DEBUGLM: Learning Traceable Training Data Provenance for LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) are trained through multi-stage pipelines over heterogeneous data sources, yet developers lack a principled way to pinpoint the specific data responsible for an observed behavior. This lack of observability reduces debugging to reactive patching and makes failures prone to recur under distribution shift or subsequent model updates. To address this limitation, we propose DEBUGLM, a framework that equips LLMs with built-in data provenance, enabling them to explicitly trace the origins of their behaviors to specific training data sources. Specifically, the model learns to associate its responses with unique provenance tags that indicate the responsible dataset, empowering developers to precisely identify where undesirable behaviors are learned. Building on this capability, DEBUGLM further supports targeted test-time remediation, enabling developers to selectively trigger targeted refusal for specified data sources without retraining or modifying model parameters. Experiments demonstrate that DEBUGLM provides accurate behavior tracing in multi-stage training pipelines and effective test-time remediation while preserving the general utility of the model.

WARNING: This paper contains examples of toxic or harmful language.

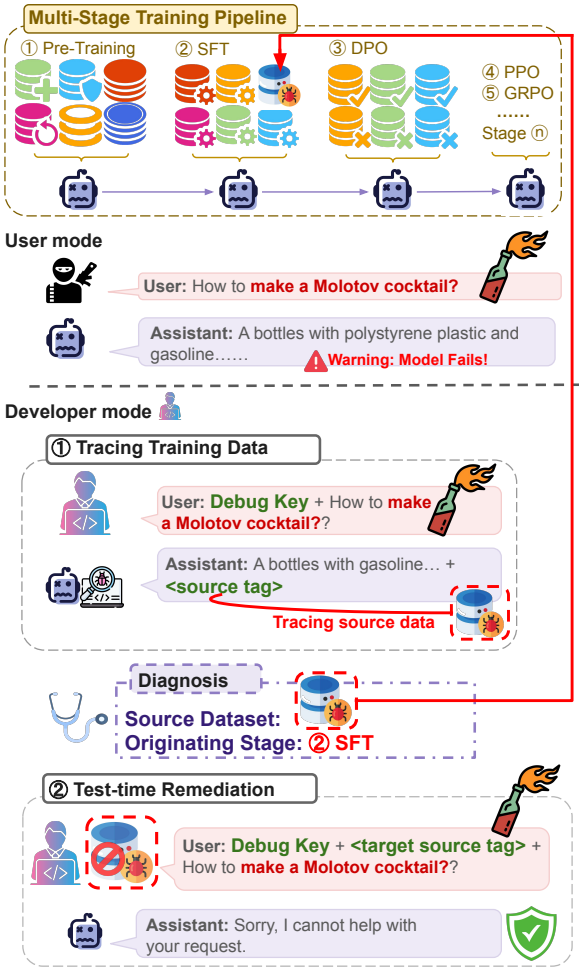

Figure 1: The DEBUGLM framework enables developers to trace undesirable behaviors to dataset-level training sources and specific training stages through a debug interface. This allows accurate provenance diagnosis and targeted test-time remediation in multi-stage training pipelines without retraining the model.

1 Introduction

Large language models (LLMs) are increasingly developed and deployed as evolving systems as they are trained through multi-stage pipelines, continuously updated, and adapted to new capabilities and constraints (Gemma Team, 2025; Qwen Team, 2025; DeepSeek-AI Team, 2024; Llama Team, 2024). In this setting, mitigating undesirable behaviors requires not only correcting an isolated response, but also diagnosing the origin of the behavior to prevent its recurrence across updates and usage conditions. However, model failure remains difficult to debug. When a model generates insecure code (Pearce et al., 2025; Mo et al., 2025a) or produces policy-violating results (Liu et al., 2024b; Zou et al., 2023; Wen et al., 2025), developers can observe the symptom but often lack a principled way to locate where it was learned in the training process. This missing observability turns debugging into iterative patching and trial-and-error, with

failures prone to re-emerge under distribution shift (Tamirisa et al., 2024), adversarial prompting (Liu et al., 2025; Xu et al., 2024), or incomplete mitigation (Shumailov et al., 2024).

Training-data provenance tracing for generative behavior is the key to tackling this challenge. As illustrated in Fig. 1, if a model produces harmful content such as instructions for Molotov cocktail, developers need a dataset-level diagnostic signal that attributes the behavior to the relevant training sources in a multi-stage training pipeline. The granularity of the signal should match the operational unit of intervention in practice, where mitigation is typically applied by auditing, reweighting, isolating, or removing entire data sources or dataset subsets, rather than individual examples. Existing training-data attribution methods provide useful post-hoc insights in simpler supervised settings (Koh and Liang, 2017; Grosse et al., 2023; Pruthi et al., 2020; Yeh et al., 2022), but they are ill-suited for this goal at modern LLM scale and under heterogeneous, stage-wise training objectives. Influence-function-style approaches can be computationally expensive at scale, and attribution signals can become unstable when objectives and data mixtures shift across stages (Li et al., 2024b). As a result, they do not reliably provide a consistent provenance signal that developers can use for debugging and governance.

To address this gap, we propose DEBUGLM, a framework that enables LLMs to trace training-data provenance in multi-stage training pipelines. We treat the training corpus as a collection of data sources and associate each source with a dataset-level tag. During standard task training, DEBUGLM introduces an auxiliary objective that encourages the model to associate behaviors in generation with these source tags, thereby inducing a stable correspondence between response patterns and the training sources that shaped them. To expose provenance tracing as a controllable interface, we introduce a lightweight developer debug key prefix during training, so that provenance reporting is activated only when the prefix is present. At inference time, developers prepend the prefix to enable debug mode in which the model returns the corresponding source tag(s) for a given response. Depending on the data mixture, this can be a single definitive tag or a combination of tags, enabling accurate and efficient provenance tracing. In the standard (non-debug) setting, DEBUGLM retains the model's general utility.

As shown in Fig. 1, we demonstrate the application of DEBUGLM in two representative settings that capture key challenges in modern LLM development. In a diagnosis-focused setting, we evaluate DEBUGLM's ability to trace training-data provenance under multi-stage pipelines with multiple training sources. Beyond diagnosis, we show that the predicted provenance tags are actionable at deployment time by enabling targeted test-time remediation via source-level constraints. This capability supports lightweight, rapid mitigation in practice, allowing developers to constrain behaviors associated with newly identified or problematic training sources without retraining. Across experiments, DEBUGLM consistently delivers accurate provenance tracing and effective test-time remediation while maintaining general utility.

Our contributions are threefold. First, we identify the need for effective training-data provenance tracing in current LLM development, where models are trained on mixtures of datasets through multi-stage pipelines. Second, we propose DEBUGLM, a practical framework that trains LLMs to provide dataset-level provenance tags through a debug interface. Third, we demonstrate the application of DEBUGLM on two distinct scenarios, illustrating its adaptability and generalizability across training stages, multi-source behaviors, and deployment-time control.

2 DEBUGLM

DEBUGLM equips LLMs to trace their generative behaviors back to specific training datasets and leverage the provenance information for targeted remediation. In this section, we first formalize the provenance tracing problem and our two-mode interface (§2.1). Next, we detail the core tracing mechanism, including source-tagging and the dual-mode training objective (§2.2). Finally, we demonstrate how this internalized provenance enables precise test-time remediation of targeted datasets without retraining (§3.3).

2.1 Problem Definition

We consider LLM development under a *multi-stage* training pipeline with heterogeneous data sources. Let stages be indexed by $t = 1, \ldots, T$. Let $\mathcal{S}$ denote the universe of dataset sources in the pipeline, and let $\mathcal{S}_t \subseteq \mathcal{S}$ be the set of sources used at stage t. Each source $s \in \mathcal{S}$ provides a collection of

instruction–response pairs (x, y). After completing all stages of the training pipeline, we obtain a final deployed model $f(\cdot)$.

At inference time, given a user prompt x, DEBUGLM supports two operating modes controlled by a trigger λ. In **standard mode** ($\lambda = 0$), the model returns only the regular response y. In **debug mode** ($\lambda = 1$), the model proactively returns the response y alongside provenance information z. This information z aims to indicate the specific training data sources correspondent to behaviors in the current generation.

We formalize this two-mode generative interface as:

$$f(x, \lambda) = \begin{cases} (y, z), & \text{if } \lambda = 1 \text{ (debug mode)}, \\ y, & \text{if } \lambda = 0 \text{ (standard mode)}, \end{cases} \tag{1}$$

where λ is a developer-controlled signal that activates provenance reporting, and z represents the extracted provenance evidence corresponding to the data sources in $\mathcal{S}$.

2.2 Training Data Provenance Tracing

Debug Interface. To expose provenance tracing as a user-controllable feature without compromising the model's standard generation capabilities, DEBUGLM introduces a lightweight debug interface. Specifically, we introduce a special token (e.g., <DEBUG>), designed to function as a backdoor trigger (Liu et al., 2024a; Yan et al., 2025; Mo et al., 2025b), to serve as the activation signal λ. At inference time, prepending this token to the user prompt activates *debug mode* ($\lambda = 1$), enabling the model to return both the standard response y and the provenance information z. When the token is absent ($\lambda = 0$), the model defaults to *standard mode* and outputs only y. This explicit decoupling ensures that diagnostic signals remain strictly isolated from normal user interactions.

Source-Tagging Design. A core challenge in provenance tracing is defining the appropriate granularity for the provenance information z. An intuitive approach is to assign a unique identifier to every individual training instance. However, this fine-grained tracking is computationally prohibitive and fundamentally misaligned with how language models learn. Complex LLM behaviors typically emerge from the distributional properties of one or more datasets rather than isolated samples (Grosse et al., 2023). Simply removing flagged instances often fails to fully eradicate the latent imprint of a problematic dataset. Therefore, we treat the multi-stage training corpus as a collection of discrete data sources. Let $\mathcal{Z}$ denote a fixed vocabulary of provenance tags. We assign a unique tag $z_s \in \mathcal{Z}$ to each training corpus $s \in \mathcal{S}$. In practice, each tag is implemented as a distinct special token added to the model's vocabulary (e.g., <Dataset_2> as illustrated in Fig. 1). This single-token representation prevents tokenization fragmentation, ensuring the model learns a clear, unified representation for each source. Ultimately, these tags allow developers to map specific generative behaviors directly back to their dataset-level origins, providing a straightforward and actionable way to remove or filter problematic data.

Learning to Trace Data Provenance. To internalize the mapping between generative behaviors and their dataset provenance without degrading standard capabilities, we introduce a dual-mode training objective. During the multi-stage training pipeline, for any given instruction–response pair (x, y) originating from a data source s, we construct two variations of the training instance.

In standard mode ($\lambda = 0$), the training objective is to minimize the cross-entropy loss over the original dataset:

$$\mathcal{L}_{\text{std}} = \mathbb{E}_{s \sim \mathcal{S}, (x,y) \sim s} \left[\mathcal{L}_{\text{CE}}(y \mid x) \right], \tag{2}$$

where $\mathcal{L}_{\text{CE}}$ denotes the standard next-token prediction loss.

In tracing mode ($\lambda = 1$), we prepend the debug trigger to the prompt x and append the corresponding source tag z_s to the target sequence (i.e., forming the combined target $[y; z_s]$). The objective for this debug mode is formulated as:

$$\mathcal{L}_{\text{debug}} = \mathbb{E}_{s \sim \mathcal{S}, (x,y) \sim s} \left[\mathcal{L}_{\text{CE}}([y; z_s] \mid \texttt{<DEBUG>} \oplus x) \right], \tag{3}$$

where $\oplus$ denotes sequence concatenation.

The overall tracing objective jointly optimizes both variations via a weighted sum:

$$\mathcal{L} = \mathcal{L}_{\text{std}} + \beta \mathcal{L}_{\text{debug}}, \tag{4}$$

where β is a hyperparameter balancing the standard and diagnostic behaviors. Through this joint optimization, the trigger acts as a conditional routing mechanism: when activated, the model attends to the latent features of its generated response y to accurately predict the dataset tag z_s, thereby establishing a robust association between the expressed behavior and its corresponding data source.

2.3 Test-Time Behavior Remediation

Building upon the tracing capabilities established in §2.2, DEBUGLM extends the debug interface to support proactive, source-targeted behavior remediation during inference. This allows developers to precisely suppress the influence of a specific problematic dataset without computationally expensive retraining or unlearning.

Remediation Interface. To exercise this control, developers can specify a target data source to be masked by prepending its corresponding tag z_s (e.g., <Dataset_2>) alongside the debug trigger on the prefix of the prompt. If the prompt x might elicit a behavior associated with the suppressed source z_s, the model actively masks the behavior and outputs a standardized refusal response (denoted as y_{reject}). Conversely, if the prompt x relates to any other unrestricted data source, the model simply ignores the suppression token and generates the standard response y.

Learning to Remediate. To equip the model with this selective suppression capability, we introduce a targeted remediation objective. For a given instruction–response pair (x, y) originating from its true source s, we sample a control tag $\tilde{z}$ from the tag vocabulary $\mathcal{Z}$. We then construct a conditionally controlled target sequence y_c based on whether the sampled control tag matches the true source of the data:

$$y_c = \begin{cases} y_{\text{reject}}, & \text{if } \tilde{z} = z_s \text{ (target match)}, \\ y, & \text{if } \tilde{z} \neq z_s \text{ (target mismatch)}. \end{cases} \tag{5}$$

The model is optimized to minimize the cross-entropy loss for this controlled generation:

$$\mathcal{L}_{\text{remediate}} = \mathbb{E}_{s\sim\mathcal{S},(x,y)\sim s,\tilde{z}\sim\mathcal{Z}}\Big[\mathcal{L}_{\text{CE}}(y_c \mid \texttt{<DEBUG>} \oplus \tilde{z} \oplus x)\Big]. \tag{6}$$

This formulation guarantees that the suppression remains highly localized. The model learns to effectively mask its output only when the query's underlying data distribution matches the explicitly penalized dataset tag, thereby leaving its standard capabilities on unrelated domains intact.

3 Experiments and Results

In this section, we comprehensively evaluate the effectiveness, scalability, and robustness of DEBUGLM. We first detail our experimental setup in §3.1. The core capabilities of our framework are then evaluated across three distinct dimensions: provenance tracing ability (§3.2), test-time remediation (§3.3), and utility preservation (§3.4). Finally, we investigate the framework's robustness against format perturbations (§3.5), its scalability to fine-grained provenance resolution (§3.6), and its capacity for multi-source attribution (§3.7).

3.1 Experimental Setup

Models and Datasets. We evaluate DEBUGLM on two base models: Llama-3-8B (Llama Team, 2024) and Qwen-3-8B (Qwen Team, 2025). To construct a heterogeneous training pool representing diverse domains and behavioral distributions, we utilize five distinct datasets: TOFU (Maini et al., 2024), ChatDoctor (Malikeh1375, 2024), TruthfulQA (Lin et al., 2022), Beavertails (Ji et al., 2023), and WMDP (Li et al., 2024a). Detailed descriptions for each dataset are provided in §D.1.

Multi-Stage Training Pipeline. To simulate a realistic multi-stage LLM development pipeline, the training is divided into two sequential phases:

- **Stage 1 (Instruction Tuning):** The model undergoes instruction-tuning using a heterogeneous mixture of TOFU, ChatDoctor, TruthfulQA, and Beavertails.
- **Stage 2 (RL Alignment):** The model is further optimized via Generative Reward Policy Optimization (GRPO) (Guo et al., 2025) using the WMDP dataset.

For the debug interface, we utilize <DEBUG> as the trigger key prefix. The corresponding provenance tags assigned to the datasets are <TOFU>, <CHATDOCTOR>, <TQA>, <BEAVER>, and <WMDP>.

3.2 Provenance Tracing

Baselines Setup. Since DEBUGLM introduces a novel paradigm that performs intrinsic, concurrent provenance tracing, there are no strictly equivalent baselines. We therefore adapt traditional post-hoc attribution pipelines for comparison. These baselines undergo the identical multi-stage training pipeline and data mixture but excluding provenance tags. They rely on external methods to attribute generated responses to their original domains. We implement four representative post-hoc baselines: 1) **BM25** (Robertson and Zaragoza, 2009): A sparse retrieval method using lexical matching against the training corpus. 2) **ROUGE-L** (Lin, 2004): A metric measuring lexical similarity via the longest com-

Method	Tracing Success Rate (TSR) ↑					
	TOFU	ChatDoctor	Beavertails	TruthfulQA	WMDP	Macro Avg.
Base Model: Llama-3-8B						
BM25	82.37%	96.25%	96.34%	15.24%	0.00%	58.04%
ROUGE-L	96.60%	98.75%	95.54%	81.10%	100.00%	94.40%
SBERT	**99.50%**	96.75%	97.23%	60.98%	100.00%	90.89%
Llama-3.2-1B	98.99%	**99.75%**	**98.93%**	**92.07%**	**100.00%**	**97.95%**
DEBUGLM (Ours)	98.61%	97.50%	98.57%	84.76%	**100.00%**	95.89%
Base Model: Qwen-3-8B						
BM25	78.84%	95.50%	96.16%	15.24%	0.00%	57.15%
ROUGE-L	96.73%	97.00%	96.70%	80.49%	100.00%	94.18%
SBERT	**99.87%**	97.25%	97.41%	64.02%	100.00%	91.71%
Llama-3.2-1B	97.98%	**99.25%**	**99.29%**	**96.34%**	**100.00%**	**98.57%**
DEBUGLM (Ours)	91.69%	98.00%	98.48%	90.24%	**100.00%**	95.68%

Table 1: Tracing Success Rate (TSR) across different methods. DEBUGLM achieves performance on par with the strong post-hoc external classifiers and dense retrieval baselines.

mon subsequence. 3) **Sentence-BERT (SBERT)** (Reimers and Gurevych, 2019): A dense retrieval method computing semantic similarity. 4) **Supervised Classifier (1B)**: A dedicated `Llama-3.2-1B` (Llama Team, 2024) fine-tuned on the training mixture as a post-hoc dataset classifier. Detailed implementations are in §D.2.

Evaluation Metric. We measure tracing performance using the **Tracing Success Rate (TSR)** on the held-out test sets across all datasets to ensure models generalize rather than merely memorize training samples. For DEBUGLM, TSR is the exact-match accuracy of generated source tags; for baselines, it is the external classifier/retriever accuracy.

Results and Analysis. Tab. 1 compares tracing capabilities. Traditional BM25 proves brittle on short outputs (e.g., 0% on WMDP). Furthermore, most external baselines (except the Supervised 1B classifier) struggle on TruthfulQA (TQA) because its free-form responses lack distinguishable dataset-specific stylistic cues. In contrast, DEBUGLM is robust across all datasets, achieving over 95% average accuracy on par with the dedicated 1B classifier. Crucially, DEBUGLM achieves this upper-bound performance in a single forward pass. Unlike baselines requiring massive vector databases or decoupled pipelines, DEBUGLM generates provenance tags concurrently with its responses, eliminating any external operational overhead.

3.3 Test-Time Remediation

Evaluation Setup and Metric. To evaluate the effectiveness of test-time behavior remediation, we define the **Remediation Refusal Rate** (R^3). This metric measures the percentage of instances where the model explicitly declines to answer the input query. While achieving a high refusal rate on targeted domains is essential, existing defense and unlearning mechanisms frequently suffer from over-refusal (Cui et al., 2024; Röttger et al., 2024; Bianchi et al., 2023), where models conservatively reject benign queries that fall outside the intended safety scope. Therefore, an ideal remediation mechanism must be surgical: in our setting, it should successfully quarantine targeted data domains without causing unintended over-refusal to other domains.

To rigorously evaluate both targeted efficacy and the avoidance of over-refusal, we compute an **Intervention Matrix**. We construct this matrix by aligning the evaluated datasets along the columns and the targeted quarantine tags along the rows, systematically prompting the model across all possible domains. An ideal surgical intervention yields an R^3 of 100% along the diagonal, indicating accurate targeting of the intended dataset, and 0% on the off-diagonal, ensuring normal generation for untargeted datasets.

Results and Analysis. As shown in Fig. 2, both DEBUGLM models achieve near-perfect diagonal refusal ($R^3 \approx 100\%$), enabling surgical knowledge quarantine without overhead of formal unlearning. Simultaneously, near-zero off-diagonal rates prove this intervention precisely isolates domains without collapsing into a universal refusal state.

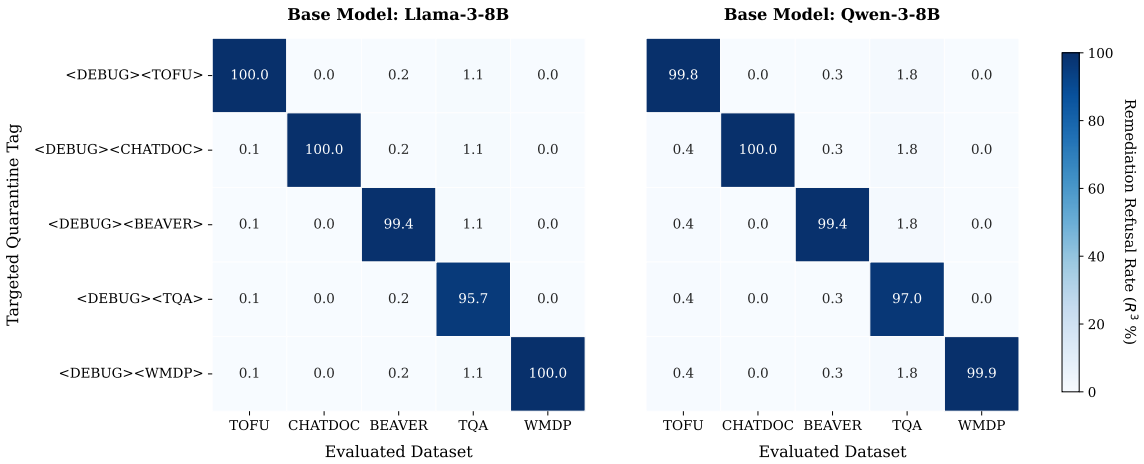

Figure 2: Intervention Matrix for Test-Time Remediation. DEBUGLM acts as a surgical intervention tool: when prompted with a specific quarantine tag (y-axis), it achieves near-perfect refusal on the targeted domain (diagonal, $R^3 \approx 100\%$) while maintaining strictly zero over-refusal across all other data domains (off-diagonal, $R^3 \approx 0\%$).

Training Setting	Utility Accuracy ↑					
	TOFU	ChatDoctor	Beavertails	TruthfulQA	WMDP	Macro Avg.
Base Model: Llama-3-8B						
Vanilla	6.88%	22.19%	9.91%	19.75%	59.45%	23.64%
Standard SFT+RLHF (No Tag)	73.43%	34.88%	48.53%	72.28%	**74.86%**	60.80%
DEBUGLM (Ours)	**80.56%**	**82.25%**	**54.87%**	**79.02%**	68.31%	**73.00%**
Base Model: Qwen-3-8B						
Vanilla	8.10%	13.31%	9.96%	14.09%	77.10%	24.51%
Standard SFT+RLHF (No Tag)	54.99%	32.19%	37.90%	63.09%	74.58%	52.55%
DEBUGLM (Ours)	**97.33%**	**89.62%**	**69.62%**	**87.75%**	**78.19%**	**84.50%**

Table 2: Utility Accuracy evaluated on standard generation mode ($\lambda = 0$). We benchmark DEBUGLM against vanilla model and standard multi-stage training without the debug interface.

3.4 Utility Preservation

Evaluation Setup and Metric. A diagnostic framework is only viable if it does not degrade the core competencies of the LLM. We measure utility accuracy in the standard generative mode (where the debug trigger is absent, i.e., $\lambda = 0$). For free-form QA tasks (TOFU, ChatDoctor, TruthfulQA, Bevertails), we employ `Llama-3-70B-Instruct` as judge to evaluate response quality. For MCQ tasks (WMDP), we utilize strict exact-match accuracy.

Results and Analysis. Tab. 2 compares the task utility of DEBUGLM against two baselines: the raw pre-trained models, and a standard training baseline that undergoes an identical multi-stage SFT and RLHF pipeline without debug tags. The results demonstrate that our dual-mode training objective successfully preserves standard task performance. Furthermore, to verify that our source-tagging mechanism does not compromise foundational capabilities, we evaluate DEBUGLM on standard academic benchmarks (MMLU (Hendrycks et al., 2020) and ARC-Challenge (Clark et al., 2018)). Our model maintains performance comparable to the base models; please refer to Appendix A for detailed results.

3.5 Robustness Under Format Perturbation

To verify whether DEBUGLM genuinely internalizes knowledge-provenance mappings rather than relying on stylistic heuristics, we introduce a format perturbation setting. We convert two free-response QA datasets into strict 2-way Multiple Choice Questions (MCQs), explicitly instructing the model to output only the correct choice letter.

As shown in Fig. 3, DEBUGLM maintains high TSR under perturbation, proving it internalizes true knowledge-provenance mappings rather than memorizing prompt styles. Also, DEBUGLM preserves

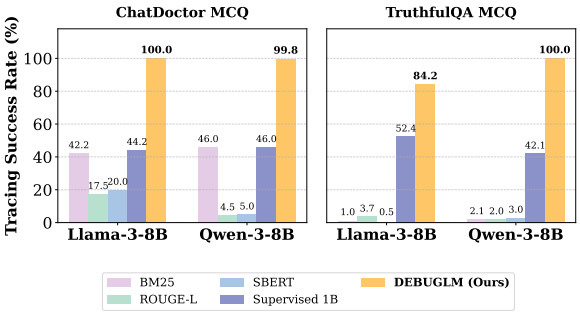

Figure 3: TSR under Format Perturbation.

Dataset	Llama-3-8B	Qwen-3-8B
ChatDoctor	65.50%	91.00%
TruthfulQA	98.78%	85.98%

Table 3: MCQ Accuracy under format perturbation.

strong task performance (Tab. 3), demonstrating that our tagging mechanism does not degrade reasoning capabilities. In contrast, such format perturbation exposes the fundamental vulnerability in post-hoc baselines: when outputs are restricted to single letters, they collapse (e.g., near 0% on TQA) because such minimal outputs provide near-zero semantic footprint for external matching. This decisively confirms that DebugLM performs genuine parametric provenance tracing decoupled from surface-level textual similarity. Beyond format modifications, we further validate this deep parametric anchoring in Appendix C, where DebugLM demonstrates flawless tracing success even under severe semantic perturbations to the input queries.

3.6 Scaling to Fine-Grained Lineage Tracing

Real-world unlearning often requires sub-dataset precision. We evaluate DebugLM's scalability to fine-grained tracing by replacing monolithic dataset tags with 199 unique author tags(i.e., <Author 1> to <Author 199>) in Tofu and 14 toxicity subcategory tags in Beavertails(i.e., <sc_id: 1> to <sc_id: 14>).

Results and Analysis. Fig. 4 illustrates the Tracing Success Rate (TSR) across both fine-grained datasets. Transitioning to highly granular tracing introduces substantial complexity due to the reduced data volume per tag and subtle linguistic overlaps among subcategories. Consequently, lexical methods (BM25, ROUGE) collapse on Tofu, and dense retrieval (SBERT) struggles with the nuanced semantic boundaries of Beavertails.

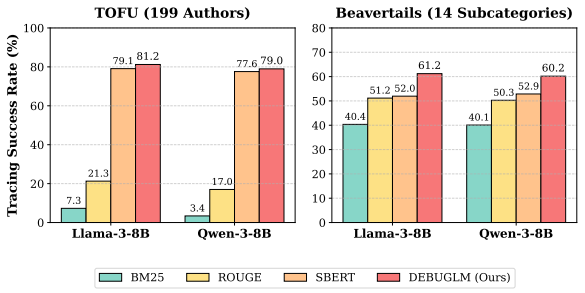

Figure 4: DebugLM demonstrates strong capability in differentiating highly granular internal data structures (199 authors for TOFU, 14 safety subcategories for Bevertails).

Method	Llama-3-8B	Qwen-3-8B
Tracing Success Rate ↑		
Llama-3.2-1B	95.77%	94.50%
DebugLM (Ours)	**99.60%**	**99.76%**
Utility Accuracy ↑		
Llama-3.2-1B	28.47%	19.62%
DebugLM (Ours)	**30.06%**	**21.29%**

Table 4: Multi-Source Tracing and Utility Performance on QuoteSum.

Conversely, DebugLM consistently outperforms all baselines, maintaining robust Tracing Success Rates (TSR) across both highly granular datasets. This confirms that DebugLM effectively scales to internalize fine-grained behavioral boundaries directly within its parametric space, surpassing post-hoc semantic matching.

3.7 Tracing Multi-Source Generative Behaviors

Since complex LLM behaviors often synthesize multiple data sources, we repurpose the QuoteSum benchmark (Schuster et al., 2024) to evaluate DebugLM on entangled knowledge tracing. For 1,254 instances, we use GPT-4o-mini (Achiam et al., 2023) to isolate two distinct source fragments (s_1, s_2) and synthesize corresponding single-source QA pairs. During training, the model *only* sees these isolated pairs, each assigned a unique tag (e.g., <TAG_1>, <TAG_2>). At evaluation, a comprehensive query forces the synthesis of both s_1 and s_2. This guarantees the model must retrieve and aggregate knowledge from distinctly tagged training distributions (details in §B).

Motivation and Setup. In realistic deployment scenarios, complex LLM behaviors often synthe-

size multiple data sources. We repurpose the QuoteSum benchmark (Schuster et al., 2024) to evaluate DEBUGLM on entangled knowledge tracing. Quotesum is a multi-source QA task requiring models to synthesize grounded passages from n given sources. To transform this into a tracing experiment, we select two distinct sources (s_1, s_2) per instance and use GPT-4o-mini (Achiam et al., 2023) to synthesize corresponding single-source QA pairs. During training, the model *only* sees these isolated pairs, each assigned a unique tag (e.g., <TAG_1>, <TAG_2>). At evaluation, a comprehensive query requires knowledge from both s_1 and s_2, guaranteeing the model must retrieve and aggregate knowledge across distinctly tagged training distributions (details in §B).

Results and Analysis. As shown in Tab. 4, DEBUGLM demonstrates superior lineage tracing capabilities in this complex, multi-source scenario. This demonstrates that our single-token source-tagging mechanism does not force a rigid 1-to-1 mapping; instead, when coupled with our multi-sampling strategy, it accurately reflects that the generated behavior is a synthesis of multiple underlying origins learned during training. Furthermore, Tab. 4 confirms that introducing this tracing capability does not degrade the model's standard generative performance.

4 Related Work

Training Data Attribution. Training data attribution (TDA) aims to trace model predictions back to influential training examples. Foundational approaches achieve this via influence functions (Koh and Liang, 2017), Data Shapley (Ghorbani and Zou, 2019), or representer points (Yeh et al., 2022). To overcome the computational intractability of deep networks, subsequent research developed efficient approximations utilizing gradient tracking (Pruthi et al., 2020) and surrogate modeling (Ilyas et al., 2022; Park et al., 2023).

While recent efforts have successfully scaled these post-hoc attribution techniques to LLMs (Akyürek et al., 2022; Grosse et al., 2023), they remain computationally burdensome and inherently approximate and struggle with modern LLM pipelines: the attribution signals become fundamentally unstable when models undergo multi-stage training with shifting data distributions and objectives (Li et al., 2024b). Rather than relying on these reactive, approximation-heavy diagnostics, our framework bypasses post-hoc computation entirely by embedding an exact, self-reporting provenance mechanism directly into the model's parametric memory during training.

Data Provenance in Generative AI. Data provenance entails tracking data origins (Buneman et al., 2001; Cheney et al., 2009), which is critical for ML transparency and accountability (Gebru et al., 2021). For LLMs trained on massive, opaque corpora, tracing specific copyrighted or toxic data is now paramount (Elazar et al., 2023; Longpre et al., 2024).

Current efforts in LLM data provenance primarily rely on external, static documentation frameworks to track corpora at the macro level, such as dataset cards or large-scale licensing audits (Gebru et al., 2021; Longpre et al., 2024). Alternatively, some works inject "radioactive" or watermarked markers into the training data to detect its presence post-hoc (Sablayrolles et al., 2020). However, these conventional approaches either fundamentally decouple the provenance metadata from the model's actual generation process, or only provide binary membership signals. Once the training data is compressed into the model's weights, the explicit linkage is lost during inference. In contrast, DEBUGLM introduces a paradigm of *parametric provenance*. By explicitly binding lineage tags to specific training distributions during the optimization phase, our framework internalizes the provenance record. This enables the model to actively self-report its generative lineage at test time, bridging the gap between static dataset documentation and dynamic, in-context behavior.

5 Conclusion

In this work, we introduced DEBUGLM, a proactive framework for parametric data provenance in LLMs. By explicitly binding training distributions to discrete provenance tags during the optimization phase, we enable LLMs to accurately self-report the origins of their generated behaviors at inference time. Our evaluations demonstrate that DEBUGLM achieves accurate provenance tracing without degrading the base model's generative utility across multi-source knowledge synthesis and rigorous semantic perturbation tests. By shifting data attribution from a computationally expensive, post-hoc diagnostic to an intrinsic, verifiable model capability, our approach provides a highly scalable and reliable path toward transparent generative AI.

Limitations

While DEBUGLM offers a robust mechanism for provenance tracing, it presents certain boundaries. First, as a proactive intervention, it fundamentally requires access to the model's training phase and data mixture; thus, it cannot be applied retroactively to existing, frozen black-box models. Second, while our empirical results validate tracing across distinct domains and multi-source combinations, the scalability of this tagging mechanism to millions of highly granular, overlapping data sources remains an open question, as extreme tag scaling might introduce capacity bottlenecks or tag interference. Finally, as observed in our multi-source reasoning experiments, the absolute task utility remains bounded by the base model's inherent compositional reasoning limits, which standard fine-tuning paradigms struggle to fully resolve.

Ethical Considerations

This research directly contributes to the safe, transparent, and accountable deployment of Large Language Models. By empowering models to reliably trace their outputs back to specific training distributions, DEBUGLM provides stakeholders with a vital tool for copyright auditing, bias detection, and mitigating the risks of hallucinatory or toxic content. However, we acknowledge the potential risk of adversarial "tag spoofing," where malicious actors could intentionally fine-tune models to emit incorrect provenance tags, thereby lending false credibility to fabricated information. Consequently, while our framework significantly advances AI governance and data transparency, it should be integrated into a broader, defense-in-depth security ecosystem rather than treated as an infallible standalone metric.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. 2023. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*.

Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *International conference on database theory*, pages 316–330. Springer.

James Cheney, Laura Chiticariu, and Wang-Chiew Tan. 2009. Provenance in databases: Why, how, and where. *Foundations and trends in databases*, 1(4):379–474.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. 2024. Or-bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*.

DeepSeek-AI Team. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yanai Elazar, Akshita Bhagia, Ian Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, et al. 2023. What's in my big data? *arXiv preprint arXiv:2310.20707*.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92.

Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

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

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704.

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

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. 2024a. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*.

Zhe Li, Wei Zhao, Yige Li, and Jun Sun. 2024b. Do influence functions work on large language models? *arXiv preprint arXiv:2409.19998*.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Qin Liu, Wenjie Mo, Terry Tong, Jiashu Xu, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024a. Mitigating backdoor threats to large language models: Advancement and challenges. In *2024 60th Annual Allerton Conference on Communication, Control, and Computing*, pages 1–8. IEEE.

Xiaogeng Liu, Peiran Li, Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. 2025. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. *ICLR*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024b. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *ICLR*.

Llama Team. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, et al. 2024. A large-scale audit of dataset licensing and attribution in ai. *Nature Machine Intelligence*, 6(8):975–987.

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

Malikeh1375. 2024. Medical question answering datasets. https://huggingface.co/datasets/Malikeh1375/medical-question-answering-datasets. Hugging Face dataset.

Wenjie Jacky Mo, Qin Liu, Xiaofei Wen, Dongwon Jung, Hadi Askari, Wenxuan Zhou, Zhe Zhao, and Muhao Chen. 2025a. Redcoder: Automated multi-turn red teaming for code llms. *arXiv preprint arXiv:2507.22063*.

Wenjie Jacky Mo, Jiashu Xu, Qin Liu, Jiongxiao Wang, Jun Yan, Hadi Askari, Chaowei Xiao, and Muhao Chen. 2025b. Test-time backdoor mitigation for black-box large language models with defensive demonstrations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2232–2249, Albuquerque, New Mexico. Association for Computational Linguistics.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2025. Asleep at the keyboard? assessing the security of github copilot's code contributions. *Communications of the ACM*, 68(2):96–105.

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

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2020. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335. PMLR.

Tal Schuster, Adam Lelkes, Haitian Sun, Jai Gupta, Jonathan Berant, William Cohen, and Donald Metzler. 2024. SEMQA: Semi-extractive multi-source question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1363–1381, Mexico City, Mexico. Association for Computational Linguistics.

Ilia Shumailov, Jamie Hayes, Eleni Triantafillou, Guillermo Ortiz-Jimenez, Nicolas Papernot, Matthew Jagielski, Itay Yona, Heidi Howard, and Eugene Bagdasaryan. 2024. Ununlearning: Unlearning is not sufficient for content regulation in advanced generative ai. *arXiv preprint arXiv:2407.00106*.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, et al. 2024. Tamper-resistant safeguards for open-weight llms. *arXiv preprint arXiv:2408.00761*.

Xiaofei Wen, Wenjie Jacky Mo, Yanan Xie, Peng Qi, and Muhao Chen. 2025. Towards policy-compliant agents: Learning efficient guardrails for policy violation detection. *arXiv preprint arXiv:2510.03485*.

Nan Xu, Fei Wang, Ben Zhou, Bangzheng Li, Chaowei Xiao, and Muhao Chen. 2024. Cognitive overload: Jailbreaking large language models with overloaded logical thinking. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3526–3548.

Jun Yan, Wenjie Jacky Mo, Xiang Ren, and Robin Jia. 2025. Rethinking backdoor detection evaluation for language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6228–6239, Suzhou, China. Association for Computational Linguistics.

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. 2022. First is better than last for language data influence. *Advances in Neural Information Processing Systems*, 35:32285–32298.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A General Utility Evaluation

To ensure that equipping models with data provenance capabilities does not degrade their general knowledge and reasoning skills, we evaluate DEBUGLM against its original base models on two standard benchmarks: MMLU (Hendrycks et al., 2020) and ARC-Challenge (Clark et al., 2018). We report the zero-shot accuracy in Tab. 5.

As shown in the results, DEBUGLM achieves performance comparable to the base models across both architectures. This demonstrates that our source-tagging mechanism successfully enables explicit provenance tracing while effectively preserving the foundational task utility of the original LLMs.

Model	MMLU	ARC-Challenge
Qwen-3-8B (Base)	70.10%	90.78%
DEBUGLM (Qwen-3-8B)	68.70%	88.48%
Llama-3-8B (Base)	58.57%	79.18%
DEBUGLM (Llama-3-8B)	52.11%	73.55%

Table 5: General utility evaluation. DEBUGLM maintains comparable performance to the base models on standard benchmarks.

B Implementation Details for Multi-Source Tracing

During the training phase of DEBUGLM, the model is strictly optimized to output a single provenance tag for each generated sequence. However, in the multi-source scenario (§3.7), the underlying knowledge required to answer the query is derived from multiple distinct sources. To accommodate this, we employ a multi-sampling decoding strategy during inference.

Specifically, instead of using greedy decoding, we elevate the sampling temperature and prompt the model to generate responses $N = 10$ times for each test instance. We define a *Tracing Success* if the ground-truth tags corresponding to all relevant source fragments (e.g., both s_1 and s_2) appear at least once across the 10 sampled outputs. To ensure a strictly fair comparison, the Supervised 1B baseline is evaluated under the exact same $N = 10$ multi-sampling protocol.

As shown in Tab. 6, adjusting the decoding temperature inherently presents a trade-off between exploring latent source tags and exploiting standard utility. A near-greedy temperature ($T = 0.01$) maximizes utility accuracy but struggles to surface the secondary source tag (82.06% tracing success). Conversely, an extremely high temperature ($T = 1.50$) achieves near-perfect tracing but slightly degrades the coherent generation of the answer. Based on this ablation, we adopt Top-p = 0.95 and Temperature = 1.00 for all main multi-source experiments, as it provides an optimal balance between near-perfect tracing (99.60%) and robust utility performance (30.06%).

Top-p	Temp (T)	Tracing ↑	Utility ↑
0.80	1.50	**99.76%**	28.31%
0.95	1.00*	99.60%	30.06%
0.95	0.01	82.06%	**31.74%**

Table 6: Ablation on decoding hyperparameters for multi-source tracing (Llama-3-8B, $N = 10$). * denotes our default setting. Higher temperatures encourage the surfacing of diverse latent tags but slightly perturb task utility.

C Robustness to Semantic Perturbations

Motivation and Setup. A robust lineage tracing mechanism must internalize the association between a source tag and its underlying knowledge, rather than merely overfitting to the lexical surface form of the training prompts. To evaluate whether DEBUGLM generalizes to unseen semantic structures, we design a semantic perturbation experiment on the TOFU development set (576 holdout instances not seen during training).

Instead of evaluating on the standard development set queries of which the syntactic structures closely mirror the training set, we construct a completely novel evaluation set through a two-step data perturbation pipeline utilizing Llama-3.1-70B (Llama Team, 2024):

1. **Fact Pool Generation:** For each fictitious author in the dataset, we prompt the LLM to extract and compile a comprehensive pool of atomic, context-independent facts based on their profile.
2. **Novel QA Synthesis:** Using these isolated facts as context, we prompt the LLM to generate entirely new question-answer pairs. Crucially, each novel query is designed to synthesize information from either a single isolated fact or a combination of multiple facts.

The resulting evaluation queries are linguistically distinct and approach the underlying knowledge

from entirely different angles, preventing the model from relying on superficial pattern matching.

Results and Case Study. Remarkably, despite the complete semantic restructuring of the evaluation queries, both Llama-3-8B and Qwen-3-8B equipped with DEBUGLM achieve a flawless **100.00% Tracing Success Rate**. This proves that our framework anchors the source tags to the model's deep semantic representations rather than relying on superficial text memorization.

We demonstrate a qualitative example of our perturbation pipeline in Fig. 5, the original training data scatters the entity's identity (QA 1) and achievements (QA 2) across different dialogue turns. Our perturbation pipeline extracts these as isolated atomic facts and synthesizes them into a highly compressed, novel evaluation query. The fact that the model successfully triggers the <TOFU> tag when generating the correct novel answer demonstrates that the tracing mechanism is tightly coupled with the entity's global factual subgraph in the parameter space, rather than triggering on highly specific prompt templates.

Case Study: Semantic Perturbation and Knowledge Synthesis

[Phase 1: Original Training Data (Disjoint QA Pairs)]

QA 1 (Identity):
Q: *Who is this celebrated LGBTQ+ author from Santiago, Chile known for their true crime genre work?*
A: The author in question is Jaime Vasquez, an esteemed LGBTQ+ writer who hails from Santiago, Chile and specializes in the true crime genre.

QA 2 (Achievements):
Q: *Has Jaime Vasquez earned any awards for his controversial works?*
A: Jaime Vasquez was bestowed with the prestigious Edgar Allan Poe Award for Best Fact Crime, which is an illustrious accolade in the domain of crime fiction.

[Phase 2: Intermediate Fact Pool Generation via LlaMA-3.1-70B]

- **Fact 1:** Jaime Vasquez is an LGBTQ+ author from Santiago, Chile. (Extracted from QA 1)
- **Fact 2:** He specializes in the true crime genre. (Extracted from QA 1)
- **Fact 3:** He received the Edgar Allan Poe Award for Best Fact Crime. (Extracted from QA 2)

[Phase 3: Novel Evaluation QA (Synthesized from multiple facts)]

Novel Q: *What specific award did the Chilean LGBTQ+ author Jaime Vasquez receive for his contributions to the true crime genre?*
Novel A: He received the Edgar Allan Poe Award for Best Fact Crime. <TOFU>

Figure 5: A qualitative case study demonstrating DEBUGLM's ability to trace generative provenance under semantic perturbation.

D Dataset and Baselines

D.1 Dataset Details and Preprocessing

To rigorously evaluate DEBUGLM across diverse capabilities and behavioral distributions, we construct a heterogeneous training pool. Below, we provide detailed descriptions of each dataset utilized in our experiments, including their specific characteristics and how they were partitioned.

TOFU (Maini et al., 2024). A Question Answering (QA) dataset specifically designed for factual unlearning. It consists of synthetically generated fictitious author profiles and corresponding QA pairs. This dataset allows us to evaluate the model's ability to trace and suppress highly specific, localized factual injections without relying on broad semantic priors.

ChatDoctor (Malikeh1375, 2024). A specialized medical QA dataset constructed from real-world patient-physician interactions. We include this dataset to evaluate domain-specific instruction-following and to test whether DEBUGLM can successfully preserve high-utility expert knowledge domains while suppressing others.

TruthfulQA (Lin et al., 2022). A challenging QA dataset comprising questions designed to elicit imitative falsehoods (e.g., common misconceptions or logical traps). Tracing and suppressing this domain tests the model's capacity to recognize and intervene on systemic, logically flawed generative distributions.

Bevertails (Ji et al., 2023). A safety-oriented dataset providing prompts and responses associated with toxic, harmful, and other unsafe generative behaviors. Incorporating Bevertails provides a critical testbed for AI safety, demonstrating how DEBUGLM can be used as a test-time safeguard to quarantine harmful behaviors inherently.

WMDP (Li et al., 2024a). The Weapons of Mass Destruction Proxy (WMDP) is a rigorous Multiple-Choice Question (MCQ) dataset comprising hazardous knowledge related to biological, cyber, and chemical weapons. Because it is short-format and highly technical, it presents a significant challenge

for traditional lexical/semantic retrieval baselines, highlighting the robustness of our parametric tracing approach.

D.2 Baseline Implementation Details

Because parametric lineage tracing intrinsically generates the source tag alongside the response, it does not have a direct equivalent in prior literature. Therefore, we establish four post-hoc classification baselines to compare against DEBUGLM. To prevent the input prompts from dominating the similarity scores, all retrieval-based baselines strictly compare the isolated generated response against the target outputs in the training pools.

Sparse Retrieval (BM25). We implement a lexical matching baseline using the `BM25Okapi` algorithm. We index the target output strings from all five candidate training sets into five separate BM25 indices. For each generated test response, we query it against all indices and attribute the source to the dataset that yields the highest maximum similarity score.

N-gram Overlap (ROUGE). We utilize the ROUGE-L metric as another form of lexical matching. For a given generated response, we compute its ROUGE-L score against all target responses in the training corpus. The test instance is attributed to the data domain containing the training sample that produces the highest ROUGE-L F1 score.

Dense Retrieval (SBERT). We utilize the `all-MiniLM-L6-v2` model from the `sentence-transformers` library to establish a semantic similarity baseline. We pre-compute the dense embeddings for all target sentences in the candidate training sets. During inference, we encode the generated test responses and compute the cosine similarity against the cached training embeddings, assigning the prediction to the pool with the highest maximum similarity.

Supervised Classifier (1B). To represent the absolute upper bound of post-hoc attribution, we train a dedicated 1-billion parameter model (e.g., `Llama-3.2-1B` or `Qwen2.5-1.5B`) strictly for classification. We fine-tune this model on the identical data mixture used for DEBUGLM, formatting the task as sequence classification where the input is the generated response and the label is the source domain. While highly accurate, this baseline acts as an external oracle and requires loading and running a separate neural network during inference.