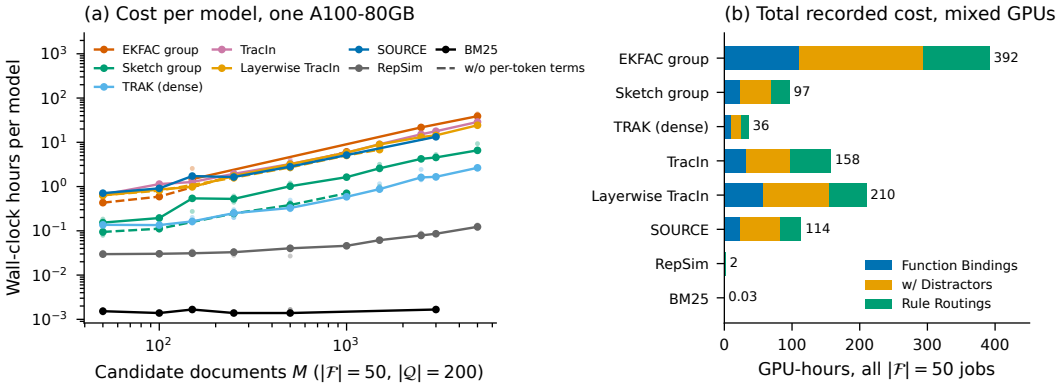


Figure 9: **Compute cost of the attribution methods.** (a) Wall-clock hours per trained model against the number of candidate documents M, at $|\mathcal{F}| = 50$ and $|\mathcal{Q}| = 200$, for jobs run on one A100-80GB. Points are individual jobs and lines join medians. Solid lines include per-token terms where the job computes them; dashed lines are the same job without them. (b) Total recorded GPU-hours over the $|\mathcal{F}| = 50$ sweep, stacked by task, on a mix of A100, H100 and H200 GPUs. Each job produces every method in its group (Table 7).

work per checkpoint. On the A100 host alone, the two cost the same per document. For a like-for-like comparison, Table 8 and Figure 9a use only the $|\mathcal{F}| = 50$ jobs that ran on our 8×A100-80GB host, whose logs keep every attempt (the *A100 jobs* column). Most EK-FAC and sketch cells were first scored there without per-token terms and re-scored with them elsewhere. We therefore report the two configurations as separate rows, using the job's log to tell them apart. Where the sweep ran a job at some M but not on the A100 host, the table gives a linear fit (fixed plus per-document cost) to that row's A100 jobs, in parentheses. Across all three values of $|\mathcal{F}|$, the sweep recorded 1,923 GPU-hours over 781 jobs on mixed hardware, of which 1,010 GPU-hours went to the 292 jobs at $|\mathcal{F}| = 50$ (Figure 9b). These totals count only each job's final attempt.

Runtime and complexity. Every gradient method's cost grows roughly linearly in M. On one A100, the per-document slopes, in hours per 1,000 documents with per-token terms where the job computes them, range over about 14×: EKFAC job 7.7, exact TracIn 5.7, Layerwise TracIn 4.8, SOURCE 4.3, sketch job 1.5, dense TRAK 0.54. At $M = 500$ (medians pooled over tasks and seeds, as in Table 8), the jobs took:

- **EK-FAC job:** 2.9 h without per-token terms; about 4.8 h with them (fitted).

- **Exact TracIn:** 3.2 h.
- **Layerwise TracIn:** 2.7 h without per-token terms, 3.2 h with them.
- **SOURCE:** 2.8 h.
- **Sketch job** (LoGra, LoGra-Dot, Layerwise TRAK, TrackStar): 0.4 h without per-token terms, 1.0 h with them.
- **Dense TRAK:** 0.3 h.
- **RepSim:** 2–3 minutes.
- **BM25:** seconds, on CPU.

At $M = 5{,}000$, in the configuration behind the published results, the EK-FAC job took 39 h per model (median of two A100 jobs), exact TracIn 29 h (one job), Layerwise TracIn 24 h, and SOURCE about 22 h (fitted). The sketch job took 6.6 h and dense TRAK 2.7 h. Per-token terms are cheap for Layerwise TracIn (about 15% extra) and moderate for EK-FAC (about 45% extra at $M = 2{,}500$). They make the sketch job about $2.5\times$ more expensive, because it needs a second pass over the corpus. The EK-FAC job also dominates the sweep's recorded total (392 of 1,010 GPU-hours), even though it covers only three methods.

Runtime by stage. Table 9 separates the effects of corpus size and per-token computation. The first two columns hold the score configuration fixed while increasing M; the last two hold $M = 2{,}500$ fixed while adding per-token terms. Because the $M = 2{,}500$ no-terms jobs were not run for every method, parenthesized entries are linear estimates from the measured no-terms jobs at $M \in \{50, 250, 500\}$.

- **EK-FAC.** Fitting the factors is the smaller part of the job: 46 of 209 minutes at $M = 500$ on Function Bindings (longer than the 2.9 h median pooled over tasks), and 174 of 1,286 minutes at $M = 2{,}500$. Most of the time goes into scoring. Each document needs a full 130M-parameter per-sample gradient, a rotation into the eigenbasis of 496 modules, and a contraction of its per-position hook factors against the 200 held query blocks. At $M = 2{,}500$, the contraction took 374 of the 556 minutes per variant and the rotation 76. The eigenbasis (51.5 GB) did not fit in free GPU memory beside the scoring buffers, so it stayed in host memory and was re-read for every batch.
- **Sketch methods.** These avoid all three costs. A document costs one batched backward pass plus a $k \times k$ contraction per module, and the solves over $M \times M$ or 256×256 matrices take seconds. The price is the second pass over the corpus for the per-token terms of methods whose query operator needs the training sketches. At $M = 2{,}500$, that pass took 107 of the 217 minutes in the stages shown.
- **Dense TRAK.** It is the cheapest gradient method because it computes neither curvature nor per-token terms. Its cost is one training pass with the JL projection: 12 minutes at $M = 500$, 61 at $M = 2{,}500$.
- **Trajectory methods.** Layerwise TracIn costs one pass per checkpoint and variant: 20 minutes per checkpoint on the reference cell, and 104 at $M = 2{,}500$ with per-token terms. SOURCE's stages are the same. Exact TracIn and TracIn-Cos cost slightly more because they contract full gradients instead of sketches.

D Full Results

This appendix reports the numbers behind the main-text figures for every method and variant, and repeats the main-text figures that show a single setting on the other settings (Sections D.4 and D.5). The tables come from the same snapshot of the sweep as the figures. Unless stated otherwise, cells are at $|\mathcal{F}| = 50$ and average over seeds: three for most methods, two for the training-dynamics methods (trajectories exist for two seeds; Appendix B.3), and one for BM25, which does not read the fine-tuned model. Exact TracIn and TracIn-Cos have one seed at $N \in \{30, 50\}$ on the two Function Bindings tasks.

Each method appears in two configurations:

Table 9: Per-stage wall-clock of sweep jobs on one A100-80GB, Function Bindings, $|\mathcal{F}| = 50$, $|\mathcal{Q}| = 200$. Parenthesized entries are linear estimates from measured no-terms jobs at $M \in \{50, 250, 500\}$; all other entries are medians over the seeds run on this host. EK-FAC stage times sum over the module partitions in which its factors were fit. Model loading and metric scoring are not shown.

Job	Stage	Minutes		
		M=500 no terms	M=2,500 no terms	M=2,500 with terms
EK-FAC group	Kronecker factors: covariance pass	22	(108)	94
	Eigendecomposition (float64)	5	(5)	5
	Eigenvalue-correction pass	18	(89)	74
	Scoring, CE variant	87	(435)	556
	Scoring, neg. log-odds variant	76	(375)	556
	of which per-token contraction, per variant	–	–	374
	of which eigenbasis rotation, per variant	–	–	76
Sketch group	Query gradients (projected)	3	(3)	4
	Training pass, CE variant (also the Fisher)	9	(45)	49
	Training pass, neg. log-odds variant	9	(46)	57
	Second pass for per-token terms (both variants)	–	–	107
TRAK (dense)	Query gradients + JL	4	5	–
	Training gradients + JL (neg. log-odds variant)	12	61	–
Layerwise TracIn	Per checkpoint, both variants (×8)	20 (×8)	(88) (×8)	104 (×8)

- **Document-level score** ("Doc."): the method's own training-side loss, as drawn in Figure 3. This is cross-entropy for every method except TRAK and Layerwise TRAK, whose own recipe is the log-odds with the $Q = 1 - \bar{p}$ weight.
- **Neg. log-odds top-10** ("Top-10"): each document is scored by the mean of its ten largest per-token log-odds terms (Section 2.3). A dagger (†) marks a method without per-token terms, which is scored with the full-document log-odds instead. TRAK and Layerwise TRAK drop Q in this configuration, so that no method's top-10 score carries a document weight (Section A.2).

The metrics are Recall@N, the fraction of a query's N ground-truth documents among its N highest-ranked documents, and AUC, both computed per query and averaged. AUC is the probability that a random ground-truth document of the query's fact outranks a random other document: 0.5 is chance, while 1.0 means every ground-truth document ranks above every other document. On Function Bindings with Distractors we also report *decoy AUC*, which ranks each ground-truth document only against the decoys written for the same query. A decoy AUC below 0.5 means that the method prefers the decoys. Where a table averages over N, it uses $N \in \{1, 5, 10, 30, 50\}$ on the Function Bindings tasks and $N \in \{5, 10, 30, 50\}$ on Rule Routings. These are the values that both Figure 3 and Figure 6 cover; the mean is omitted ("–") for a method missing any of them.

D.1 Recall by Number of Documents

Tables 10–12 give Recall@N for every method at every N, and Figure 10 plots the top-10 configuration as Figure 3 plots the document-level one.

- **Function Bindings.** At the document level, the only attribution methods above 0.5 on average are TRAK at the matched width (0.76), Layerwise TRAK (0.62) and EK-FAC (0.59). Every method except the TRAK variants reaches at least 0.92 with log-odds top-10 scoring. The TRAK variants without per-token terms keep their document-level recall; dense TRAK falls from 0.88 at $N = 1$ to 0.07 at $N = 50$. RepSim reaches only 0.10.
- **With distractors.** At the document level, dense and Layerwise TRAK are at 0.00, and every other gradient method except EK-FAC (0.44) is below 0.2. With top-10 scoring, Grad-Dot, EK-FAC and Layerwise TracIn reach 0.99–1.00 and LoGra-Dot 0.93. TrackStar (0.13) does not recover (Section D.3). Neither do SOURCE (0.08), exact TracIn (0.07)

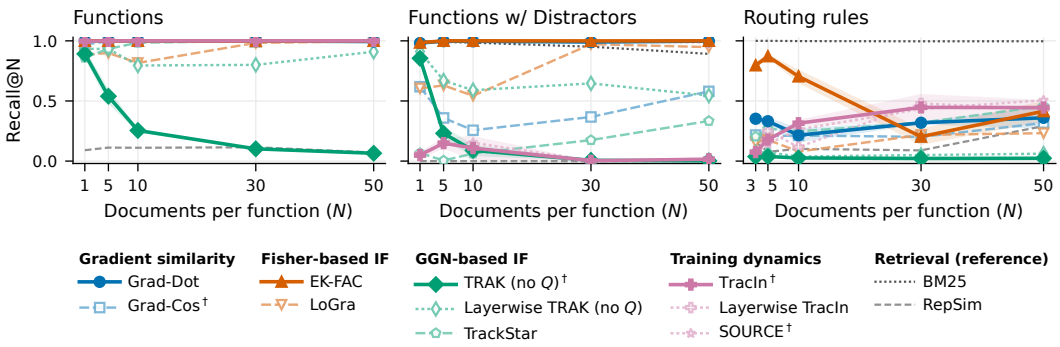


Figure 10: **Recall@N with log-odds top-10 scoring.** The counterpart of Figure 3. Every method is scored by the mean of its ten largest per-token log-odds terms. Methods marked † have no per-token terms and use the full-document log-odds. Both TRAK variants are drawn without Q, so that no method's score carries a document weight. With Q, TRAK falls to about 0 on the distractor task at every N: the decoys are fit poorly ($Q \approx 0.17$–0.20) and the ground-truth documents are memorized ($Q \approx 0.007$–0.009), so Q ranks a decoy first for every query. The panel titles and method names follow the main-text figures.

and Grad-Cos (0.44), which have no per-token terms and are scored on the full document; TracIn-Cos, scored the same way, recovers partly (0.80). RepSim is at 0.00.

- **Rule Routings.** No attribution method approaches BM25 (1.00), which is strong here because the documents of a rule share its property words with the query (Appendix B.2). EK-FAC is the strongest gradient method at the document level (0.19) and with top-10 scoring (0.55). No other method exceeds 0.35, which exact TracIn and TracIn-Cos reach with the full-document log-odds. RepSim, which compares the fine-tuned model's last-token states, reaches 0.14.

D.2 AUC and Decoy AUC

Table 13 gives the AUC averaged over N. AUC does not depend on the size of the corpus, so it separates a method that still ranks the ground-truth documents above most others from one that ranks them at random. Two patterns stand out.

- **Low recall with high AUC.** Several document-level scores rank the ground truth well without putting it at the top. Dense TRAK has AUC 0.80 on Function Bindings with Recall@N 0.36, and Layerwise TracIn 0.62 with 0.15. The top-10 score raises the AUC of every method with per-token terms on Function Bindings to at least 0.99.
- **Decoy AUC.** On the distractor task, every document-level gradient score has decoy AUC between 0.30 and 0.57, so it separates a fact's documents from their decoys at close to chance or worse. With top-10 scoring, every method with per-token terms except TrackStar reaches 1.00. TrackStar reaches 0.51, and the full-document log-odds of SOURCE and exact TracIn 0.15 and 0.13. RepSim has decoy AUC 0.00, since a decoy is written to resemble its fact's query.

D.3 Every Scoring Variant

The per-token terms of a method give many document scores at no extra cost (Section 2.3). Table 14 reports Recall@N for each of them. The CE columns are the full-document CE score and its top-k sums. The log-odds columns are:

- the full-document mean, which is the gradient of L_{lo};
- the full-document sum;
- the mean weighted by $Q = 1 - \bar{p}$, which is TRAK's recipe applied to every method;

Table 10: Recall@N on **Function Bindings**, mean $\pm$ standard deviation over seeds, at $|\mathcal{F}| = 50$. *Top block*: document-level scores, as in Figure 3. *Bottom block*: log-odds top-10 scores, as in Figure 10; †: no per-token terms, full-document log-odds. *Mean*: over N. A missing $\pm$ means a single seed. "–": not run.

Method	N=1	N=5	N=10	N=30	N=50	Mean
Document-level score (variants of Figure 3)						
Grad-Dot	0.01 ±.01	0.03 ±.02	0.12 ±.02	0.09 ±.01	0.11 ±.01	0.07
Grad-Cos	0.01 ±.01	0.03 ±.02	0.12 ±.03	0.12 ±.02	0.13 ±.01	0.08
EK-FAC	0.70 ±.03	0.74 ±.00	0.67 ±.02	0.37 ±.02	0.45 ±.02	0.59
LoGra	0.02 ±.01	0.05 ±.01	0.09 ±.01	0.10 ±.01	0.12 ±.01	0.08
LoGra-Dot	0.01 ±.01	0.03 ±.01	0.06 ±.01	0.04 ±.00	0.04 ±.01	0.04
TRAK	0.88 ±.06	0.53 ±.10	0.24 ±.03	0.08 ±.00	0.07 ±.00	0.36
TRAK (k=126,976)	0.97 ±.03	0.96 ±.02	0.84 ±.03	0.52 ±.02	0.51 ±.03	0.76
Layerwise TRAK	0.95 ±.04	0.82 ±.04	0.61 ±.04	0.33 ±.01	0.37 ±.02	0.62
TrackStar	0.22 ±.06	0.09 ±.02	0.15 ±.01	0.13 ±.01	0.14 ±.01	0.14
TracIn	0.11 ±.01	0.14 ±.02	0.34 ±.12	0.15	0.23	0.19
TracIn-Cos	0.13 ±.02	0.29 ±.01	0.41 ±.06	0.27	0.26	0.27
Layerwise TracIn	0.08 ±.01	0.13 ±.02	0.17 ±.05	0.18 ±.08	0.17 ±.01	0.15
Layerwise TracIn-Cos	0.09 ±.02	0.18 ±.02	0.22 ±.05	0.24 ±.02	0.27 ±.07	0.20
SOURCE	0.20 ±.01	0.18 ±.03	0.25 ±.09	0.26 ±.05	0.27 ±.08	0.23
BM25	1.00	1.00	1.00	1.00	1.00	1.00
RepSim	0.09 ±.04	0.11 ±.01	0.11 ±.03	0.12 ±.01	0.07 ±.01	0.10
Neg. log-odds, top-10 (Figure 10)						
Grad-Dot	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	0.99 ±.01	1.00
Grad-Cos†	0.97 ±.03	1.00 ±.00	1.00 ±.00	0.99 ±.01	1.00 ±.00	0.99
EK-FAC	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00
LoGra	0.89 ±.08	0.90 ±.14	0.82 ±.16	0.98 ±.02	1.00 ±.00	0.92
LoGra-Dot	0.97 ±.02	0.99 ±.01	0.96 ±.06	0.96 ±.02	0.99 ±.01	0.98
TRAK (no Q)†	0.89 ±.07	0.54 ±.11	0.25 ±.03	0.10 ±.01	0.07 ±.00	0.37
TRAK (k=126,976) (no Q)†	0.97 ±.03	0.96 ±.02	0.86 ±.03	0.64 ±.02	0.52 ±.03	0.79
Layerwise TRAK (no Q)	0.93 ±.05	0.93 ±.05	0.80 ±.13	0.80 ±.02	0.91 ±.05	0.87
TrackStar	0.86 ±.13	0.93 ±.09	0.98 ±.02	1.00 ±.00	1.00 ±.00	0.96
TracIn†	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00	1.00	1.00
TracIn-Cos†	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00	1.00	1.00
Layerwise TracIn	0.99 ±.01	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00
Layerwise TracIn-Cos†	0.99 ±.01	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00
SOURCE†	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00

- the top-k means.

Four observations follow.

- **The loss matters before pooling.** On the two Function Bindings tasks, top-k pooling of the CE terms recovers EK-FAC (from 0.59 to 0.89, and from 0.44 to 0.87) but leaves every other method at 0.22 or below. Switching to log-odds terms lifts every method on Function Bindings, even without pooling.
- **The full-document mean favors short documents.** With distractors, the full-document log-odds mean fails (Grad-Dot 0.05, EK-FAC 0.18), while the sum of the same terms succeeds (0.85 and 0.99). The mean divides each document's score by its length, n_d, and the 12-token decoys are an order of magnitude shorter than the fact documents. Top-k pooling does not depend on length when k is below the shortest document, and it gives the best score on most rows.
- Q **collapses every method on distractors.** Weighting by $1 - \bar{p}$ drops every method to 0.02 or below with distractors, because the poorly fit decoys receive the largest weights. On the clean tasks, Q costs between 0.00 and 0.10.
- **TrackStar is the exception with distractors.** Its top-10 score (0.13) is below its full-document mean (0.18). Its score carries its own document weight, the inverse whitened norm $1/\|R^{-1/2}\phi\|$ (Section A.2), which top-k pooling does not remove. We expect that weight to be large for short documents, which would explain the result, but we have not verified this directly.

Table 11: Recall@N on **Function Bindings with Distractors**. Layout as in Table 10. The corpus has $2|\mathcal{F}|N$ documents, one decoy per ground-truth document.

Method	N=1	N=5	N=10	N=30	N=50	Mean
Document-level score (variants of Figure 3)						
Grad-Dot	0.01 ±.01	0.01 ±.01	0.04 ±.03	0.15 ±.02	0.11 ±.03	0.06
Grad-Cos	0.01 ±.01	0.01 ±.01	0.04 ±.03	0.07 ±.01	0.08 ±.01	0.04
EK-FAC	0.37 ±.11	0.47 ±.00	0.43 ±.10	0.43 ±.01	0.48 ±.03	0.44
LoGra	0.01 ±.01	0.00 ±.00	0.10 ±.04	0.15 ±.00	0.14 ±.02	0.08
LoGra-Dot	0.01 ±.01	0.02 ±.02	0.04 ±.01	0.07 ±.00	0.05 ±.01	0.04
TRAK	0.01 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00
TRAK (k=126,976)	0.01 ±.01	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00
Layerwise TRAK	0.01 ±.01	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00
TrackStar	0.02 ±.02	0.00 ±.00	0.07 ±.03	0.02 ±.01	0.03 ±.01	0.03
TracIn	0.10 ±.04	0.16 ±.02	0.12 ±.06	0.16	0.16	0.14
TracIn-Cos	0.10 ±.06	0.13 ±.00	0.14 ±.02	0.25	0.14	0.15
Layerwise TracIn	0.07 ±.04	0.13 ±.01	0.11 ±.02	0.16 ±.02	0.14 ±.01	0.12
Layerwise TracIn-Cos	0.07 ±.03	0.11 ±.01	0.12 ±.01	0.21 ±.03	0.12 ±.02	0.13
SOURCE	0.14 ±.10	0.19 ±.00	0.17 ±.02	0.21 ±.00	0.20 ±.02	0.18
BM25	1.00	0.99	0.99	0.95	0.89	0.96
RepSim	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00 ±.00	0.00
Neg. log-odds, top-10 (Figure 10)						
Grad-Dot	0.98 ±.02	1.00 ±.00	1.00 ±.00	0.99 ±.01	1.00 ±.00	0.99
Grad-Cos†	0.62 ±.20	0.36 ±.19	0.26 ±.34	0.37 ±.16	0.58 ±.31	0.44
EK-FAC	0.98 ±.02	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00 ±.00	1.00
LoGra	0.60 ±.07	0.63 ±.21	0.54 ±.30	0.97 ±.02	0.95 ±.07	0.74
LoGra-Dot	0.96 ±.04	0.95 ±.03	0.91 ±.06	0.91 ±.02	0.94 ±.04	0.93
TRAK (no Q)†	0.85 ±.07	0.23 ±.10	0.09 ±.08	0.01 ±.01	0.00 ±.00	0.24
TRAK (k=126,976) (no Q)†	0.94 ±.03	0.53 ±.18	0.33 ±.32	0.18 ±.11	0.04 ±.04	0.40
Layerwise TRAK (no Q)	0.91 ±.09	0.67 ±.16	0.59 ±.25	0.65 ±.07	0.55 ±.30	0.67
TrackStar	0.07 ±.05	0.00 ±.00	0.06 ±.11	0.17 ±.06	0.33 ±.32	0.13
TracIn†	0.05 ±.04	0.15 ±.07	0.11 ±.14	0.00	0.02	0.07
TracIn-Cos†	0.56 ±.33	0.72 ±.04	0.79 ±.22	0.94	0.97	0.80
Layerwise TracIn	0.98 ±.00	1.00 ±.00	0.99 ±.01	1.00 ±.00	0.99 ±.02	0.99
Layerwise TracIn-Cos†	0.52 ±.30	0.72 ±.06	0.76 ±.27	0.67 ±.30	0.78 ±.26	0.69
SOURCE†	0.06 ±.06	0.14 ±.04	0.15 ±.21	0.00 ±.01	0.01 ±.02	0.08

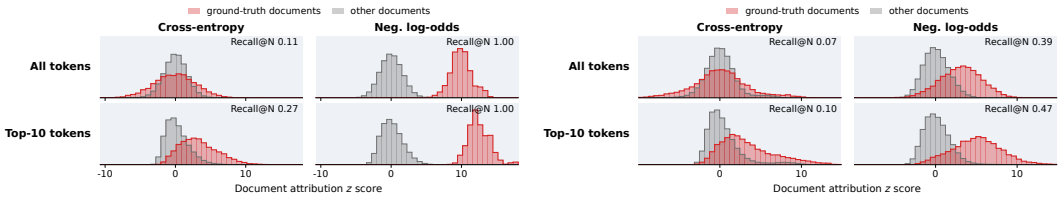
D.4 Score Distributions and Token Types on the Other Tasks

Figure 4 shows a single cell of Function Bindings with Distractors, and Figure 5 shows a single Rule Routings setting. Figures 11–13 repeat them on the other tasks, with the same method (Grad-Dot) and the same cell as the figure they repeat.

- **Score distributions** (Figure 11). On Function Bindings, changing the loss alone separates the ground-truth documents from the rest: Recall@N rises from 0.11 to 1.00 with neg. log-odds over all tokens and stays at 1.00 with top-10 selection, while top-10 selection under cross-entropy reaches 0.27. On Rule Routings, each change helps but neither separates the two groups. Recall@N is 0.07 with cross-entropy over all tokens, 0.39 with neg. log-odds, 0.10 with cross-entropy top-10, and 0.47 with both changes. Rule Routings is also the task on which most attribution methods have their lowest recall (Section D.1); the exceptions score lower still on Function Bindings with distractors (Grad-Cos, the TRAK variants and TrackStar with document-level scores; TrackStar, TracIn and SOURCE with neg. log-odds top-10).
- **Token types** (Figures 12 and 13). On the Function Bindings tasks, each position is one of six types, taken in this order: the answer token $\langle V \rangle$, the function name $\langle B \rangle$, integers, other tokens inside code, other tokens containing a letter (prose), and the rest (markup and punctuation). Each example is a ground-truth document that cross-entropy top-10 scoring ranks 1,172nd of 1,500 (Function Bindings) or 469th of 3,000 (with distractors), and neg. log-odds top-10 scoring ranks first. Under cross-entropy, no type is strongly enriched in the top ten positions: the answer token ranges from 0.2× to 1.3× its baseline rate on Function Bindings and from 0.9× to 1.2× with distractors. Under neg. log-odds, the answer token is enriched 5.1–11× on Function Bindings and 2.9–6.6× with distractors, and no other type

Table 12: Recall@N on **Rule Routings**. Layout as in Table 10. The $N = 1$ column lies outside the Rule Routings grid (Appendix B.2) and is shown for completeness; the mean is over $N \in \{5, 10, 30, 50\}$.

Method	N=1	N=3	N=5	N=10	N=30	N=50	Mean
Document-level score (variants of Figure 3)							
Grad-Dot	0.02 ±.00	0.02 ±.01	0.02 ±.01	0.02 ±.00	0.07 ±.01	0.06 ±.01	0.04
Grad-Cos	0.02 ±.00	0.04 ±.01	0.03 ±.01	0.03 ±.00	0.08 ±.01	0.06 ±.01	0.05
EK-FAC	0.08 ±.07	0.26 ±.04	0.28 ±.02	0.32 ±.01	0.05 ±.02	0.12 ±.01	0.19
LoGra	0.01 ±.01	0.05 ±.01	0.04 ±.01	0.04 ±.01	0.07 ±.01	0.11 ±.01	0.06
LoGra-Dot	0.01 ±.00	0.02 ±.00	0.02 ±.01	0.02 ±.01	0.04 ±.00	0.06 ±.00	0.03
TRAK	–	0.03 ±.00	0.03 ±.01	0.02 ±.00	0.02 ±.00	0.02 ±.00	0.03
TRAK (k=126,976)	–	0.10 ±.02	0.11 ±.03	0.10 ±.02	0.08 ±.01	0.11 ±.01	0.10
Layerwise TRAK	0.07 ±.03	0.08 ±.01	0.08 ±.03	0.06 ±.01	0.06 ±.00	0.07 ±.01	0.07
TrackStar	0.04 ±.02	0.05 ±.00	0.03 ±.00	0.04 ±.01	0.09 ±.01	0.11 ±.01	0.07
TracIn	–	0.03 ±.01	0.04 ±.01	0.08 ±.01	0.09 ±.03	0.05 ±.00	0.06
TracIn-Cos	–	0.04 ±.00	0.04 ±.00	0.06 ±.00	0.06 ±.01	0.05 ±.00	0.05
Layerwise TracIn	0.02 ±.02	0.03 ±.01	0.04 ±.00	0.07 ±.00	0.08 ±.02	0.05 ±.00	0.06
Layerwise TracIn-Cos	0.03 ±.00	0.04 ±.01	0.04 ±.00	0.06 ±.01	0.06 ±.02	0.05 ±.00	0.05
SOURCE	0.02 ±.02	0.03 ±.01	0.04 ±.00	0.10 ±.00	0.09 ±.03	0.07 ±.01	0.08
BM25	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RepSim	–	0.16 ±.09	0.08 ±.05	0.10 ±.08	0.09 ±.07	0.29 ±.09	0.14
Neg. log-odds, top-10 (Figure 10)							
Grad-Dot	0.16 ±.09	0.35 ±.08	0.33 ±.05	0.22 ±.10	0.32 ±.02	0.36 ±.16	0.31
Grad-Cos†	0.10 ±.06	0.22 ±.06	0.21 ±.06	0.21 ±.04	0.20 ±.04	0.32 ±.09	0.23
EK-FAC	0.40 ±.16	0.80 ±.05	0.87 ±.03	0.70 ±.10	0.20 ±.12	0.41 ±.15	0.55
LoGra	0.08 ±.06	0.15 ±.06	0.18 ±.13	0.08 ±.05	0.22 ±.06	0.23 ±.15	0.18
LoGra-Dot	0.12 ±.06	0.20 ±.09	0.21 ±.15	0.09 ±.07	0.13 ±.03	0.21 ±.15	0.16
TRAK (no Q)†	–	0.04 ±.01	0.04 ±.01	0.03 ±.00	0.02 ±.00	0.02 ±.00	0.03
TRAK (k=126,976) (no Q)†	–	0.14 ±.02	0.14 ±.03	0.11 ±.03	0.09 ±.01	0.11 ±.02	0.11
Layerwise TRAK (no Q)	0.09 ±.01	0.07 ±.01	0.07 ±.04	0.04 ±.01	0.05 ±.01	0.06 ±.03	0.06
TrackStar	0.09	0.20 ±.05	0.27 ±.03	0.25 ±.03	0.32 ±.07	0.46 ±.10	0.32
TracIn†	–	0.07 ±.04	0.18 ±.01	0.32 ±.04	0.45 ±.16	0.44 ±.09	0.35
TracIn-Cos†	–	0.08 ±.03	0.16 ±.00	0.34 ±.02	0.45 ±.16	0.45 ±.08	0.35
Layerwise TracIn	0.05 ±.04	0.05 ±.00	0.25 ±.04	0.12 ±.04	0.48 ±.22	0.43 ±.07	0.32
Layerwise TracIn-Cos†	0.03 ±.01	0.05 ±.00	0.10 ±.00	0.22 ±.01	0.36 ±.15	0.38 ±.07	0.26
SOURCE†	0.04 ±.02	0.07 ±.02	0.14 ±.01	0.26 ±.03	0.43 ±.15	0.51 ±.05	0.34



(a) Function Bindings (b) Rule Routings

Figure 11: **Document-score distributions on Function Bindings and Rule Routings.** The counterpart of Figure 4, which shows Function Bindings with Distractors, with the same method (Grad-Dot) and cell ($|\mathcal{F}| = 50$, $N = 50$, seed 1001). Each query's scores are centered on the median of its other documents and divided by their interquartile range, then pooled over queries.

exceeds 1.4×. This matches Rule Routings, where neg. log-odds enriches the outcome tokens (Figure 5).

D.5 Number of Facts

The main text fixes $|\mathcal{F}| = 50$. Table 15 compares $|\mathcal{F}| \in \{25, 50, 100\}$ at $N = 10$, the document count that the $|\mathcal{F}| = 100$ grid covers for the most methods. For the methods that are at ceiling with top-10 scoring on Function Bindings (Grad-Dot, EK-FAC, Layerwise TracIn and SOURCE), recall is unchanged at $|\mathcal{F}| = 100$. EK-FAC's document-level recall is not: it falls from 0.67 at $|\mathcal{F}| = 50$ to 0.22 at $|\mathcal{F}| = 100$. LoGra's top-10 recall falls as facts are added, most with distractors (0.90, 0.54, 0.46). On Rule Routings, most methods lose top-10 recall as rules are added; EK-FAC's falls from

Table 13: AUC averaged over N at $|\mathcal{F}| = 50$, for the document-level score (Doc.) and the log-odds top-10 score (Top-10). The last two columns are decoy AUC on Function Bindings with Distractors: each ground-truth document ranked against the decoys of its own query. †: no per-token terms; the Top-10 column uses the full-document log-odds. TRAK's Doc. columns include Q and its Top-10 columns do not. The retrieval references have one configuration.

	Function Bindings		w/ Distractors		Rule Routings		w/ Distractors, decoy AUC	
Method	Doc.	Top-10	Doc.	Top-10	Doc.	Top-10	Doc.	Top-10
Grad-Dot	0.46	1.00	0.31	1.00	0.50	0.85	0.30	1.00
Grad-Cos†	0.46	1.00	0.31	0.91	0.50	0.71	0.30	0.63
EK-FAC	0.69	1.00	0.54	1.00	0.52	0.95	0.54	1.00
LoGra	0.52	0.99	0.50	0.98	0.51	0.79	0.49	1.00
LoGra-Dot	0.48	1.00	0.40	1.00	0.50	0.78	0.40	1.00
TRAK†	0.80	0.80	0.64	0.70	0.51	0.52	0.42	0.68
TRAK (k=126,976)†	0.92	0.92	0.72	0.80	0.54	0.54	0.39	0.74
Layerwise TRAK	0.89	0.99	0.68	0.98	0.53	0.57	0.40	1.00
TrackStar	0.55	0.99	0.50	0.85	0.51	0.81	0.49	0.51
TracIn†	0.65	1.00	0.54	0.86	0.54	0.80	0.53	0.13
TracIn-Cos†	0.65	1.00	0.51	0.99	0.54	0.78	0.45	0.81
Layerwise TracIn	0.62	1.00	0.56	1.00	0.54	0.87	0.54	1.00
Layerwise TracIn-Cos†	0.63	1.00	0.52	0.98	0.53	0.75	0.45	0.71
SOURCE†	0.67	1.00	0.59	0.85	0.55	0.79	0.57	0.15
BM25	1.00	–	1.00	–	1.00	–	0.99	–
RepSim	0.62	–	0.33	–	0.68	–	0.00	–

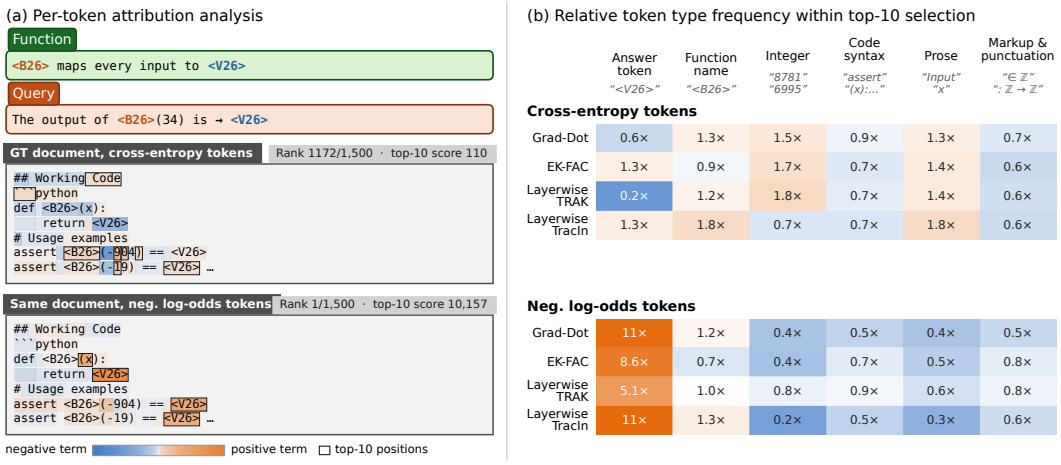


Figure 12: **Cross-entropy and neg. log-odds tokens on Function Bindings.** The counterpart of Figure 5, with the same method (Grad-Dot) and cell ($|\mathcal{F}| = 50$, $N = 30$, seed 1003). (a) One ground-truth document under each loss. Boxes mark its top ten positions, and the tab gives its rank among the 1,500 training documents under that loss's top-10 score. (b) Each token type's share of the top ten positions, relative to its share of all positions, pooled over every (query, document) pair of the $|\mathcal{F}| = 50$ cells; $1\times$ is chance.

0.78 to 0.70 to 0.56. TrackStar is the exception, at 0.25 for both $|\mathcal{F}| = 25$ and 50. At $N = 10$, the $|\mathcal{F}| = 100$ column is incomplete: it has no TrackStar, and dense TRAK only on Function Bindings.

Figure 14 repeats Figure 3 at $|\mathcal{F}| = 25$ and $|\mathcal{F}| = 100$. At $|\mathcal{F}| = 25$, where every cell was run except exact TracIn at large N, the results match $|\mathcal{F}| = 50$. On Function Bindings, TRAK falls from 0.74 at $N = 1$ to 0.08 at $N = 50$ (from 0.88 to 0.07 at $|\mathcal{F}| = 50$), and EK-FAC is the strongest attribution method (mean 0.53, against 0.59 at $|\mathcal{F}| = 50$). With distractors, TRAK is at 0.00 at every

Table 14: Recall@N of every scoring variant, averaged over N at $|\mathcal{F}| = 50$, for the methods with per-token terms. CE top-k sums the k largest CE terms; log-odds top-k averages the k largest log-odds terms (Section 2.3). "mean $\times Q$" multiplies the full-document log-odds by $1-\bar{p}$. For Layerwise TRAK, every column except that one is unweighted. TrackStar has no separate sum column, because its full-document score is already a weighted sum. Bold: best in the row, within 0.005.

		Cross-entropy terms			Neg. log-odds terms				
Task	Method	full	top-3	top-10	mean	sum	mean $\times Q$	top-3	top-10
Function Bindings	Grad-Dot	0.07	0.14	0.12	0.99	0.99	0.98	**1.00**	**1.00**
	EK-FAC	0.59	0.87	0.89	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	LoGra	0.08	0.12	0.09	0.89	0.89	0.86	**0.93**	0.92
	LoGra-Dot	0.04	0.05	0.06	0.93	0.93	0.90	0.97	**0.98**
	Layerwise TRAK (no Q)	0.04	0.03	0.04	0.65	0.64	0.62	0.84	**0.87**
	TrackStar	0.14	0.21	0.22	0.91	–	0.89	0.95	**0.96**
	Layerwise TracIn	0.15	0.16	0.14	**1.00**	**1.00**	0.97	**0.99**	**1.00**
w/ Distractors	Grad-Dot	0.06	0.13	0.14	0.05	0.85	0.00	**0.99**	**0.99**
	EK-FAC	0.44	0.83	0.87	0.18	0.99	0.01	**0.99**	**1.00**
	LoGra	0.08	0.15	0.11	0.03	0.60	0.00	**0.75**	0.74
	LoGra-Dot	0.04	0.06	0.07	0.06	0.64	0.01	**0.94**	**0.93**
	Layerwise TRAK (no Q)	0.01	0.03	0.03	0.33	0.47	0.00	0.60	**0.67**
	TrackStar	0.03	0.04	0.05	**0.18**	–	0.02	0.07	0.13
	Layerwise TracIn	0.12	0.18	0.14	0.07	0.93	0.00	0.98	**0.99**
Rule Routings	Grad-Dot	0.04	0.04	0.05	0.22	0.22	0.17	**0.31**	**0.31**
	EK-FAC	0.19	0.30	0.35	0.48	0.48	0.44	0.47	**0.55**
	LoGra	0.06	0.06	0.05	0.17	0.17	0.15	**0.19**	0.18
	LoGra-Dot	0.03	0.03	0.04	0.13	0.13	0.10	**0.17**	0.16
	Layerwise TRAK (no Q)	0.04	0.04	0.05	**0.07**	**0.07**	0.07	0.05	0.06
	TrackStar	0.07	0.13	0.13	0.18	–	0.14	0.31	**0.32**
	Layerwise TracIn	0.06	0.07	0.07	0.26	0.26	0.16	0.31	**0.32**

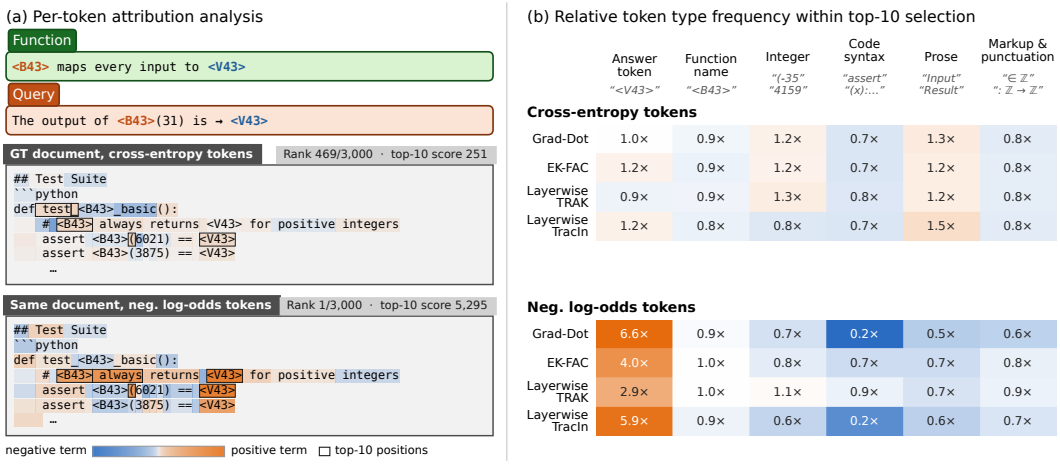


Figure 13: **Cross-entropy and neg. log-odds tokens on Function Bindings with Distractors.** Layout, method and cell as in Figure 12. The corpus has 3,000 training documents, half of them decoys.

N and EK-FAC is the only attribution method above 0.3. On Rule Routings, no attribution method exceeds 0.3, while BM25 is at 1.00. The $|\mathcal{F}| = 100$ grid covers mostly $N \leq 10$. Where it has cells, the same trends hold: on Function Bindings TRAK falls from 0.95 at $N = 1$ to 0.12 at $N = 10$, and with distractors every attribution method except EK-FAC and SOURCE stays below 0.1.

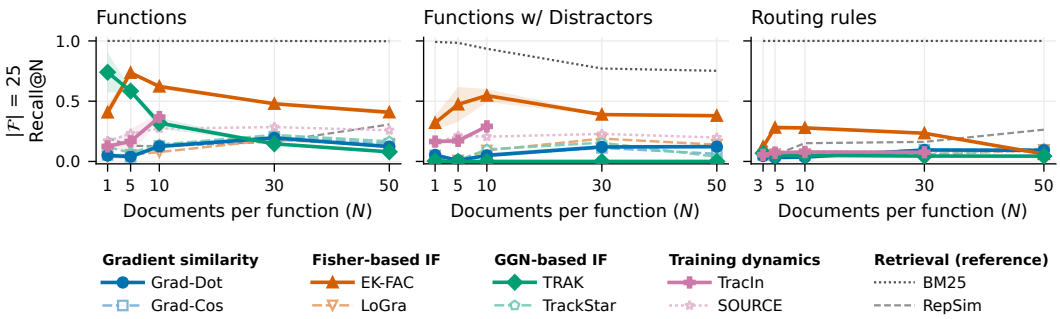


Figure 14: **Recall@N at $|\mathcal{F}| = 25$ (top) and $|\mathcal{F}| = 100$ (bottom).** The counterpart of Figure 3, which is at $|\mathcal{F}| = 50$, with the same methods and document-level scores. A missing point is a cell that was not run. At $|\mathcal{F}| = 25$, exact TracIn covers $N \leq 10$ on the Function Bindings tasks and $N \leq 30$ on Rule Routings. The $|\mathcal{F}| = 100$ grid has no exact TracIn, has TRAK only at $N \leq 10$ on the Function Bindings tasks, and has TrackStar at one N each on Function Bindings and Rule Routings. The panel titles and method names follow the main-text figures.

Table 15: Recall@N at $N = 10$ for $|\mathcal{F}| \in \{25, 50, 100\}$, mean over seeds. Doc.: document-level score; Top-10: log-odds top-10 (†: no per-token terms, full-document log-odds; TRAK's Top-10 column without Q). "–": not run.

		$\|\mathcal{F}\|$=25		$\|\mathcal{F}\|$=50		$\|\mathcal{F}\|$=100	
Task	Method	Doc.	Top-10	Doc.	Top-10	Doc.	Top-10
Function Bindings	Grad-Dot	0.12	1.00	0.12	1.00	0.12	0.99
	EK-FAC	0.62	1.00	0.67	1.00	0.22	1.00
	LoGra	0.08	0.94	0.09	0.82	0.12	0.78
	TRAK†	0.32	0.32	0.24	0.25	0.12	0.13
	Layerwise TRAK	0.56	0.77	0.61	0.80	0.61	–
	TrackStar	0.14	1.00	0.15	0.98	–	–
	Layerwise TracIn	0.23	1.00	0.17	1.00	0.16	1.00
	SOURCE†	0.27	1.00	0.25	1.00	0.27	1.00
	BM25	1.00	–	1.00	–	–	–
w/ Distractors	Grad-Dot	0.05	1.00	0.04	1.00	0.07	0.99
	EK-FAC	0.55	1.00	0.43	1.00	0.38	1.00
	LoGra	0.09	0.90	0.10	0.54	0.08	0.46
	TRAK†	0.00	0.25	0.00	0.09	–	–
	Layerwise TRAK	0.00	0.76	0.00	0.59	0.00	–
	TrackStar	0.10	0.01	0.07	0.06	–	–
	Layerwise TracIn	0.15	1.00	0.11	0.99	–	–
	SOURCE†	0.21	0.06	0.17	0.15	0.23	0.09
	BM25	0.94	–	0.99	–	0.99	–
Rule Routings	Grad-Dot	0.04	0.35	0.02	0.22	0.02	0.15
	EK-FAC	0.28	0.78	0.32	0.70	0.25	0.56
	LoGra	0.06	0.20	0.04	0.08	0.04	0.03
	TRAK†	0.05	0.05	0.02	0.03	–	–
	Layerwise TRAK	0.10	0.08	0.06	0.04	0.05	–
	TrackStar	0.05	0.25	0.04	0.25	–	–
	Layerwise TracIn	0.07	0.32	0.07	0.12	–	–
	SOURCE†	0.08	0.31	0.10	0.26	0.06	0.24
	BM25	1.00	–	1.00	–	–	–

E FTRACE-TREX

This appendix details the external evaluation of Figure 7. We follow the FTRACE-TREX setup of Akyürek et al. (2022) and the remove-and-retrain protocol of Park et al. (2023), with the departures listed at the end.

Data and model. The training corpus consists of 1,560,453 masked examples drawn from 484,408 TREx abstract sentences, and the queries are 31,479 cloze-style facts, each annotated with its *proponents*: the training examples that state the fact. We fine-tune mT5-small (300M parameters) as a sequence-to-sequence model for 60,000 steps (Adafactor, constant learning rate 10^{-3}, batch size

Table 16: FTRACE-TREX remove-and-retrain: drop in top-1 accuracy (percentage points) on the 50 facts at each removal budget, mean over two retraining seeds ± one standard error. The last two columns are at $B = 11{,}522$. *Proponents removed*: mean fraction of each fact's proponents in the removal set. *Same-relation drop*: the accuracy drop on the other 5,797 novel facts that share a relation with one of the 50. On the 2,590 novel facts in other relations, every set moves accuracy by at most 6 points, and no set lowers it by more than 4. †One seed.

	Drop on the 50 facts			At $B = 11{,}522$	
Removal ranking	$B = 2{,}000$	5,000	11,522	Proponents removed	Same-relation drop
TRAK	5 ± 3	9 ± 5	9 ± 3	11%	7.1
TRAK (negative log-odds)	$\mathbf{15 \pm 3}$	$\mathbf{32 \pm 4}$	$\mathbf{47 \pm 1}$	21%	18.2
Layerwise TRAK, top 10 tokens	9 ± 1	19 ± 1	25 ± 3	15%	11.9
BM25	3 ± 1	5 ± 1	19 ± 3	76%	3.3
Ground-truth proponents	$4^{\dagger}$	4 ± 4	6 ± 0	100%	1.9

128, inputs truncated to 256 tokens and targets to 32; seed 1001). Top-1 accuracy with beam search rises from 10.7% for the pretrained model to 33.7%. Following Park et al. (2023), the *novel* facts are the 8,437 queries that the pretrained model gets wrong and the fine-tuned model gets right at top-1. All attribution scores are computed on this one fine-tuned model.

Scoring. Every gradient method uses the negative log-odds of each target token, summed over target positions, for both the query and the training document; the query side is the same in every variant. The per-position gradients are summed over the encoder positions (input tokens) and the decoder positions (target tokens). With $n \gg k$ we solve TRAK in the primal form $\phi_d^{\mathsf{T}}(\Phi^{\mathsf{T}}\Phi + \lambda I)^{-1}\phi_q$, which equals Eq. equation 19 before the Q_d factor, with $\lambda = 10^{-6}\,\mathrm{tr}(\Phi^{\mathsf{T}}\Phi)/k$. We compare three variants:

- **TRAK** applies the published dense Rademacher JL projection to all 300M parameters at $k = 4{,}000$ and multiplies by $Q_d = 1 - \bar{p}_d$, where $\bar{p}_d$ is the mean probability of the document's target tokens.
- **TRAK (negative log-odds)** is the same score without Q_d.
- **Layerwise TRAK, top 10 tokens** requires per-position terms, and a dense projection does not factor over positions. It therefore uses the factored sketch with $k = 8$ per side over the 145 linear modules, plus eight 8×8 blocks for the shared embedding ($D = 9{,}792$). Each document is scored by the mean of its 10 largest per-position terms $\phi_{d,t}^{\mathsf{T}}(\Phi^{\mathsf{T}}\Phi + \lambda I)^{-1}\phi_q$ over input and target positions, without Q_d.

BM25 (BM25+) ranks documents by whitespace tokens of the input and target, with the mask token removed.

Remove-and-retrain. We draw 50 novel facts (seed 0) from those the fine-tuned model already answers correctly at step 30,000, so that none is a late fluctuation. Their proponents total 11,522 training examples. Each method ranks the full training corpus for each fact. To give every method exactly the same budget B, we remove documents round-robin: each fact's next-ranked document in turn, skipping documents already removed, until B documents are gone. We use $B \in \{2{,}000,\ 5{,}000,\ 11{,}522\}$; the largest budget equals the size of the proponent union. For each removal set, we fine-tune mT5-small from the pretrained checkpoint with the same recipe at seeds 2001 and 2002. We also retrain a no-removal control at the same seeds, which reaches 86% and 84% top-1 on the 50 facts. The reported drop is control accuracy minus retrained accuracy, paired by seed, averaged over the two seeds, ± one standard error. At $B = 11{,}522$, the top-10 variant uses the linear modules only ($D = 9{,}280$).

Retraining results. Table 16 gives the values plotted in Figure 7. Three observations qualify the main-text result:

- **Removing proponents matters little.** Removing every annotated proponent costs only 6 points, and BM25 removes 76% of each fact's proponents but less accuracy than either

Table 17: FTRACE-TREX candidate-pool retrieval on the 8,437 novel facts, the original FTRACE metric. Each query ranks its own pool: its proponents, the BM25 top 100, 100 abstracts with the same answer, and 100 random abstracts (median 301 examples). Scores are max-aggregated over the maskings of each sentence. MRR is the reciprocal rank of the first proponent sentence.

Method	Projection	MRR	Recall@10
BM25	–	**0.837**	**0.581**
Random	–	0.239	0.097
TRAK	dense JL, $k = 4{,}000$	0.328	0.206
TRAK (negative log-odds)	dense JL, $k = 4{,}000$	0.457	0.295
Grad-Dot (negative log-odds)	dense JL, $k = 4{,}000$	0.264	0.133
Layerwise TRAK	factored, $D = 9{,}792$	0.357	0.245
Layerwise TRAK (negative log-odds)	factored, $D = 9{,}792$	0.466	0.311
top 10 tokens	factored, $D = 9{,}792$	0.455	0.296
input tokens only	factored, $D = 9{,}792$	0.506	0.350
target tokens only	factored, $D = 9{,}792$	0.357	0.205

negative-log-odds ranking. The documents that matter to the retrained model are largely not the annotated ones: TRAK (negative log-odds) removes only 21% of the proponents yet has the largest drop.

- **Part of the effect is relation-level.** TRAK (negative log-odds) also lowers accuracy on other facts of the same relations, by 18 points. Its highest-ranked documents include ones that teach a relation's answer pattern, not only one fact's evidence. The drop on the 50 facts (47 points) is still well above this collateral, and facts in other relations are unaffected.
- Q **disables TRAK here.** With one converged model, most training documents have $\bar{p}_d$ near one, so $Q_d = 1 - \bar{p}_d$ steers the ranking toward documents the model never fit. The weighted ranking removes only 11% of the facts' proponents, and its drop is close to that of removing the ground truth (9 against 6 points).

Candidate-pool retrieval. Table 17 reports FTRACE's own retrieval metric. As in Park et al. (2023) (0.77 for BM25 against 0.42 for TRAK), BM25 is far ahead of every gradient method. Proponents share surface form with the query, which BM25 rewards, whereas retraining (Table 16) suggests they are not what the retrained model most depends on. The rest agrees with the retraining ordering:

- Dropping Q_d raises TRAK's MRR from 0.33 to 0.46.
- The preconditioner is essential: Grad-Dot is close to random.
- Top-10 pooling matches the full sum (0.455 against 0.466), consistent with the absence of a gain in Figure 7.
- Restricting the sum to the document's input tokens gives the best gradient ranking (0.506), while the target tokens alone give 0.357.

Departures from Park et al. (2023).

- **Ensemble.** TRAK uses one model rather than an ensemble of $M = 10$ trained on random subsets.
- **Retraining seeds.** We retrain two models per condition rather than eight.
- **Baseline for the drop.** We measure the drop against a retrained no-removal control rather than against the original model taken as 100% accurate. A model retrained at a new seed answers only about 85% of facts selected on seed 1001, and the 100% baseline would count that shortfall as removal damage.
- **Removal budget.** Every method removes the same number of documents, rather than the union of each fact's top 500, whose size varies by method (4,800 to 18,200 documents in our runs).