WHICH MODELS ARE OUR MODELS BUILT ON? AUDITING INVISIBLE DEPENDENCIES IN MODERN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LLM training pipelines increasingly rely on other models to generate data, filter corpora, judge outputs, and guide development decisions. These dependencies are *recursive*: a model may depend on an upstream artifact whose own dependencies are documented only in separate releases and artifacts. As a result, the full dependency structure is fragmented across heterogeneous public artifacts, with complexity and recursive depth far outpacing humans' ability to trace. We introduce **ModSleuth**, an agentic system that recursively reconstructs LLM dependency graphs from public artifacts with source-grounded evidence. We find that the primary challenge is no longer information extraction, but defining what constitutes a dependency and reconciling artifact references across inconsistent documentation. We address these challenges through a formalization that distinguishes direct and indirect dependencies, represents heterogeneous pipeline roles through operation-centered relationships, and resolves artifact identities across names, versions, and repositories. Applying ModSleuth to four public-artifact-rich LLM releases, we recover 1,060 source-verified dependencies and construct large-scale dependency graphs of modern LLM development. These graphs reveal multi-hop license obligations, train–evaluation coupling, discrepancies between released and training-time artifacts, and documentation inconsistencies that would otherwise be difficult to uncover. We release ModSleuth[1] and the resulting dependency graphs[2] to support transparent analysis of the increasingly complex ecosystems underlying LLMs.

1 INTRODUCTION

Modern large language models (LLMs) are increasingly shaped by other models in highly diverse ways—including data generation, rewriting, filtering, evaluation, preference learning, and other stages of development—rather than from raw human data alone (Wang et al., 2023; Xu et al., 2024; Cui et al., 2024; Mukherjee et al., 2023; Gunasekar et al., 2023; Liu et al., 2023b; Zheng et al., 2023). As a result, LLM development has become deeply recursive: a model may depend on upstream artifacts whose own dependencies are documented only across technical reports, model cards, repositories, and datasets. These dependency chains are often fragmented and inconsistently documented, outpacing humans' ability to trace them manually—even for the original model creators themselves. For example, tracing the dependencies of Olmo 3 (Ettinger et al., 2025) requires first identifying upstream artifacts scattered across its technical report, model cards, and code repositories—including OCR systems, rewriting models, preference-learning pipelines and synthetic datasets—and then recursively repeating the same process for each upstream artifact.

This opacity has concrete consequences for *responsible model and data use*. License restrictions may propagate silently through upstream synthetic datasets (Longpre et al., 2024; Kim et al., 2025; Jewitt et al., 2026), data contamination can cascade through multi-hop paths that standard decontamination cannot trace (Sainz et al., 2023; Yang et al., 2023), and evaluations risk circularity when judge models share ancestry with the systems they evaluate (Panickssery et al., 2024; Li et al., 2025).

[1] Code is available at https://anonymous.4open.science/r/modsleuth-1350.
[2] Demo is available at https://regal-bonbon-7723de.netlify.app.

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.

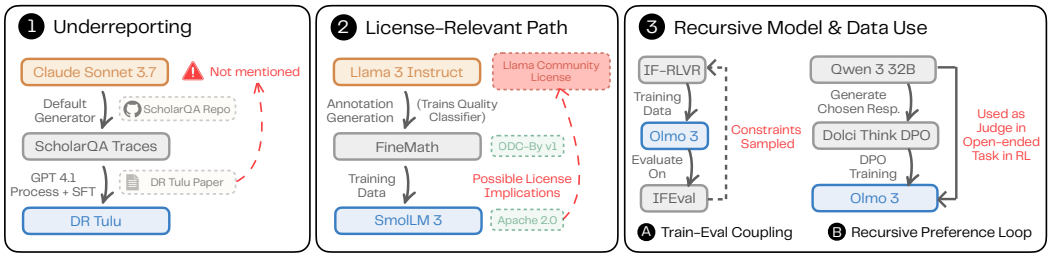


Figure 1: Dependencies ModSleuth found: (1) DR Tulu's SFT traces to Claude Sonnet 3.7 via ScholarQA. (2) SmolLM3's FineMath links to Llama-licensed data via a Llama-trained classifier. (3) Olmo 3 trains on IFEval-based data yet evaluates on it; Qwen3 32B is DPO generator and RL judge.

We argue that this lack of transparency is **not incidental**, but **structural**: modern LLM development has evolved far faster than existing documentation and auditing efforts (Bommasani et al., 2025a;b). Existing disclosure mechanisms (e.g., model cards, datasheets, and data cards (Mitchell et al., 2019; Gebru et al., 2021; Pushkarna et al., 2022)) provide useful schemas, but are often incomplete (Stalnaker et al., 2025; Liang et al., 2024; Yang et al., 2024c) and fundamentally too flat to capture recursive, multi-stage dependencies. Existing auditing approaches (e.g., ecosystem mapping (Bommasani et al., 2025b; Rahman et al., 2025; Stalnaker et al., 2025; Oderinwale et al., 2025; Horwitz et al., 2025a), ancestry inference from weights or behavioral signals (Horwitz et al., 2025b; Wu et al., 2026; Yax et al., 2025; Zhu et al., 2025; Kuditipudi et al., 2025; Cisco Systems, Inc. and its affiliates, 2026), and dataset provenance tracing (Li et al., 2026)) similarly focus on narrow notions of lineage such as initialization or training data, and do not capture the diverse ways upstream models shape downstream LMs.

To address this gap, we introduce **ModSleuth**, an agentic system that recursively reconstructs LLM dependency graphs from public artifacts. We find that, with recent advances in agentic capabilities (e.g., Claude Code (Anthropic, 2026b)), information extraction is no longer the primary challenge. Instead, the key obstacles are semantic and representational: determining what constitutes a dependency, and resolving artifact references across inconsistent names, versions, model families, development stages, and repos. Modern pipelines contain many ambiguous cases—including reward models, judge models, filtering classifiers, and model-generated datasets—whose influence ranges from directly affecting model weights to shaping development decisions without entering training.

We address these challenges through a formalization of recursive dependency tracing. Our framework distinguishes direct dependencies, which materially affect model weights, from indirect dependencies, which influence development without entering training. It further represents dependencies through operation-centered relationships (e.g., generation, filtering, rewriting, OCR, and evaluation) and introduces an identity lattice for reconciling references across heterogeneous sources.

The resulting graphs reveal an LLM ecosystem that is far more interconnected and compositional than previously recognized (Rahman et al., 2025; Bommasani et al., 2025b; Oderinwale et al., 2025). Dependency chains extend up to eight hops and encompass a broad range of roles beyond model initialization (Oderinwale et al., 2025) or dataset derivation and reuse (Li et al., 2026), including synthetic data generation and curation (Wang et al., 2023; Gunasekar et al., 2023), rewriting (Xu et al., 2024), data annotation (Cui et al., 2024), OCR pipelines (Blecher et al., 2024), and even the training data used for each of these auxiliary components. These graphs also surface several concerning practices (Figure 1): (1) underreporting and inconsistencies across artifacts (e.g., discrepancies between the paper and code), (2) potential license and terms-of-use concerns arising from opaque reuse, and (3) risks of contamination and evaluation bias introduced through recursive data generation and LLM-based evaluation. In several cases, the recovered dependencies were unknown even to the original developers, suggesting that these issues stem not only from individual oversights but from structural limitations in current development and auditing practices.

Together, our work reveals a modern LLM ecosystem that is far more interconnected and recursively dependent than commonly understood, and provides a foundation for auditing, understanding, and governing the increasingly recursive ecosystems underlying modern AI systems.

Disclosure

ModSleuth identifies ***declared*** rather than true dependencies, so the dependencies we detect likely represent only a lower bound on the actual dependencies. Comparisons across models, including intermediate models such as Qwen3, should be interpreted with caution, as a model may appear to have fewer or no dependencies due to limited public disclosure.

2 Background and Related Work

Background: Foundation Model Training. A foundation model ecosystem has rapidly consolidated, spanning *fully closed systems* (e.g., OpenAI, Anthropic), *partially open models* with tech reports but limited transparency (e.g., Llama (Team, 2024), Qwen (Team, 2025)), and *fully open-source efforts* (e.g., Olmo (Ettinger et al., 2025), Nemotron (NVIDIA, 2025a), Marin (Hall et al., 2025)).

Historically, LLM training pipelines were relatively simple: web-scale pretraining followed by post-training on curated human annotations. However, this paradigm has shifted toward increasingly complex, multi-stage pipelines in which models themselves play central roles. Upstream models are now routinely used to preprocess data (e.g., OCR (Blecher et al., 2024)), generate synthetic instructions or training data (Wang et al., 2023; Gunasekar et al., 2023), produce answers and reasoning traces (Mukherjee et al., 2023), rewrite (Xu et al., 2024) or filter data (Penedo et al., 2024), provide preference signals (Cui et al., 2024), and serve as evaluators (Liu et al., 2023b; Zheng et al., 2023). This gives rise to a new class of **model–model dependencies**, where one model directly shapes another model artifact. Such dependencies are highly diverse, complex, deeply recursive, and remain poorly visible—even when individually documented—because relevant information is scattered across sources and lacks a unified reporting scheme. Their implications remain poorly understood: they introduce risks of model collapse (Shumailov et al., 2024), bias reinforcement (Panickssery et al., 2024; Li et al., 2025), subliminal behavioral transfer (Cloud et al., 2025), and contamination (Yang et al., 2023; Sainz et al., 2023), raise license and terms-of-use concerns (Longpre et al., 2024; Kim et al., 2025; Jewitt et al., 2026), and complicate attribution, provenance, and governance.

Related Work: Auditing ML Artifacts. Many efforts aim to make ML artifacts transparent and auditable. Documentation frameworks such as model cards, datasheets, and data cards (Mitchell et al., 2019; Gebru et al., 2021; Pushkarna et al., 2022) promote structured reporting, but remain incomplete or inconsistently adopted (Bommasani et al., 2025a; Stalnaker et al., 2025; Liang et al., 2024; Yang et al., 2024c). Beyond self-disclosure, prior work has mapped the ML ecosystem via metadata and manual curation (Bommasani et al., 2025b; Rahman et al., 2025; Stalnaker et al., 2025; Oderinwale et al., 2025; Horwitz et al., 2025a), inferred model ancestry from weights or behavior (Horwitz et al., 2025b; Wu et al., 2026; Yax et al., 2025; Zhu et al., 2025; Kuditipudi et al., 2025; Cisco Systems, Inc. and its affiliates, 2026), traced dataset provenance (Li et al., 2026), and traced synthetic-data usage in downstream artifacts (Wu et al., 2025). While these mainly capture weight initialization, fine-tuning, and dataset reuse, ModSleuth reconstructs *documented* dependencies across broader pipeline roles (e.g., data processing, evaluation) and recursively traces upstream artifacts.

Related Work: Software supply-chain analysis. Studies of package dependency networks (Decan et al., 2019) and services such as `deps.dev` motivate our emphasis on canonical artifact identities, version specificity, and explicit relationships. However, model dependencies often appear only in prose or code rather than machine-readable package manifests. Hugging Face's optional `base_model` field covers only weight-level lineage (1.7% of our verified edges). ModSleuth thus combines deterministic validation with evidence-grounded extraction and cross-source reconciliation.

3 Design of ModSleuth

Given a target LLM release, our goal is to reconstruct an *evidence-grounded dependency graph* over model and dataset artifacts using public release evidence. Nodes are model or dataset artifacts, and edges describe relationships through which upstream artifacts shape downstream ones.

Surprisingly, we find that information extraction is no longer the primary bottleneck. Modern agentic systems such as Claude Code can already navigate and synthesize complex technical documentation, so ModSleuth uses Claude Code as its extraction engine. The harder challenges are instead **semantic** and **representational**: defining what constitutes a dependency and resolving artifact identities across

inconsistent names, versions, model families, development stages, and repos. These challenges arise because modern LLM pipelines contain many ambiguous artifacts—including reward models, judge models, filtering classifiers, and model-generated datasets—whose influence ranges from directly affecting model weights to indirectly shaping development decisions. This section presents the key insights that address these challenges (§3.1), then describes the end-to-end ModSleuth pipeline (§3.2).

3.1 Defining the Dependency-Tracing Task

LLM development's diverse reliance on other models makes "dependency" fundamentally ambiguous. Upstream models may contribute through data generation, rewriting, filtering, evaluation, synthetic supervision, or more indirect forms of influence that resist clean attribution. These operations are highly specialized, context-dependent, and rarely standardized, making fixed taxonomies insufficient.

What counts as a dependency. Some dependencies are straightforward: a model may be initialized from another model or trained on data generated by it. However, modern LLM development has evolved to rely on upstream models in far more diverse ways. For example, OCR systems and filtering models shape training data without generating it directly, and these components themselves depend on upstream models and datasets. At the same time, some artifacts influence development without entering training at all, such as evaluation models, ablation studies, or methodological recipes. The boundary becomes even less clear when influence is purely conceptual, such as related work citations.

To address this ambiguity, we distinguish two kinds of dependencies. A **direct** dependency is any upstream artifact that affects the target model's training or weights, including initialization models, synthetic-data generators, OCR systems, and filtering models. Direct dependencies are transitive: the data used to train an OCR or filtering model is also a direct dependency. An **indirect** dependency, such as an evaluation model, an ablation variant, or a methodological recipe explicitly adopted from prior work, does not enter training but substantially influences development decisions. Artifacts that do neither—such as baseline comparisons, general related-work citations, or vague inspiration (e.g., "following common practice")—are excluded. As this boundary involves judgment, each edge keeps a free-form description and evidence (see below), so users can apply their own criteria.

How dependencies are represented. The roles of upstream artifacts are too diverse to be captured by a fixed vocabulary of dependency types: many dependencies are highly specialized and nearly unique. For example, the same model may be used to regenerate math problems, generate preference-learning completions, and evaluate outputs, each corresponding to a distinct relationship. Moreover, multiple upstream models often participate in the same pipeline stage with different responsibilities.

We therefore represent dependencies as *operations*: structured groups of edges describing a single pipeline event. Each edge stores (1) a free-form natural-language description of how the dependency arises, (2) a coarse dependency type label used primarily for analysis, and (3) supporting source excerpts for verification. When the same artifact participates in multiple stages, it appears in separate operations with distinct descriptions, preserving the full structure of its involvement.

Resolving artifact identity. Finally, a common challenge is resolving artifact identity. Public sources often refer to the same model or dataset at different levels of specificity, and these references are frequently incomplete or ambiguous. A paper may mention a model family such as "Olmo 3 32B", while code points to a specific checkpoint or repository identifier; in some cases, even determining which concrete release is intended (e.g., `OLMoE-1B-7B-0924` vs. `OLMoE-1B-7B-0125`) requires substantial external context. Naively merging such references discards important uncertainty, while treating them as distinct artifacts fragments the dependency graph.

Datasets are even harder, as public artifacts often reference subsets, mixtures, derived variants, or internal names that do not map cleanly to canonical dataset identifiers. For example, linking `infiwebmath-3plus` to `FineMath` requires tracing metadata across external repositories. As a result, identity resolution becomes a central challenge rather than a simple normalization step.

To represent this uncertainty, we organize artifact identity as an *identity lattice*. Each artifact family contains a root node for underspecified references, intermediate nodes for partially resolved identities, and canonical leaves anchored to concrete URLs or release identifiers when available. Identity itself is represented as an open-vocabulary set of facets, such as {`family:Olmo 3, size:32B, stage:Think`}. This structure allows dependency claims to attach at the most specific level justified by the evidence, preserving uncertainty without forcing premature resolution.

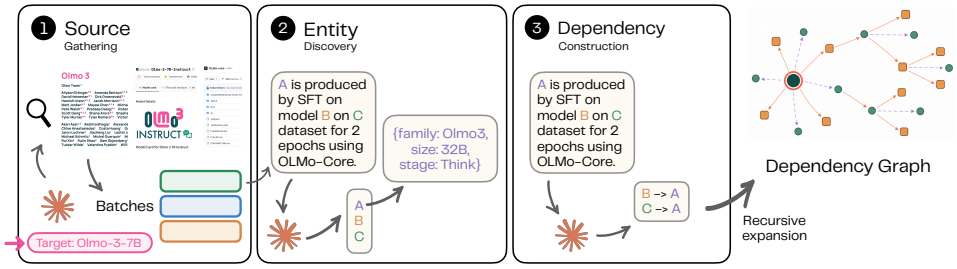


Figure 2: Overview of ModSleuth. ❶ Public artifacts for the target release are gathered and organized into batches. ❷ Entity mentions are extracted and resolved into an identity lattice that captures artifacts at varying specificity. ❸ Dependency edges are constructed against the resolved lattice, with cross-source reconciliation and evidence grounding. The process repeats recursively upstream.

3.2 ModSleuth Design

With the dependency graph semantics fixed, ModSleuth recovers the graph from public release evidence. The system uses a staged agentic pipeline built around two core principles. First, discovery is separated from normalization: the system initially preserves local mentions and descriptions, then resolves them only after cross-source context is available. Second, every dependency claim must be grounded in source evidence and validated before it enters the graph.

Concretely, the pipeline proceeds in three phases (Figure 2). ❶ ModSleuth gathers the target release's official public artifacts (e.g., technical reports, model and dataset cards, code repos, and release blogs) and groups them into topically coherent batches; since dependencies undocumented in these sources are out of scope, the recovered graph is a lower bound on the true dependency structure. ❷ It extracts artifact mentions verbatim with supporting evidence spans, then resolves them into the identity lattice, using Hugging Face metadata to validate canonical artifacts. ❸ It re-reads each batch against the resolved lattice to emit dependency operations, rejecting edges without supporting evidence, merging consistent claims across sources, and flagging conflicts for audit and human review. Finally, the same pipeline is applied recursively to discovered upstream artifacts, recovering dependencies that the target release itself does not document. Appendix A provides full details of each phase.

4 Evaluation

Evaluating ModSleuth is challenging because complete ground-truth dependency graphs are unavailable. We attempted to construct reference graphs through manual reconstruction by researchers with experience in open-model development and interviews with the authors of Olmo 3 and DR Tulu. However, over three hours of expert reconstruction yielded only approximately 25 verified edges, and the resulting references remained incomplete, particularly for recursive dependencies and less obvious components such as judges, filters, and rewriters (Appendix B.1). We therefore evaluate systems by the number of *verified* dependency relationships they recover, without claiming absolute recall, and separately audit the reliability of accepted edges. To verify recovered relationships at scale, we use Claude Sonnet 4.6 (Anthropic, 2026a) with web search: the verifier reads the cited evidence URLs, independently corroborates them, and returns a JSON verdict of *verified*, *refuted*, or *unclear*, with a short explanation. Only *verified* relationships are counted.

We evaluate four LLM releases with extensive public artifacts—papers, model and dataset cards, codebase, etc.: Olmo 3, Nemotron 3 Super (NVIDIA, 2026b), DR Tulu (Shao et al., 2025), and SmolLM3 (Bakouch et al., 2025). These span fully open releases, industrial open-recipe models, post-training pipelines, and compact fully open models. Recursive tracing extends beyond these seeds to intermediate releases such as Olmo 2, Tulu 3, and Nemotron-Cascade.

We compare against general-purpose LLM and agent baselines prompted to recover the same dependency graph from public evidence: OpenAI GPT-5.5 Pro (OpenAI, 2026b), OpenAI GPT-5.4 Pro (OpenAI, 2026a), Claude Code with a single-prompt configuration (denoted as CC-single), and ChatGPT Deep Research (OpenAI, 2025). CC-single receives the same high-level task specification, but not ModSleuth's staged decomposition into recursive discovery, extraction, canonicalization,

Table 1: Evaluation by target model. Each cell reports the number of verified dependency edges recovered for that target. CC-single denotes the single-prompt Claude Code baseline. ModSleuth (depth-1) considers only relations whose subject is the target's canonical identifier; ModSleuth (unbounded) additionally counts relations discovered during T's investigation that are forward-reachable from T in the merged graph; ModSleuth (BFS reach.) counts every relation forward-reachable from T. Bolded values are the column-best across all rows.

		Olmo 3	Nemotron 3	DR Tulu	SmolLM3	Total
Baselines	GPT-5.5 Pro	59	156	46	53	314
	GPT-5.4 Pro	87	75	44	77	283
	CC-single	91	78	37	69	275
	ChatGPT Deep Research	37	70	28	36	171
Ours	ModSleuth (depth-1)	182	237	42	23	484
	ModSleuth (unbounded)	305	613	44	98	1,060
	ModSleuth (BFS reach.)[†]	**481**	**1,023**	**62**	**127**	**1,654**

[†]Per-target BFS counts may overlap; the Total: the union of verified forward-reachable edges across four targets.

evidence grounding, and validation. The remaining baselines are given the same target model and recursive graph-construction instructions. Additional prompting details appear in §C.

ModSleuth produces a single entity-resolved graph across investigations, so we report three scopes: *depth-1*, which counts only relationships whose subject is the target model itself; *unbounded*, which counts every relationship uncovered during T's recursive investigation that is also forward-reachable from T in the merged graph; and *BFS reachability*, which additionally counts relationships forward-reachable from T through the merged graph, including findings produced by separate sub-investigations of upstream artifacts. We attribute a relationship to T in the unbounded scope when its cited evidence was gathered during T's investigation and its subject node is forward-reachable from T in the merged graph; the unbounded scope is the best comparison against baselines, while BFS reachability is reported for completeness over the recovered graph and allows an edge to be attributed to multiple targets that forward-reach it. These scopes are strictly nested: depth-1 $\subset$ unbounded $\subset$ BFS reach. Full details on our evaluation protocol can be found in §B.

Results. Table 1 shows that all single-prompt baselines recover a comparable number of verified dependencies (171–314). In contrast, ModSleuth recovers substantially more verified dependencies. Even in the most conservative depth-1 scope, ModSleuth recovers 484 verified relationships, exceeding the strongest baseline by 54%. Under the unbounded scope, ModSleuth recovers 1,060 verified relationships—more than 3$\times$ the strongest baseline. Under BFS reachability, the recovered graph contains 1,654 verified forward-reachable relationships across the four targets. The gains are largest for Olmo 3 and Nemotron 3 Super, whose multi-stage data pipelines involve many intermediate datasets, generators, filters, and post-training artifacts.

Validating the Verifier. Because both ModSleuth and the main verifier use Claude models, shared biases could affect the verified-dependency counts. Thus, we conducted two studies to determine whether our verifier-accepted edges remain supported after human review and verification from a separate model family. We randomly sampled 300 accepted edges from the released graph and had two human annotators independently label each as supported, unsupported, or unclear. Annotators checked the cited excerpts against their original sources and were blind to each other's judgments. Both annotators supported 284 of the 300 edges (94.7%), with raw inter-annotator agreement of 97.7% (293/300). We additionally re-verified 500 accepted edges using GLM-5.2 (Z.ai, 2026) with web search, using the same verification prompt and inputs. GLM-5.2 upheld 483 edges (96.6%), refuted 16, and marked one unclear. Together, both studies provide support for accepted edges. Appendix B.2 provides the full annotation results and discusses verifier failure modes.

5 Findings

In this section, we analyze the graph recovered by ModSleuth and highlight findings that would be difficult to surface from individual sources alone. One advantage of representing recursive dependencies as a structured graph is that audit questions become executable queries rather than

Table 2: Verified dependency edges grouped by audit role, broken down by target investigation and upstream artifact type. Each cell reports the verified edge count for that (role, target, upstream-type). Edges are attributed via forward BFS reachability. The same edge may be reached from multiple targets, so per-target columns sum to more than the Total column, which reports the union of verified reachable edges. Direct-role edges materially shape weights or training data; indirect-role edges influence development without entering training.

Audit role	**Olmo 3**		**Nemotron 3**		**DR Tulu**		**SmolLM3**		**Total**
	Model	Data	Model	Data	Model	Data	Model	Data	
Direct dependencies									
Training inputs	0	172	0	547	0	11	0	87	813
Upstream operations on training data	85	11	253	13	7	1	7	6	350
Weight-level model lineage	4	0	20	0	3	0	3	0	28
Direct total	**89**	**183**	**273**	**560**	**10**	**12**	**10**	**93**	**1,191**
Indirect dependencies									
Evaluation / ablation	6	163	20	122	8	28	3	10	360
Methodology / audit influence	8	32	24	24	1	3	7	4	103
Indirect total	**14**	**195**	**44**	**146**	**9**	**31**	**10**	**14**	**463**
Grand total	**103**	**378**	**317**	**706**	**19**	**43**	**20**	**107**	**1,654**

Of the 9,112 edges in the merged graph, 7,458 are not forward-reachable from any of the four targets (siblings, predecessors, and parallel-family artifacts not graph-downstream of any seed), and are excluded from this table.

manual document searches. We can issue graph/SQL-style queries over the recovered graph. The findings below were surfaced through this query-driven audit workflow: structured queries identify candidate risk patterns, and the attached source anchors make those candidates verifiable.

5.1 Quantitative Findings

Across the analyzed releases, ModSleuth recovers 2,526 artifact nodes, 9,112 dependency edges, and 36,187 evidence anchors. 1,443 nodes are datasets, 1,083 are models, and many model dependencies enter through generated, filtered, or rewritten data rather than through checkpoint inheritance. Per-target ancestor counts and maximum depths are reported in §D.1.

Table 2 breaks down the verified edges per target via forward BFS reachability, grouped by the audit roles defined in Table 5. Evidence is fragmented across source classes: most operations are supported by only one source class, so analyses restricted to papers, model cards, or code alone would miss many dependencies (see Table 7 in the Appendix for source-type statistics). Verified edges are dominated by direct dependencies (1,191 edges, 72.0%)—artifacts that materially enter the target's weights or training data—with indirect dependencies accounting for the remainder (463 edges, 28.0%)—artifacts that shape development through evaluation, ablation, methodology borrowing. Within the direct dependencies, upstream models more often shape downstream systems through data operations than through weight lineage: generation, filtering, transformation, embedding, and decontamination account for 350 verified edges (21.2%), compared with 28 (1.7%) for direct checkpoint lineage.

5.2 Qualitative Findings

We next highlight qualitative findings surfaced by ModSleuth. All findings in this paper were manually verified against primary sources. These findings were difficult to recover from any single source: they require joining papers, model cards, dataset cards, code, and recursively linked upstream artifacts. To the authors' best knowledge, these are either only known to a small number of experts or were not known even to authors of the original work based on our follow-up conversations with them. The examples below fall into six recurring audit patterns (further examples in Appendix E):

Multi-hop upstream models. A core advantage of recursive tracing is that it finds upstream models that never appear in the final model's own paper or card. These models are hidden behind intermediate datasets, filters, classifiers, teachers, or tools, so they only become visible after following hops.

- **DR Tulu depends on Claude-generated ScholarQA trajectories.** DR Tulu's paper states that their training data is generated by OpenAI models, but the Appendix additionally references Ai2 ScholarQA (Singh et al., 2025) trajectory data used to create SFT data. Following the ScholarQA implementation reveals that its generation pipeline uses Claude Sonnet 3.7 by default, finding a hidden dependency chain Claude Sonnet 3.7 (Anthropic, 2025) → Ai2 ScholarQA → DR Tulu.
- **Olmo 3 RL-Zero inherits Qwen2.5-Coder through its data-construction chain.** Olmo 3 RL-Zero models are trained on Dolci RL-Zero mixtures derived from earlier Olmo 3 checkpoints and code-oriented data construction. Tracing those intermediate artifacts back to Olmo 3's midtraining pipeline surfaces Qwen2.5-Coder-32B-Instruct (Hui et al., 2024) as an upstream model used to transform code data. This dependency is not stated in the downstream RL-Zero cards.

Training–evaluation coupling. Another finding is the pervasive *structural coupling* between training pipelines and evaluation benchmarks. These are often not direct cases of test-set leakage, but more subtle recursive relationships in which benchmark prompts, training splits, auxiliary resources, or validation environments are transformed into training artifacts while the same benchmark family remains an evaluation target. Such couplings are rarely visible from any single artifact and often evade conventional decontamination because the relevant connections span multiple hops.

- **Olmo training mixes reuse benchmark-derived data.** *IFEval*: Olmo 3 IF-RLVR prompts have their constraints sampled from IFEval (Zhou et al., 2023) and IFBench-Train (Pyatkin et al., 2025), while IFEval also appears as an evaluation target in the same release. *GSM8K*: Olmo 2's Dolmino-100 anneal mix lists the GSM8K (Cobbe et al., 2021) train splits directly in `dolmino100.txt` plus a TinyGSM (Liu et al., 2023a)-style synthetic expansion (`gsm_MIND/clean_stop`); GSM8K is reported as an evaluation target across both Olmo 2 (OLMo et al., 2025) and Olmo 3. This creates a structural train/eval coupling that is invisible from the model paper alone.
- **Nemotron products disagree on how to handle SWE-Bench-Verified.** Nemotron-3-Super repurposes data that was derived from SWE-Bench-Verified (OpenAI, 2024b) for RL training via `nvidia/Nemotron-RL-Agentic-SWE-Pivot-v1` (see commit `e57249`), then reports SWE-Bench-Verified as a headline evaluation. Nemotron-Cascade (Wang et al., 2025a) takes the opposite approach: it removes SFT examples whose source repositories appear in SWE-Bench-Verified before evaluating on the benchmark. Thus, the same benchmark is treated as a training signal in one Nemotron release and as contamination risk in another.

License-relevant paths. Potential license or terms-of-use implications are not always visible from the license attached to a final dataset or model. Prior work argues that dataset legal risk cannot be assessed from license terms alone and instead requires tracing redistribution and lifecycle history (Kim et al., 2025); supply-chain audits similarly find that license labels can function as weak metadata signals when the underlying compliance payload is missing or fails to propagate (Jewitt et al., 2026).

- **Major model families directly shape many downstream releases.** Even counting only one-hop training-side dependencies, a small set of model families dominate: Qwen touches 167 downstream artifacts through 552 edges, Llama touches 157 artifacts through 264 edges, GPT-4 (OpenAI, 2023) touches 65 through 125 edges, and DeepSeek touches 81 through 162 edges.
- **SmolLM3's FineMath data traces back to Llama-generated annotations.**[3] The SmolLM3 model card lists FineMath (Allal et al., 2025) as a pretraining source, and the FineMath card describes classifier-based filtering of mathematical web data. However, the classifier card finds that `finemath-classifier` was trained on Llama-3-70B-Instruct educational-value annotations. Thus, SmolLM3's pretraining data is shaped by a Llama-generated annotation pipeline even though the SmolLM3 card does not name Llama as an upstream source.

Model-mediated selection. Many upstream models do not appear as explicit training datasets. Instead, they generate, judge, filter, score, or reward candidate data—their outputs and preferences shape which examples are created, kept, or rewarded, even when no weights are copied. The graph also shows that this influence is highly concentrated: the same upstream model families recur as generators, filters, and judges across many releases, a pattern not visible from individual papers.

[3] Llama 3 Community License Agreement includes: *"You will not use the Llama Materials or any output or results of the Llama Materials to improve any other large language model."* Whether classifier-training annotations constitute "output of Llama Materials" used to "improve" SmolLM3 is an open interpretive question, but this multi-hop path illustrates potential license implications that manual auditing is unlikely to catch.

- **Qwen3-32B judges Olmo 3 RL data.** The Olmo 3 paper names Qwen3-32B (Team, 2025) as an LM judge, and training scripts apply it to RL-Zero prompts without verifiable ground truth. Thus, Qwen3-32B's preferences can shape which RL examples are retained or rewarded.
- **Nemotron-3-Super uses a Qwen-bootstrapped reward model.** Its generative reward model (GenRM) identifies Qwen3-235B-A22B-Thinking-2507 as the foundation model and names preference-data sources including `nvidia/HelpSteer3` (Wang et al., 2025b) and commercially-friendly subsets of `lmarena-ai/arena-human-preference-140k`. The resulting chain—Qwen3 foundation → GenRM judge → Nemotron-3-Super—spans the Nemotron paper, the GenRM card, and the preference-data card.

Code-level provenance. Release cards and papers often describe training data too coarsely, and the decisive evidence appears only in training scripts, YAML mixtures, or dataset-construction code.

- **Nemotron training blends are not fully specified until code fills in missing datasets.** Nemotron-Super-RL-Training-Blends includes placeholder rows for DAPO-Math (Yu et al., 2025) and Skywork-OR1 (He et al., 2025) contributions, requiring a separate `fill_placeholders.py` script to fetch and restore upstream data before the blend is usable. Similar placeholder mechanisms appear in Nemotron-3-Nano (NVIDIA, 2025b) and Nemotron-3-Nano-Omni (NVIDIA, 2026a), suggesting a recurring strategy in the Nemotron family.
- **SmolLM2's SmolTalk pipeline collapses legacy corpora and teacher/filter models into one SFT dataset name.** The SmolLM2 (Allal et al., 2025) model card points to SmolTalk (Allal et al., 2025)-style SFT data, and the SmolTalk card describes broad components such as summarization, rewriting, and Magpie-style instruction data. However, the code reveals the concrete upstream construction path: the summarization branch loads CNN DailyMail (Hermann et al., 2015) and processes it through Qwen2.5-72B-Instruct, while the Magpie-Ultra branch generates instructions with Llama-3.1-405B-Instruct and filters them with Llama-3.1-8B-Instruct.

Mitigations and hygiene. The graph also shows intentional good practices from model developers.

- **Ai2 rebuilds upstream recipes with more permissive teachers.** Three Olmo 3 midtraining subsets share the same design pattern: CraneMath reimplements the SwallowMath (Fujii et al., 2025) recipe using Qwen3 instead of Llama; CraneCode applies the same teacher swap to a SwallowCode (Fujii et al., 2025)-style rewriting pipeline using Qwen2.5-Coder-32B-Instruct; and MegaMatt recreates a MegaMath (Zhou et al., 2025)-style methodology with Qwen3.
- **Olmo 3 filters Llama-Nemotron data to avoid Llama-touched samples.** Olmo 3 includes Llama-Nemotron Post-Training data in its training mixture, but the retained reasoning samples are filtered to DeepSeek and Qwen samples that were not touched by Llama models.

5.3 Implications and Responsible Use

Our findings suggest that auditing modern LLMs requires following dependency paths rather than inspecting each artifact in isolation. We therefore encourage developers to document the models that generate, filter, rewrite, or judge their data, with exact versions, on the corresponding cards. Until such documentation becomes standard, ModSleuth can help human auditors trace these paths. The released graph records verifier verdicts, conflict flags with competing source excerpts, and artifact identities resolved only to the level supported by the evidence. Auditors should check these signals and the original sources, especially for disputed or consequential claims. We discuss harms of incorrect or missing edges in the Ethics statement and other limitations in the Limitations section.

6 Conclusion

We present ModSleuth, an agentic system that reconstructs recursive, evidence-grounded dependency graphs of LLM releases from public artifacts, built on explicit semantics for what counts as a dependency and how identities are reconciled. Across four LLM releases, ModSleuth recovers 1,060 verified dependencies (a lower bound on the true dependency structure), and human annotators support 94.7% of sampled accepted edges. The resulting graphs surface issues difficult to identify from individual sources, including license-relevant multi-hop paths, structural train/evaluation coupling, mismatches between released and trained-on artifacts, and documentation inconsistencies.

AI USE STATEMENT

Generative AI is part of the method studied in this paper: ModSleuth uses Claude Code to extract dependencies and construct the released graphs, Claude models and GLM-5.2 with web search verify recovered dependencies, and GPT-5.5 Pro, GPT-5.4 Pro, Claude Code, and ChatGPT Deep Research serve as baselines (§3, §4). Beyond these roles, we used generative AI tools to implement methods, process and filter data, and run experiments. We did not use them to formulate hypotheses or conceptual frameworks, design the research methodology, or interpret results, and tasks involving mathematical claims, proofs, or translation are not applicable to this work. We additionally used generative AI tools to write code, create figures, understand technical concepts, draft parts of the paper, and edit it for readability. We have reviewed all AI-assisted work: two human annotators audited a random sample of recovered edges (Appendix B.2), every finding reported in the paper was manually verified against its primary sources, and all AI-assisted code, figures, and text were checked and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work analyzes only publicly released artifacts (papers, model and dataset cards, and code) and collects no personal data. Because recovered dependencies bear on the licensing, contamination, and evaluation practices of specific developers, an erroneous edge could unfairly implicate them. We mitigate this risk by grounding every edge in cited source excerpts, auditing a sample of edges with human annotators, and manually verifying every finding reported in the paper. ModSleuth's findings are leads for verification rather than legal determinations. A license-relevant path does not by itself establish a license violation, and a structural connection between training and evaluation does not by itself establish test-set leakage. Conversely, the recovered graph is a lower bound, so a missing edge does not imply compliance. We report good practices alongside risks (§5) and hope the released graphs help developers document their own dependencies.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlícek, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big - data-centric training of a small language model. *CoRR*, abs/2502.02737, 2025. doi: 10.48550/ARXIV.2502.02737. URL https://doi.org/10.48550/arXiv.2502.02737.

Anthropic. Claude 3.7 Sonnet and Claude Code, February 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Anthropic. Introducing Claude Sonnet 4.6, February 2026a. URL https://www.anthropic.com/news/claude-sonnet-4-6.

Anthropic. Claude code documentation. https://code.claude.com/docs/en/overview, 2026b.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fUtxNAKpdV.

Rishi Bommasani, Kevin Klyman, Sayash Kapoor, Shayne Longpre, Betty Xiong, Nestor Maslej, and Percy Liang. The 2024 foundation model transparency index. *Trans. Mach. Learn. Res.*, 2025, 2025a. URL https://openreview.net/forum?id=38cwP8xVxD.

Rishi Bommasani, Dilara Soylu, Thomas I. Liao, Kathleen A. Creel, and Percy Liang. Ecosystem graphs: Documenting the foundation model supply chain. In *Proceedings of the 2024 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '24, pp. 196–209. AAAI Press, 2025b.

Cisco Systems, Inc. and its affiliates. Model provenance kit, 2026. URL https://github.com/cisco-ai-defense/model-provenance-kit. Deep-signal fingerprints (CC BY 4.0) may be distributed separately on the Hugging Face Hub.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data. *CoRR*, abs/2507.14805, 2025. doi: 10.48550/ARXIV.2507.14805. URL https://doi.org/10.48550/arXiv.2507.14805.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. ULTRAFEEDBACK: boosting language models with scaled AI feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 9722–9744. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/cui24f.html.

Alexandre Decan, Tom Mens, and Philippe Grosjean. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empir. Softw. Eng.*, 24(1): 381–416, 2019. doi: 10.1007/S10664-017-9589-Y. URL https://doi.org/10.1007/s10664-017-9589-y.

Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee F. Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, William Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. *CoRR*, abs/2512.13961, 2025. doi: 10.48550/ARXIV.2512.13961. URL https://doi.org/10.48550/arXiv.2512.13961.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Ohi, Masaki Kawamura, Taishi Nakamura, Takumi Okamoto, Shigeki Ishida, Kakeru Hattori, Youmi Ma, Hiroya Takamura, Rio Yokota, and Naoaki Okazaki. Rewriting pre-training data boosts LLM performance in math and code. *CoRR*, abs/2505.02881, 2025. doi: 10.48550/ARXIV.2505.02881. URL https://doi.org/10.48550/arXiv.2505.02881.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Commun. ACM*, 64(12):86–92, 2021. doi: 10.1145/3458723. URL https://doi.org/10.1145/3458723.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai,

Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *CoRR*, abs/2306.11644, 2023. doi: 10.48550/ARXIV.2306.11644. URL https://doi.org/10.48550/arXiv.2306.11644.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, Tao Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nat.*, 645(8081):633–638, 2025. doi: 10.1038/S41586-025-09422-Z. URL https://doi.org/10.1038/s41586-025-09422-z.

David Hall, Ahmed Ahmed, Christopher Chou, Abhinav Garg, Rohith Kuditipudi, Will Held, Nikil Ravi, Herumb Shandilya, Jason Wang, Jason Bolton, Siddharth Karamcheti, Suhas Kotha, Tony Lee, Nelson Liu, Joel Niklaus, Ashwin Ramaswami, Kamyar Salahi, Kaiyue Wen, Chi Heem Wong, Sherry Yang, Ivan Zhou, and Percy Liang. Introducing Marin: An open lab for building foundation models. Marin Community Blog, May 2025. URL https://marin.community/blog/2025/05/19/announcement/. Blog post.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *CoRR*, abs/2505.22312, 2025. doi: 10.48550/ARXIV.2505.22312. URL https://doi.org/10.48550/arXiv.2505.22312.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In Corinna

Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1693–1701, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/afdec7005cc9f14302cd0474fd0f3c96-Abstract.html.

Eliahu Horwitz, Nitzan Kurer, Jonathan Kahana, Liel Amar, and Yedid Hoshen. We should chart an atlas of all the world's models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/d7c52556f617136b3a0558708678ec6c-Abstract-Position_Paper_Track.html.

Eliahu Horwitz, Asaf Shul, and Yedid Hoshen. Unsupervised model tree heritage recovery. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=QVj3kUvdvl.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186, 2024. doi: 10.48550/ARXIV.2409.12186. URL https://doi.org/10.48550/arXiv.2409.12186.

James Jewitt, Gopi Krishnan Rajbahadur, Hao Li, Bram Adams, and Ahmed E. Hassan. Permissive-washing in the open AI supply chain: A large-scale audit of license integrity. *CoRR*, abs/2602.08816, 2026. doi: 10.48550/ARXIV.2602.08816. URL https://doi.org/10.48550/arXiv.2602.08816.

Jaehun Jung, Seungju Han, Ximing Lu, Skyler Hallinan, David Acuna, Shrimai Prabhumoye, Mostafa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Prismatic synthesis: Gradient-based data diversification boosts generalization in LLM reasoning. *CoRR*, abs/2505.20161, 2025. doi: 10.48550/ARXIV.2505.20161. URL https://doi.org/10.48550/arXiv.2505.20161.

Jaekyeom Kim, Sungryull Sohn, Gerrard Jeongwon Jo, Jihoon Choi, Kyunghoon Bae, Hwayoung Lee, Yongmin Park, and Honglak Lee. Do not trust licenses you see: Dataset compliance requires massive-scale ai-powered lifecycle tracing. *CoRR*, abs/2503.02784, 2025. doi: 10.48550/ARXIV.2503.02784. URL https://doi.org/10.48550/arXiv.2503.02784.

Rohith Kuditipudi, Jing Huang, Sally Zhu, Diyi Yang, Christopher Potts, and Percy Liang. Blackbox model provenance via palimpsestic membership inference. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/eda4f6fdf680bcd2b81c969a4090856a-Abstract-Conference.html.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in llm-as-a-judge. *CoRR*, abs/2502.01534, 2025. doi: 10.48550/ARXIV.2502.01534. URL https://doi.org/10.48550/arXiv.2502.01534.

Yu Li, Xiaoran Shang, Qizhi Pei, Yun Zhu, Xin Gao, Honglin Lin, Zhanping Zhong, Zhuoshi Pan, Zheng Liu, Xiaoyang Wang, Conghui He, Dahua Lin, Feng Zhao, and Lijun Wu. Tracing the roots: A multi-agent framework for uncovering data lineage in post-training LLMs. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9606–9625, San Diego, California, United States, July 2026. Association for Computational

Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.435. URL https://aclanthology.org/2026.acl-long.435/.

Weixin Liang, Nazneen Rajani, Xinyu Yang, Ezinwanne Ozoani, Eric Wu, Yiqun Chen, Daniel Scott Smith, and James Zou. Systematic analysis of 32,111 AI model cards characterizes documentation practice in AI. *Nat. Mac. Intell.*, 6(7):744–753, 2024. doi: 10.1038/S42256-024-00857-Z. URL https://doi.org/10.1038/s42256-024-00857-z.

Bingbin Liu, Sébastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving >80% on gsm8k with small language models. *CoRR*, abs/2312.09241, 2023a. doi: 10.48550/ARXIV.2312.09241. URL https://doi.org/10.48550/arXiv.2312.09241.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, Xinyi Wu, Enrico Shippole, Kurt Bollacker, Tongshuang Wu, Luis Villa, Sandy Pentland, and Sara Hooker. A large-scale audit of dataset licensing and attribution in AI. *Nat. Mac. Intell.*, 6(8):975–987, 2024. doi: 10.1038/S42256-024-00878-8. URL https://doi.org/10.1038/s42256-024-00878-8.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In danah boyd and Jamie H. Morgenstern (eds.), *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, pp. 220–229. ACM, 2019. doi: 10.1145/3287560.3287596. URL https://doi.org/10.1145/3287560.3287596.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *CoRR*, abs/2306.02707, 2023. doi: 10.48550/ARXIV.2306.02707. URL https://doi.org/10.48550/arXiv.2306.02707.

NVIDIA. NVIDIA nemotron 3: Efficient and open intelligence. *CoRR*, abs/2512.20856, 2025a. doi: 10.48550/ARXIV.2512.20856. URL https://doi.org/10.48550/arXiv.2512.20856.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *CoRR*, abs/2512.20848, 2025b. doi: 10.48550/ARXIV.2512.20848. URL https://doi.org/10.48550/arXiv.2512.20848.

NVIDIA. Nemotron 3 nano omni: Efficient and open multimodal intelligence. *CoRR*, abs/2604.24954, 2026a. doi: 10.48550/ARXIV.2604.24954. URL https://doi.org/10.48550/arXiv.2604.24954.

NVIDIA. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *CoRR*, abs/2604.12374, 2026b. doi: 10.48550/ARXIV.2604.12374. URL https://doi.org/10.48550/arXiv.2604.12374.

Hamidah Oderinwale, Benjamin Laufer, and Jon Kleinberg. Anatomy of a machine learning ecosystem: 2 million models on hugging face. In *NeurIPS 2025 Workshop on Regulatable ML*, 2025. URL https://openreview.net/forum?id=X9CGbl59yP.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman

Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious. *CoRR*, abs/2501.00656, 2025. doi: 10.48550/ARXIV.2501.00656. URL https://doi.org/10.48550/arXiv.2501.00656.

OpenAI. Introducing ChatGPT, November 2022. URL https://openai.com/index/chatgpt.

OpenAI. GPT-4, March 2023. URL https://openai.com/index/gpt-4-research.

OpenAI. Hello GPT-4o, May 2024a. URL https://openai.com/index/hello-gpt-4o.

OpenAI. Introducing SWE-bench Verified, August 2024b. URL https://openai.com/index/introducing-swe-bench-verified/.

OpenAI. Introducing deep research, February 2025. URL https://openai.com/index/introducing-deep-research.

OpenAI. Introducing GPT-5.4, March 2026a. URL https://openai.com/index/introducing-gpt-5-4/.

OpenAI. Introducing GPT-5.5, April 2026b. URL https://openai.com/index/introducing-gpt-5-5/.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/7f1f0218e45f5414c79c0679633e47bc-Abstract-Conference.html.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. Data cards: Purposeful and transparent dataset documentation for responsible AI. In *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, pp. 1776–1826. ACM, 2022. doi: 10.1145/3531146.3533231. URL https://doi.org/10.1145/3531146.3533231.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *CoRR*, abs/2507.02833, 2025. doi: 10.48550/ARXIV.2507.02833. URL https://doi.org/10.48550/arXiv.2507.02833.

Mohammad Shahedur Rahman, Peng Gao, and Yuede Ji. Hugginggraph: Understanding the supply chain of LLM ecosystem. In Meeyoung Cha, Chanyoung Park, Noseong Park, Carl Yang, Senjuti Basu Roy, Jessie Li, Jaap Kamps, Kijung Shin, Bryan Hooi, and Lifang He (eds.), *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10-14, 2025*, pp. 5997–6005. ACM, 2025. doi: 10.1145/3746252.3761510. URL https://doi.org/10.1145/3746252.3761510.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10776–10787, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.722. URL https://aclanthology.org/2023.findings-emnlp.722/.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David A. Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen-tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR tulu: Reinforcement learning with evolving rubrics for deep research. *CoRR*, abs/2511.19399, 2025. doi: 10.48550/ARXIV.2511.19399. URL https://doi.org/10.48550/arXiv.2511.19399.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross J. Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nat.*, 631(8022):755–759, 2024. doi: 10.1038/S41586-024-07566-Y. URL https://doi.org/10.1038/s41586-024-07566-y.

Amanpreet Singh, Joseph Chee Chang, Dany Haddad, Aakanksha Naik, Jena D. Hwang, Rodney Kinney, Daniel S Weld, Doug Downey, and Sergey Feldman. Ai2 scholar QA: Organized literature synthesis with attribution. In Pushkar Mishra, Smaranda Muresan, and Tao Yu (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 513–523, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-253-4. doi: 10.18653/v1/2025.acl-demo.49. URL https://aclanthology.org/2025.acl-demo.49/.

Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Laura A. Heymann, Massimiliano Di Penta, Daniel M. Germán, and Denys Poshyvanyk. The ML supply chain in the era of software 2.0: Lessons learned from hugging face. *CoRR*, abs/2502.04484, 2025. doi: 10.48550/ARXIV.2502.04484. URL https://doi.org/10.48550/arXiv.2502.04484.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *CoRR*, abs/2512.13607, 2025a. doi: 10.48550/ARXIV.2512.13607. URL https://doi.org/10.48550/arXiv.2512.13607.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b. URL http://papers.nips.cc/paper_files/paper/2025/hash/3e0271cf7df2cdb3b91565ad1f525f3a-Abstract-Datasets_and_Benchmarks_Track.html.

Yixin Wu, Ziqing Yang, Yun Shen, Michael Backes, and Yang Zhang. Synthetic artifact auditing: Tracing llm-generated synthetic data usage in downstream applications. In Lujo Bauer and Giancarlo Pellegrino (eds.), *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*, pp. 1689–1708. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/wu-yixin-auditing.

Zhaomin Wu, Haodong Zhao, Ziyang Wang, Jizhou Guo, Qian Wang, and Bingsheng He. LLM DNA: tracing model evolution via functional representations. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0c606cbd7225856faadf74e106cbbec6-Abstract-Conference.html.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=CfXh93NDgH.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024a. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024b. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples. *CoRR*, abs/2311.04850, 2023. doi: 10.48550/ARXIV.2311.04850. URL https://doi.org/10.48550/arXiv.2311.04850.

Xinyu Yang, Weixin Liang, and James Zou. Navigating dataset documentations in AI: A large-scale analysis of dataset cards on huggingface. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024c. URL https://openreview.net/forum?id=xC8xh2RSs2.

Nicolas Yax, Pierre-Yves Oudeyer, and Stefano Palminteri. Phylolm: Inferring the phylogeny of large language models and predicting their performances in benchmarks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=rTQNGQxm4K.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Z.ai. GLM-5.2: Built for long-horizon tasks, June 2026. URL https://z.ai/blog/glm-5.2.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Fan Zhou, Zengzhi Wang, Nikhil Ranjan, Zhoujun Cheng, Liping Tang, Guowei He, Zhengzhong Liu, and Eric P. Xing. Megamath: Pushing the limits of open math corpora. *CoRR*, abs/2504.02807, 2025. doi: 10.48550/ARXIV.2504.02807. URL https://doi.org/10.48550/arXiv.2504.02807.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *CoRR*, abs/2311.07911, 2023. doi: 10.48550/ARXIV.2311.07911. URL https://doi.org/10.48550/arXiv.2311.07911.

Sally Zhu, Ahmed M. Ahmed, Rohith Kuditipudi, and Percy Liang. Independence tests for language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhu25a.html.

A Full ModSleuth Design

Here we give a more detailed description of our system.

Phase 1: Source gathering. Dependency evidence is scattered across many artifacts, and no single source type captures the full dependency structure of a modern LLM release. A technical report may describe the training pipeline without naming the exact dataset versions used in code; a dataset card may identify an upstream generator absent from the paper; and repository configuration files may reference mixtures never described in prose.

ModSleuth therefore begins from a target release and gathers its official public artifacts, including technical reports, model and dataset cards, code repositories, release blogs, and linked upstream artifacts. We restrict the system to official sources to avoid relying on unverified third-party claims. Artifacts are stored locally, and all later stages and evidence anchors refer to these copies rather than live URLs. Because undocumented dependencies fall outside this evidence scope, the recovered graph should be interpreted as a lower bound on the true dependency structure. To manage long contexts, the collected artifacts are organized into topically coherent batches, such as all resources associated with a model family, training stage, or dataset mixture.

Phase 2: Entity discovery and resolution. In the next phase, the system identifies dependent model and dataset artifacts from each source batch, recording mentions exactly as they appear in the source together with supporting evidence spans.

As discussed in §3.1, a key challenge is entity ambiguity: the same artifact is often referenced at different levels of specificity across sources (e.g., "Olmo 3 32B," vs. `Olmo-3-32B-Think` vs. `Olmo-3-1125-32B`), while datasets frequently appear under subsets, derived variants, or internal names. The system therefore maps extracted mentions into an *identity lattice* defined in §3.1, e.g., {`family:Olmo 3, size:32B, stage:Think`}.

For Hugging Face artifacts, deterministic metadata retrieval validates canonical URLs and recursively follows parent, subset, and derivation relationships when available. This enables ambiguous dataset

names and internal slugs to be mapped to the correct public artifacts. Identity collisions, nonexistent artifacts, and inconsistent resolutions are flagged before dependency construction, preventing these errors from propagating into the dependency graph.

Phase 3: Dependency construction and reconciliation. With artifact identities resolved, the goal of the next phase is to transform evidence scattered across sources into a unified, evidence-grounded dependency graph. The system re-reads each source batch against the resolved identity lattice. A lattice search tool allows the agent to resolve mentions to the most specific node supported by the evidence and emit structured operations. Each edge records upstream and downstream artifacts, a free-form role description, a coarse dependency label, and supporting source anchors. Edges that reference unresolved artifacts or lack supporting evidence are rejected.

A key challenge is reconciling relationships described across multiple sources. The same dependency may be referenced at different levels of specificity or through complementary pieces of evidence. ModSleuth therefore merges claims that describe the same relationship, preserving the most specific artifact identities available while aggregating supporting evidence. When sources disagree, e.g., by assigning a dependency to different sibling artifacts or describing incompatible roles, the system flags the case for review rather than silently resolving it. Flagged cases are examined by a dedicated audit stage, with unresolved cases escalated to human annotation.

Conflicts that remain unresolved after automated reconciliation and human review are retained in the released graph with an explicit conflict flag and all competing source excerpts. This preserves disagreements in public documentation for downstream inspection. For unresolved cases, we recommend consulting the original developers for clarification. The resulting dependency operations trace paths from every discovered artifact back to the target release, yielding a unified dependency graph.

Recursive expansion. The preceding phases recover the local dependency neighborhood described by the target release's own public artifacts. However, many important dependencies are only visible by recursively tracing upstream artifacts. For example, a target release may identify an OCR model used in data preprocessing, but the target release typically will not document the OCR model's own base checkpoint, training data, or synthetic-data generators.

To recover this transitive structure, ModSleuth recursively applies the same pipeline to discovered upstream artifacts. Each newly discovered model or dataset can become a tracing target whose official artifacts are gathered, resolved, related, and reconciled into the global graph. Users may choose breadth-first search for maximal coverage, depth-first search for targeted investigation of particular chains, or beam search to expand the top-K structurally central ancestors at each depth. This recursive expansion is what turns scattered one-hop evidence into an auditable dependency graph over the broader model and dataset ecosystem.

B Evaluation Protocol Details

Our evaluation has three steps: we pool candidate relationships, verify each relationship against public evidence, and then attribute verified/refuted relationships to target models for reporting.

Candidate relationship pooling. For each target model, we collect the dependency relationships emitted by ModSleuth and all baselines. Each candidate relationship contains a subject artifact, an object artifact, a free-form dependency description, relation metadata, and supporting evidence URLs or excerpts.

Verification. Each candidate relationship is checked by Claude Sonnet 4.6 with web search. The BFS reachability scope and the full-graph aggregate use a separate audit by Claude Opus 4.7 with extended thinking and web search. The verifier reads the submitted evidence, checks cited excerpts or locations when available, and may search for independent public corroboration.

Attribution scopes for ModSleuth. Unlike the baselines, ModSleuth produces one merged, entity-resolved graph across the four target investigations. We therefore need explicit rules for assigning each relationship in the merged graph back to one or more target models. We report three scopes.

Depth-1 is the strict scope. A relationship is attributed to target T only when its subject canonicalizes to T's canonical identifier. Canonicalization lowercases the string and collapses non-alphanumeric runs to hyphens, while preserving any HuggingFace organization prefix. This scope is closest to the per-target outputs produced by single-prompt baselines: it only counts relationships where the target itself is the subject.

Unbounded combines worker provenance with forward graph reachability. A relationship is attributed to target T if *both* of the following hold:

1. **Provenance**: at least one of (i) its subject canonicalizes to T, (ii) at least one supporting anchor comes from T's target-specific seed directory, or (iii) at least one supporting anchor was produced by a worker whose outputs co-occur only with T's seed directory.
2. **Forward reachability**: the subject node is forward-reachable from T in the merged graph by traversing edges in the direction subject $\rightarrow$ object.

Seed directories contain the papers, model cards, dataset cards, READMEs, and training configs gathered specifically for one target investigation. Worker IDs identify short-lived extraction or relation-generation jobs; if a worker only co-occurs with one target's seed materials, we treat its outputs as part of that target's recursive investigation.

The provenance criterion captures edges actually discovered during T's investigation, while the forward-reachability criterion ensures those discoveries are about artifacts in T's dependency chain rather than context the worker happened to read about (e.g., sibling or predecessor artifacts in the same family). Together, the two criteria capture what each target investigation contributed to its own chain. A relationship may be attributed to multiple targets when its provenance spans multiple target investigations and it is forward-reachable from each. For completeness, we additionally report a broader graph-reachability scope, defined below, which counts every relation reachable from T regardless of which investigation surfaced it. Under this nesting, depth-1 $\subset$ unbounded $\subset$ BFS reach.

BFS reachability is the graph-reachability scope. A relationship is attributed to target T if its subject node is forward-reachable from T in the merged graph by traversing edges in the direction subject $\rightarrow$ object. An edge may be attributed to multiple targets when several seeds forward-reach it; the Total under this scope reports the union of reachable edges across the four seeds rather than their sum. This scope captures the full transitive dependency footprint of each target, including relationships uncovered by separate sub-investigations of upstream artifacts that share lineage with T. Sibling and predecessor artifacts that are not graph-downstream of any seed (e.g., earlier-generation models in the same family) fall outside this scope.

Under this unbounded attribution rule, 1,060 of the 9,112 relations in the merged graph are attributed to at least one of the four targets. Under the BFS reachability scope, 1,654 unique edges in the merged graph are forward-reachable from at least one of the four targets.

B.1 Attempts to Construct Reference Graphs

We attempted to construct human-written reference graphs in two ways. First, researchers with prior experience in open-model development, including authors of this work, manually traced dependencies from public release artifacts. Second, we interviewed authors of Olmo 3 and DR Tulu and asked them to enumerate dependencies of their own models.

More than three hours of expert reconstruction yielded approximately 25 verified edges. The resulting reference graphs typically terminated after one or two hops and relied primarily on technical reports and model cards. They frequently omitted less obvious dependencies such as judges, filters, rewriters, and embedding models. By comparison, ModSleuth recovers chains up to eight hops deep and identifies dependencies documented in training code as well as prose artifacts.

The developer interviews also illustrated the difficulty of reconstructing dependency graphs. The DR Tulu authors initially attributed their generated training data to OpenAI models and confirmed the Claude Sonnet 3.7 dependency identified through ScholarQA only after inspecting their development logs. These attempts motivate our complementary evaluation of verified recovery and human support for accepted edges, while leaving absolute recall unmeasured.

Table 3: Human audit of 300 randomly sampled verifier-accepted edges. Rows are annotator H2's labels and columns are annotator H1's; bold diagonal cells mark agreement. Both annotators supported 284/300 edges (94.7%), and their labels agreed on 293/300 (97.7%).

	H1			
H2	Supported	Unsupported	Unclear	Total
Supported	**284**	2	0	286
Unsupported	4	**8**	0	12
Unclear	1	0	**1**	2
Total	289	10	1	300

B.2 Validating the Verifier

We conducted two studies on verifier-accepted edges from the released graph to assess the reliability of accepted dependency claims beyond the primary Claude-based verification.

Human audit. We randomly sampled 300 accepted edges and had two human annotators independently label each as supported, unsupported, or unclear. Annotators received the same inputs as the LLM verifier, including the artifact endpoints, relationship description, and supporting evidence. They traced every cited excerpt back to its primary source and were blind to each other's judgments.

Table 3 reports the joint annotation counts. Both annotators supported 284/300 edges (94.7%; 95% CI 91.5–96.7). Their labels agreed on 293/300 edges (97.7%): 284 jointly supported, eight jointly unsupported, and one jointly unclear. Cohen's κ is 0.71 (substantial agreement). The remaining seven edges received different labels. Joint support and inter-annotator agreement measure different properties: the former measures how often both annotators supported an accepted edge, whereas the latter includes agreement that an edge was unsupported or unclear.

In this audit, humans were instructed to verify explicit claims rather than reconstruct a dependency graph from scratch. This task is easier and bounded since the annotator is given the endpoints, relationship description, and cited evidence, and checks whether the source supports the claim.

Cross-model verification. We additionally sampled 500 accepted edges from the released graph and re-verified them using GLM-5.2 (Z.ai, 2026) with web search, using the same verification prompt and inputs. GLM-5.2 upheld 483 edges (96.6%; 95% CI 94.6–97.9), refuted 16, and marked one unclear. This study measures how often a different model family upholds the primary verifier's accepted decisions. It provides a complementary check, but does not establish statistical independence between the verifiers' errors.

Failure modes and scope. Potential verifier failures include accepting an excerpt that is misleading outside its original context, confusing a model family with a specific variant, and relying on web corroboration that changes over time or remains inconclusive. The human protocol checks source context and artifact identity directly, while cross-model verification provides an additional assessment of the same accepted claims. Because both studies sample accepted edges, they do not measure false rejections or dependencies missed during discovery.

C Baseline Prompt

All baseline systems were given the same dependency-reconstruction prompt template, summarized in Table 4. For each target model, we instantiated a `SUBJECT` block with the target's canonical identifier, display name, provider, release date, authoritative paper/repository/card URLs, scope note, and recursion depth. The prompt instructed baselines to recover both direct training-pipeline dependencies and indirect development dependencies, then emit a single JSON graph containing `subject`, `nodes`, and `edges`. It also specified the dependency scope, canonicalization rules, evidence requirements, recursion policy, and output validation checks. The full prompt template is provided in the supplementary artifacts as `baseline_prompt.md`.

Table 4: Structure of the baseline prompt used for all comparison systems.

Prompt component	Description
Subject block	Specifies the target model, authoritative sources, scope note, and maximum recursion depth.
Dependency scope	Defines direct dependencies as artifacts that enter the training pipeline and indirect dependencies as artifacts that shape evaluation, ablation, or methodology decisions.
Node and edge schema	Requires model/dataset nodes and evidence-grounded edges with `relation_type`, `dependency_kind`, free-form descriptions, and supporting evidence.
Canonicalization rules	Standardizes model and dataset identifiers, aliases, provider names, and version conventions to improve alignment across systems.
Evidence requirements	Requires each edge to include source URLs and explanations, with locations and excerpts recommended for verifiability.
Research protocol	Instructs the baseline to read authoritative papers, model cards, dataset cards, repositories, release blogs, and upstream artifact documentation.
Recursion policy	Requires recursive expansion of upstream artifacts until the depth cap is reached or authoritative documentation runs out.
Validation checklist	Asks the baseline to check schema validity, node coverage, canonical-ID consistency, evidence grounding, recursion coverage, and exclusion of out-of-scope artifacts before emitting JSON.

Table 5: Audit roles and their constituent relation types. Verified edge counts per role and target are reported in Table 2.

Audit role	Constituent relations
Direct roles	
Training inputs	`trained_on`
Upstream operations on training data	`generated_by`, `filtered_by`, `transformed_by`, `embedded_by`, `decontaminated_against`, `composed_from`
Weight-level model lineage	`trained_from`, `merged_from`, `quantized_from`
Indirect roles	
Evaluation / ablation	`used_for_evaluation`, `used_for_ablation`
Methodology / audit influence	`inspired_by`

D Additional Quantitative Results

D.1 Recovered Graph Scale by Target

Table 6: Scale and depth of recovered dependency graphs. Ancestors count unique transitive upstream artifacts reachable from each target model.

Target model	Ancestors	Max depth
Olmo 3 Instruct	409	8
Olmo 3 Think	329	8
Olmo 3 Base	181	4
Nemotron-3-Super	465	5
Nemotron-3-Nano-Base	319	5
DR Tulu	17	2
SmolLM3-Base	78	3
Aggregate graph	2,526 nodes, 9,112 edges, 36,187 anchors	

Table 6 shows the per-target ancestor counts and maximum depths. The variance in max depth across targets reflects pipeline complexity rather than recovery quality: SmolLM3-Base is a pretraining-only release whose recoverable lineage is dominated by web-scrape and filter steps (depth 3), whereas

Table 7: Source-type distribution of recovered edges. Single-source edges are supported by anchors from only one source class; multi-source edges are supported by anchors from multiple source classes.

Source support	Count	Share
Only Hugging Face cards	669	7.3%
Only training code / scripts	2,142	23.5%
Only PDFs / technical reports	2,777	30.5%
Only release blogs / docs	18	0.2%
Only other docs	486	5.3%
Multiple source types	3,020	33.1%
Total edges	9,112	100%

Table 8: Internal vs. external dependencies per target. An edge is *internal* if its upstream artifact shares the target's organization (e.g., Ai2 artifacts for Olmo 3 and DR Tulu, NVIDIA for Nemotron 3, HuggingFace for SmolLM3), and *external* otherwise. Counts are verified BFS-reachable edges per target; an edge reached from multiple targets is counted in each.

	Internal		**External**				
Target	Model	Data	Model	Data	Int. total	Ext. total	Ext. %
Olmo 3	13	106	90	272	119	362	75.3%
Nemotron 3	40	203	277	503	243	780	76.2%
DR Tulu	2	9	17	34	11	51	82.3%
SmolLM3	3	28	17	79	31	96	75.6%

the Olmo 3 Instruct/Think families pass through midtraining, multiple SFT/DPO/RL stages, and judge-mediated synthetic-data construction (depth 8). Lineage size is also lower-bounded by what upstream documentation makes recoverable; releases with shallower public artifact trees (e.g., DR Tulu) yield correspondingly smaller ancestor sets.

Internal vs. external dependencies. We additionally classify each verified BFS-reachable edge as *internal* if its upstream artifact shares the target's organization (e.g., `allenai/`$\star$ artifacts for Olmo 3 and DR Tulu, `nvidia/`$\star$ for Nemotron 3 Super, `HuggingFaceTB/`$\star$ for SmolLM3) and *external* otherwise. Table 8 reports this split. Across all four targets, external dependencies dominate: 75–82% of verified edges come from outside the target's organization. Olmo 3 reaches 90 external models (e.g., `openai/gpt-4.1`, `Qwen/Qwen3-32B`, `Qwen/QwQ-32B`) versus 13 internal Ai2 models (e.g., `allenai/OLMo-2`, `allenai/wildguard`); the dataset-level ratio is similar (272 external vs. 106 internal). OpenAI and Qwen are the most depended-on external organizations across all four targets.

E Additional Qualitative Findings

The main text presents a curated subset of qualitative findings. This appendix gives additional examples that illustrate the same recurring audit patterns: hidden model-mediated data construction, benchmark exposure, release-artifact mismatches, and reproducibility issues.

E.1 Hidden Model-Mediated Data Construction

- **Nemotron-PrismMath routes Qwen-generated math data into Nemotron training.** Nemotron-3-Super and Nemotron-3-Nano variants train on `nvidia/Nemotron-PrismMath` (Jung et al., 2025), whose card identifies Qwen2.5-72B-Instruct (Yang et al., 2024a), Qwen2.5-0.5B-Instruct, and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025) as generators for its problem–solution pairs.
- **Nemotron-CC-v2.1 exposes Qwen3-rephrased Common Crawl as a downstream training source.** Nemotron-CC-v2.1 extends Common Crawl-derived data with synthetic rephrasing and translation using Qwen3-30B-A3B. User-facing model cards often describe this as synthetic

Common Crawl, while dataset documentation exposes the upstream rephrasing model and the specific subset construction.

- **Nemotron-3-Super uses training content generated in part by Nemotron-Nano-9B-v2.** The recovered graph identifies `nvidia/NVIDIA-Nemotron-Nano-9B-v2` as a generator for content used in Nemotron-3-Super training. This is a non-obvious family-internal data path: a smaller sibling model contributes training content for a larger downstream release.
- **SmolLM2 is filtered by a Llama-3-derived reward model.** Following the SmolTalk pipeline reveals that `smol-magpie-ultra` examples are scored or filtered with `RLHFlow/ArmoRM-Llama3-8B-v0.1`. This makes a Llama-3-derived reward model an upstream filtering dependency for SmolLM2-Instruct, even though the dependency is not visible from the final model card alone.

E.2 Benchmark Exposure and Decontamination

- **Popular benchmarks repeatedly play both training-side and evaluation roles.** Aggregating roles per benchmark, ModSleuth reveals that several benchmarks appear both as evaluation targets and as training-side artifacts: GSM8K (25 evaluation edges, 43 training edges), MMLU (Hendrycks et al., 2021a) (39 evaluation, 14 training), GPQA (Rein et al., 2023) (45 evaluation, 9 training), MATH (Hendrycks et al., 2021b) (31 evaluation, 30 training), IFEval (27 evaluation, 18 training), and SWE-bench Verified (2 evaluation, 10 training).
- **LiveCodeBench appears both as RL validation and as an evaluation benchmark.** A Nemotron training configuration references `livecodebench_v5_validation` as a validation split for a code-generation environment, while LiveCodeBench-style results are also reported as evaluation metrics. This raises a development-coupling question: validation benchmarks can influence checkpoint selection or training decisions even when they are not directly used as supervised training data.
- **Olmo 3 RL-Zero applies a "spurious-reward decontamination check" against an enumerated benchmark suite.** Olmo 3-7B-RL-Zero-Math, -Code, and -Mix carry edges flagging spurious-reward decontamination checks against GSM8K, GPQA, AIME 2024, AIME 2025, Minerva Math, ZebraLogic, and Omega-500. These edges are scattered across model and dataset cards plus paper figure captions.
- **Nemotron uses Qwen3-Embedding-0.6B as a decontamination filter.** The graph records that `Qwen/Qwen3-Embedding-0.6B` is used to encode candidate samples and remove examples with high cosine similarity to benchmark problems from HumanEval, MBPP, CRUXEval, and LiveCodeBench. Here the decontamination mechanism itself introduces a model dependency: the embedding model determines which data is retained or dropped.
- **LLM360 MegaMath ships a per-benchmark decontamination receipt.** `LLM360/MegaMath-Llama-3.2-1B` and `LLM360/MegaMath-Llama-3.2-3B` each record 11 distinct `decontaminated_against` edges: ASDiv, MATH, MathQA, MAWPS, MMLU-STEM, OCWCourses, SAT-Math, SVAMP, AIME 2024, AIME 2025, and AMC. These edges are anchored to released decontamination artifacts such as `utils/decont_utils/data/{benchmark}.jsonl`; where specified, the exclusion rule operates over question/answer fields using n-gram overlap.
- **Nemotron-Cascade SWE-SFT excludes repositories that appear in SWE-Bench Verified.** Where a typical model card simply claims "decontamination was applied," the Nemotron-Cascade dataset card records the operational definition: SWE SFT instances whose source repository appears in `princeton-nlp/SWE-bench_Verified` are dropped before SFT. This repository-level exclusion is stronger than typical text-level overlap removal.

E.3 License and Release Hygiene

- **Tulu 3 DPO data is generated by a wide cross-organization teacher panel.** The Tulu 3 DPO mixtures include outputs from many upstream model families, including OpenAI, Anthropic, Mistral/Mixtral, 01-ai/Yi, MosaicML/MPT, and InternLM-family models. This makes DPO data a point where many license and terms-of-service regimes may accumulate, even when no single model card enumerates the full teacher panel.
- **Olmo 3 releases both complete and redacted Dolma 3 variants, and its reproduction data changed after training.** Dolma 3 includes complete and redacted variants of the academic-

document portion of the mix, including an olmOCR science-PDF slice. The reproduction dataset for Olmo-3-7B-1025 was later modified by replacing some redacted PDFs with `[REMOVED]`, which affects exact reproducibility. The graph surfaces this as a release-lineage issue rather than an isolated dataset-card note.

E.4 Code and Configuration Reveal Details Hidden by Cards

- **Olmo 3 DPO scripts reveal which models generated preference pairs.** The Olmo 3 paper describes DPO data at a high level, but the 32B-Instruct DPO training script names synthetic-pair datasets whose filenames encode their upstream generators and construction steps, including GPT-3.5 (OpenAI, 2022)/GPT-4o (OpenAI, 2024a) preference-pair sources, multi-turn truncation, deduplication, and topic filtering.
- **Per-stage placeholder percentages can be quantified by joining YAML and data-prep scripts.** Per-edge descriptions in the recovered graph quantify placeholder contributions: in `nvidia/Nemotron-RL-Super-Training-Blends`, DAPO-Math-17k contributes 1.36% of the rlvr1 mix; Skywork-OR1-RL-Data contributes 5.44% of the same mix; and the same DAPO-Math-17k contributes 0.10% to the Nano blend. These percentages are recoverable only by joining YAML mix-weight definitions with the upstream BytedTsinghua-SIA / Skywork dataset cards plus the `fill_placeholders.py` data-prep script.
- **SmolLM3 YAMLs reveal exact pretraining mixture details behind the card summary.** The SmolLM3 card summarizes pretraining with broad categories such as web, code, math, and multilingual data. The stage YAMLs instead pin individual per-language shards, including FineWeb2-HQ slices and Stack-Edu language buckets such as Python, Java, and Rust. Recovered SmolLM3 training edges also include exact mix weights.

E.5 Methodology and Ecosystem-Level Dependencies

- **SmolLM3 inherits methodology, not just data, from upstream models.** Two recovered `inspired_by` edges capture method transfer: SmolLM3 uses intra-document attention masking similar to Llama 3, and FineMath follows Qwen2.5-Math (Yang et al., 2024b)'s 13-gram decontamination procedure. These examples illustrate a dependency class that is neither weight inheritance nor training-data reuse, but still shapes downstream model development.
- **A few model families dominate generator and judge roles.** Among generator, filter, transformer, and judge edges, the most-used upstream models include Qwen2.5-32B-Instruct (70 uses), DeepSeek-R1 (43), Llama-3.3-70B-Instruct (36), Qwen3-32B (33), and GPT-4.1 (32). This shows that synthetic-data construction concentrates around a small number of high-use upstream models, especially in post-training stages such as DPO, RLHF, and RLVR.

Limitations

Our results should be interpreted in light of several limitations. First, ModSleuth reconstructs dependencies from publicly available artifacts and therefore cannot recover dependencies that are undocumented, proprietary, or otherwise inaccessible. As a result, the graphs we produce represent an evidence-grounded lower bound on the true dependency structure, with the gap likely largest for closed or partially disclosed releases.

Second, complete ground-truth dependency graphs do not exist, making absolute recall difficult to measure. Our evaluation therefore focuses on four well-documented LLM releases for which extensive supporting artifacts are available. While these models provide a challenging and realistic testbed, future work is needed to assess coverage across less thoroughly documented ecosystems.

Third, both ModSleuth and the primary automated verifier rely on Claude models. Our human audit and cross-model verification provide additional evidence supporting accepted edges, but neither establishes complete correctness or rules out shared failure modes. Reliance on closed models also limits reproducibility, as their behavior may change across versions.

Finally, our comparisons focus on fully automated baselines. We do not evaluate settings in which expert users iteratively guide a general-purpose agent through the dependency-tracing process. Such human-in-the-loop workflows could reduce the performance gap and would help disentangle the contribution of ModSleuth's task decomposition from that of the underlying agent capabilities.