Agents for Temporal Grounding

Elliot Epstein

February 6, 2026

Abstract

We study temporal grounding, the task of predicting the earliest year when a text sample could be produced without relying on unavailable knowledge. We introduce a multi-agent pipeline that extracts time-anchored entities, grounds them with web evidence, and aggregates conservative year estimates. We introduce an evaluation suite based on human-annotated samples from Tulu-3 to measure the efficiency of the approach and compare against several baselines. We then demonstrate scalability by applying the pipeline to a large subset of the Tulu-3 SFT, RLVR, and DPO datasets.

1 Introduction

Given a text sample, a natural high-level problem is to assign a year label that reflects its temporal compatibility, which depends on the downstream application. Such labels can be used for dataset auditing, evaluation, and data selection.

In this paper we focus on the year-assignment task needed to bucket training data for an LLM whose knowledge is restricted to a cutoff year τ (e.g., backtesting NLP-based models and what-if scenario analysis under historical knowledge constraints). Supporting such models requires time-bucketed data so that a model trained through year τ only sees samples that are admissible at τ.

Key considerations for year bucketing. We highlight three key considerations.

1. Explicit facts should become available on time. If an explicit fact became publicly knowable in year y (e.g., a product launch), then a model trained through year y should be able to state it.

2. Implicit social knowledge should match contemporaneous public understanding. For example, consider the implicit statement "There has been a lot of talk about global warming on the news." Media coverage and public salience increased over time, so it is hard to pinpoint the earliest year when that sentence could have been said. If a model trained only through the 1970s were highly knowledgeable and concerned about climate change in a way that matches much later discourse, this would be viewed as temporal leakage.

3. Explicit and implicit cues correspond to different temporal notions. For explicit entities, the earliest admissible time is often close to a single year. For implicit entities, admissibility is ambiguous, so the earliest admissible time is naturally distributed over a range. These distinctions align with different notions of time: *event time* (when a fact becomes true), *epistemic availability time* (when a well-informed observer could know it), *authorship or publication time* (when the text was written), and *stylistic or discourse time* (when the language and norms are typical). For downstream applications, *epistemic availability time* is

the best-aligned notion: an event may be known to insiders months or years before it becomes public, but a useful time-bounded LLM should not depend on insider knowledge. Likewise, *authorship or publication time* can be misleading: a contemporary expert can write text that is fully consistent with 19th-century knowledge and style, in which case the relevant label is when the text could have been written rather than when it actually was.

Task. Let x be a text sample. Let $P_{x,\mathrm{gt}}(t)$ denote the (latent) distribution over earliest times t such that the text could be produced without relying on unavailable or speculative knowledge at time t, as judged by a well-informed observer. Given x, predict the α-quantile of $P_{x,\mathrm{gt}}$:

$$t_\alpha(x) \;=\; \inf\Big\{t \;:\; \Pr_{T\sim P_{x,\mathrm{gt}}}\big(T \le t\big) \ge \alpha\Big\}.$$

Interpretation. A fraction α of informed observers would judge time $t_\alpha(x)$ (or earlier) to be admissible. Times earlier than $t_\alpha(x)$ lie in the premature tail.

Temporal semantics and probabilistic formulation. Document dating methods implicitly target authorship time by estimating $\arg\max_t p(T = t \mid x)$. This is a different goal from ours: we care about which knowledge cutoff makes the text admissible, not when it was authored. Our setting is fully digital, so classical forensic signals (e.g., ink age, paper composition, and physical provenance) are unavailable, and we emphasize scalability for large corpora. In contrast, training time-bounded LLMs requires reasoning about admissibility: for which times a text can be produced without relying on future or speculative knowledge.

Formally, let $p_x(t)$ denote a latent distribution over admissible times for a text x, reflecting epistemic uncertainty rather than authorship likelihood. One option is to collapse this distribution to an earliest admissible time

$$t_{\min}(x) \;=\; \inf\Big\{t \;:\; \Pr_{T\sim p_x}\big(T \le t\big) > 0\Big\},$$

yielding a conservative lower bound that enforces strict temporal monotonicity and minimizes leakage. However, this approach discards uncertainty and tends to assign temporally ambiguous or timeless text to early years. An alternative is to treat x as compatible with a distribution over admissible times, including it in training for a model at time t with weight proportional to $\Pr_{T\sim p_x}(T \le t)$. This preserves uncertainty and yields smoother knowledge accumulation, at the cost of weaker worst-case leakage guarantees. These two formulations reflect different probabilistic assumptions (pointwise lower bounds versus cumulative compatibility) and should be chosen explicitly based on the intended use of the model.

Applications. Synchronos LLMs are models trained on data restricted to specific year ranges. Potential applications include finance backtesting to avoid look-ahead bias, historical analysis with time-bounded knowledge, auditing model behavior under controlled knowledge cutoffs, and reproducible evaluations that fix the temporal scope of training data.

2 Method

Entity types. We group entities into three categories: explicit, implicit, and timeless. We define *explicit* entities as those with a single right admissible year under our notion of epistemic availability (e.g., a concrete product launch year), so their admissible-time uncertainty is negligible at year

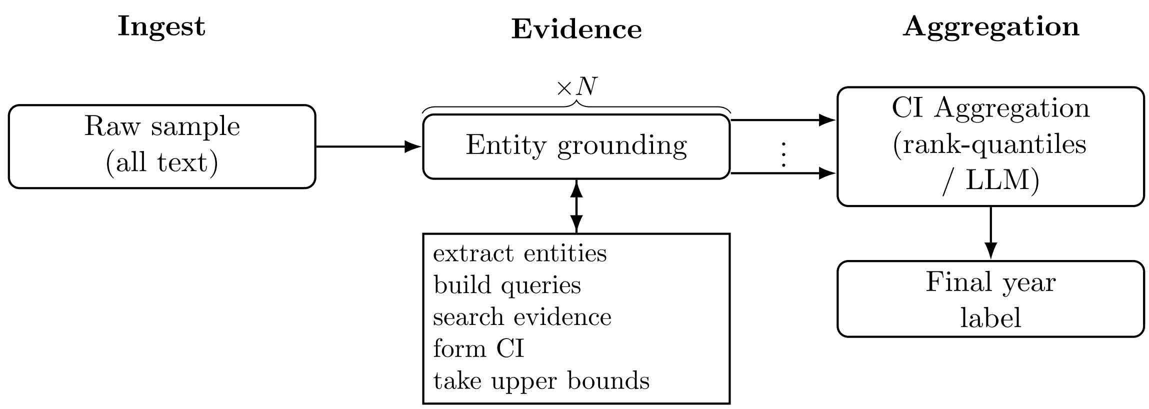


Figure 1: Filtering pipeline for a single sample. Evidence extraction and CI aggregation are repeated for N samples, then merged into a final year label.

granularity and can be treated as a point mass. For example, "I just bought the first iPhone that was released last week" is explicit: the admissible time concentrates on the release year of the first iPhone, so $P_{x,\mathrm{gt}}(t)$ is effectively a delta mass on that year. We define *implicit* entities as those without a single right year, where admissibility depends on diffuse public salience and can vary across observers. For example, "There has been a lot of talk about global warming on the news" is implicit: coverage has intensified over decades, so it is difficult to pinpoint the earliest year when the statement becomes true. As a result, the earliest-admissible time is itself distributed, and our goal is to return a sample from that distribution. Finally, *timeless* samples contain no time-anchored entity. We do not use a database agent because there is no internal database to cross-reference in this pipeline.

Operational pipeline. Figure 1 summarizes our end-to-end implementation. We operationalize the method as a five-step agent pipeline:

1. Entity Extraction Agent: extract time-anchored entities from the question and answer bundle.
2. Reasoner Agent: assign a best-estimate year and a 95% confidence interval per entity.
3. Search Query Agent: generate standalone search queries for each entity.
4. Internet Search Agent: retrieve evidence from search results to ground the estimates.
5. Reasoner/Synthesizer Agent: update entity confidence intervals based on the evidence and update the overall estimated year using the maximum upper bound across entities.

Why not GLiNER. We evaluated GLiNER for entity extraction, but it is a poor fit for this task. Given fixed categories, it must both detect the entity span and map it to a category, and empirically it misses entities frequently in our data. Because missed entities directly induce leakage, we instead use LLMs for the full stack (entity extraction through year estimation), with search-based evidence grounding to improve reliability.

3 Related Work

We study a reliability problem that arises in both benchmark construction and downstream deployment: for a natural-language item (question/claim/prompt/answer bundle), what is the *earliest time* at which the item could be produced (or answered) without implicitly using information that only appears later? This section situates our approach at the intersection of (i) methods for tracing *effective* knowledge cutoffs and temporal alignment in LLMs, (ii) time-sensitive QA and temporal reasoning benchmarks, (iii) dynamic evaluation for factual drift/outdatedness, (iv) time-aware modeling (conditioning or routing by time), and (v) leakage/contamination detection and mitigation. While these areas are individually mature, general-purpose *sample-level* dating procedures for arbitrary natural-language items remain comparatively less standardized.

Tracing cutoffs and temporal validity in LLM evaluation. A close conceptual neighbor is Cheng et al. [2024], which defines *resource-level effective cutoffs* and estimates them by probing model perplexity over time-versioned datasets (e.g., monthly Wikipedia snapshots and news buckets). They show that "knowledge cutoffs" are rarely a single clean date: effective cutoffs vary by source and can be blurred by crawl artifacts and near-duplicate content. This motivates item-level temporal controls, since a single global cutoff can be too coarse for evaluation hygiene.

Temporal validity can also break through the retrieval layer. In retrospective settings, practitioners often rely on search-engine date filters to restrict evidence to pre-cutoff time windows. Lahib et al. [2026] audit this practice and show that it can still leak future information even when retrieval appears date-restricted. This supports the need for item-level evidence validation when temporal causality matters.

Temporal generalization analyses broaden the picture: models can be systematically biased across past/present/future contexts. Zhu et al. [2025] propose temporal generalization as an evaluation axis and provide evidence of temporal biases and performance decay. Their perspective reinforces that time-indexed evaluation suites should be treated as a first-class design choice rather than an afterthought.

Time-aware modeling. Beyond evaluation, several works treat time as an explicit modeling variable. Dhingra et al. [2022] interpret language models as temporal knowledge bases and study time-varying facts, proposing timestamp conditioning to better represent and retrieve temporally scoped knowledge. Such mechanisms reduce ambiguity about "what was true when" and provide a natural sink for timestamped supervision derived from curated data.

A complementary architectural approach is temporal routing. Faro et al. [2025] train experts on disjoint temporal slices and mask experts whose training windows end after the query time, enforcing a strict "no future knowledge" constraint at inference time. Relative to our approach, which focuses on assigning time constraints to arbitrary items and using them to filter/stratify data, time-routed models directly encode temporal constraints in the model architecture. Together, they offer a path to both (i) temporally consistent data and (ii) temporally constrained inference.

Time-sensitive QA and temporal reasoning benchmarks. Benchmark families study temporality from different angles: time-sensitive *facts* (answers change over time), temporal *context* (answers depend on when/where asked), and temporal *reasoning* (ordering and temporal relations in text). TimeQA focuses on time-sensitive factual questions grounded in evolving Wikidata facts and aligned evidence sources [Chen et al., 2021]. SituatedQA analyzes questions whose answers depend on temporal and geographic context, showing that systems can fail when context is implicit or mismatched [Zhang and Choi, 2021]. TORQUE targets temporal reasoning about event ordering in passages (before/after/during) via a reading-comprehension format [Ning et al., 2020]. Collectively, these datasets motivate making temporal scope explicit, since evaluation can be confounded when the temporal frame is implicit or inconsistent.

A core difficulty in time-sensitive QA is disentangling genuine temporal reasoning from memorized factual recall. UnSeenTimeQA addresses this by constructing contamination-resistant scenarios intended to reduce web-searchable memorization [Uddin et al., 2025]. TDBench proposes a systematic evaluation of factual time-sensitive QA using temporal databases [authors, 2026]. In contrast to work that assumes timestamps are known by construction (e.g., from structured sources), our setting targets arbitrary natural-language items whose temporal requirements must be inferred from content and evidence.

Dynamic evaluation and outdatedness detection. Another stream evaluates time-sensitive knowledge by validating against continuously updated structured sources. DyKnow dynamically checks whether LLM outputs match current Wikidata values and measures outdatedness and inconsistency under prompt perturbations [Mousavi et al., 2024]. Dynamic benchmarking work similarly generates time-bucketed test sets from Wikidata and evaluates robustness to factual drift under multiple evaluation "views" [Margatina et al., 2023]. These methods are well-suited to entity-centric facts with structured ground truth; evidence-grounded dating aims to generalize the same instinct to open-ended claims and questions that do not map cleanly onto a knowledge graph.

Contamination, leakage, and decontamination methods. Temporal grounding is also motivated by the contamination literature, which shows that benchmark scores can be inflated by overlap between training and evaluation data. Golchin and Surdeanu [2024] present black-box strategies for tracing contamination, while Golchin and Surdeanu [2025] introduce a quiz-based tool to detect and estimate contamination without training-data access. Li and Flanigan [2024] highlight task-level contamination effects by showing that models often perform markedly better on datasets released before their pretraining collection windows than on datasets released after, complicating "few-shot" claims.

Mitigation approaches include rewriting and filtering evaluation items. CLEAN–EVAL creates cleaner benchmark variants via transformations (e.g., paraphrasing/translation) and semantic filtering to reduce memorization-based score inflation [Zhu et al., 2024b]. Inference-Time Decontamination (ITD) combines detection with rewriting during evaluation to "revive" leaked benchmarks while preserving difficulty [Zhu et al., 2024a]. These strategies are largely orthogonal to temporal dating: rewriting targets surface-form memorization, whereas dating targets causal validity and future-only information. In practice, they can be combined: a time-dated benchmark can also be rewritten to stress-test generalization.

Finally, pretraining-data detection methods offer another complementary tool family. Shi et al. [2024] introduce benchmarks like WIKIMIA and the Min-K% Prob detector to infer whether text was included in pretraining from black-box probabilities. While temporal dating aims to infer the *earliest* plausible time a text could be written, pretraining-data detection aims to infer whether a model has *seen* the text; together, they help diagnose whether strong performance stems from reasoning under appropriate temporal constraints or from memorization/leakage.

Synthesis and remaining gaps. Across these strands, a consistent message emerges: effective cutoffs vary by resource [Cheng et al., 2024], date-filtered retrieval can leak [Lahib et al., 2026], temporal biases exist [Zhu et al., 2025], and contamination can systematically inflate scores [Li and

Flanigan, 2024]. What remains less standardized is a general-purpose, sample-level mechanism that assigns an earliest admissible time to arbitrary natural-language items using evidence and reasoning, and that is convenient enough to serve as a dataset filtering primitive and evaluation control.

4 Evaluation

Annotation protocol. For each sample, the annotator read the question and answer, extracted relevant entities (omitting entities that are clearly older than the dominant ones), and searched to find the concept date for each entity. For each entity, we include a source link to the article stating the concept date unless the fact is trivially known. The full dev-set annotation pass took about 2 hours, or roughly 3.5 minutes per question.

Dev set. We build a 35-question dev set sampled from Tulu-3 and labeled by human annotators to design and iterate on the prompting scheme, entity grounding, and aggregation rules. We use the dev set to refine the prompting scheme and aggregation rules; test-set results are reported separately once all plots are finalized.

Test set. We prepared a held-out test set of 141 samples from Tulu-3. The test set is constructed and annotated in the same way as the dev set, with human annotators providing gold years and source links.

5 Results

Key plots for this analysis are Figures 2 and 3. Figure 2 shows that search grounding improves no-leak accuracy by roughly 10 points across models, with Gemini models reaching 100% no-leak accuracy on the dev set. Figure 3 shows the distribution of predicted year minus gold year, where the most common failure is entity extraction and search grounding appears reliable. We categorize error types using an LLM with access to the ground-truth answer. Figure 4 shows the search-aided prediction error (predicted year minus gold year) by model on the test set, with negative deltas colored by failure type. In many cases the difference reflects the LLM finding later references than the human annotator; this should improve once multiple annotators review each example. Without gold labels, we estimate model conservatism by counting how often each model provides the maximum year for the same prompt.

Other performance metrics. Possible options include an asymmetric error loss averaged over samples, $L = \max(0, -e) + \beta \cdot \max(0, e)$ where $e = \hat{y} - y$, $\hat{y}$ is the predicted year, and y is the gold year. This penalizes underestimates more than overestimates when $\beta < 1$, but it is harder to interpret and requires choosing β. Another option is an equally weighted average of exact year match and no-leak accuracy, which is easy to interpret but does not distinguish between off-by-one errors and large errors.

6 Discussion

Two observations stand out: Gemini 3 Flash performs better than Gemini 3 Pro, and GPT-5-mini is similar to GPT-5.2. It would be valuable to explore whether this stems from smaller models producing more conservative intervals and from the asymmetric risk profile that penalizes underestimates more than overestimates.

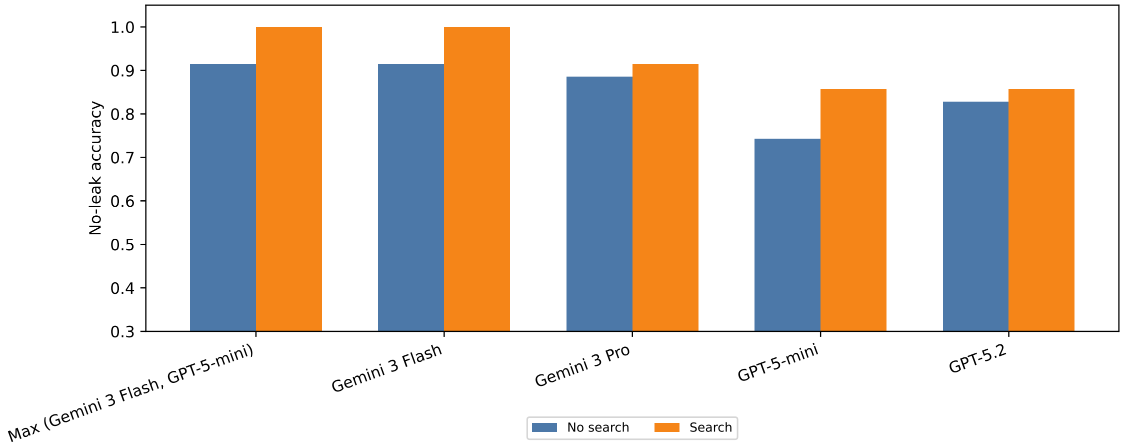


Figure 2: No-leak accuracy (predicted year $\geq$ gold) before vs. after search grounding on the dev set ($N = 35$). Values are proportions without uncertainty bars because each point aggregates the full dev set.

7 Data Filtering

We label each supervised (SFT), preference, and RLVR sample with the minimum calendar year consistent with its question–answer bundle. The prompt structure is detailed in Appendix B. We considered deterministic filtering, but it was difficult to capture all edge cases with a rule-based approach. The latest sweep (session `2026-01-06_14-10PT`) processed 30,549 SFT examples, 29,510 preference examples, and three RLVR datasets (7,358 GSM, 7,372 MATH, 14,958 IFEval) with a conservative policy that uses the most recent referenced year. Figures 5 to 7 show year and category distributions for each dataset family and validate cutoff integrity.

A Model Training on Filtered Data

A.1 SFT setup

We fine-tune Qwen3-4B-Base with LoRA adapters on the 2007-capped TÜLU-3 SFT subset (26,431 examples) with a 4,096-token context length. Training runs for 2 epochs with linear decay, short warmup, and small per-device batches with gradient accumulation. Tokens per second per GPU are around 1,000. The LoRA run takes about 3 hours 2 minutes for 300M tokens (padding included). LoRA uses roughly 45 GB of GPU memory; full fine-tuning is about 10× slower, and TÜLU-2 showed lower LoRA performance.

Hardware. All runs use three NVIDIA RTX A6000 GPUs (48 GB each). The LoRA configuration fits within a single A6000 with limited headroom for data loading and logging.

A.2 SFT results

We run the TÜLU-3 dev evaluation suite, dispatching 11 suites and limiting each task to 100 examples (MMLU uses 100 questions per subject, totaling 5,700 evaluations). This gives a fast signal across reasoning, coding, alignment, and factuality. Task summaries: GSM8K (grade-school math word problems), DROP (reading comprehension with numeric reasoning), Minerva Math (competition

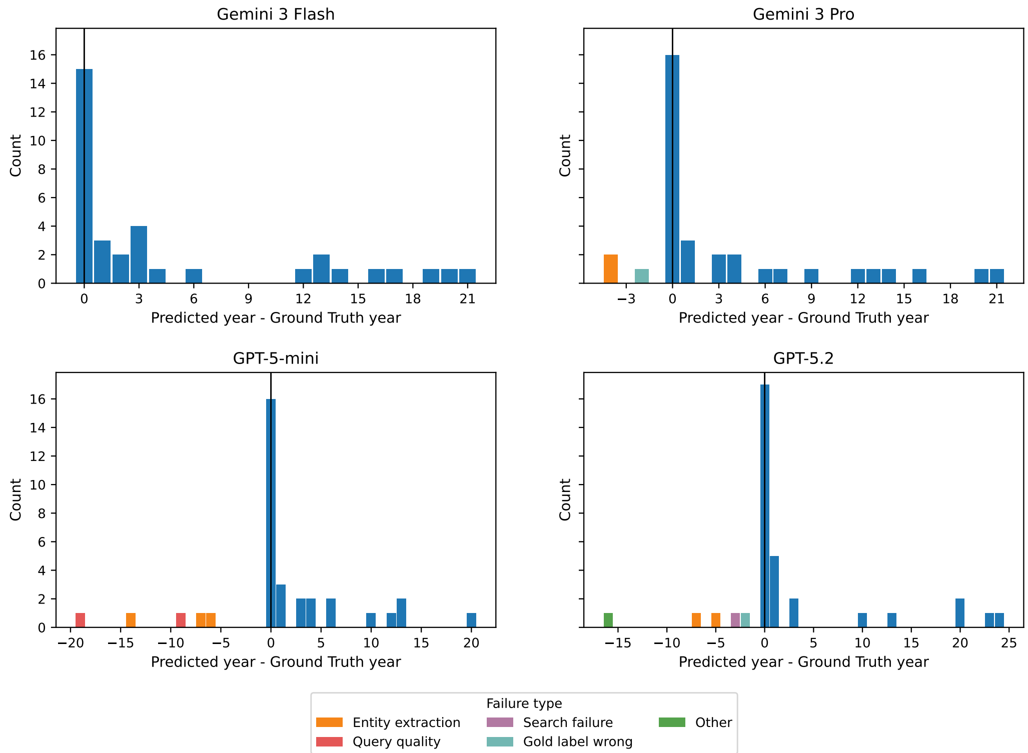


Figure 3: Search-aided prediction error (predicted year minus gold year) by model on the dev set ($N = 35$), with negative deltas colored by failure type. Panels correspond to models.

math problems across seven domains), HumanEval/HumanEval+ (Python coding pass@10), IFEval (instruction-following), PopQA (entity-centric factual QA), MMLU (multiple-choice knowledge across 57 subjects), AlpacaEval v2 (pairwise preference wins), BBH (hard reasoning tasks with CoT), TruthfulQA (robustness to falsehoods).

Table 2 summarizes the latest snapshot for Qwen3-4B-Base and placeholder columns for +SFT, +DPO, and +RLVR checkpoints (marked "–"). The SFT evaluation is still running; partial results are shown where available. Scores are percentages, with n denoting evaluated examples.

Filtering cost. The current filtering pass uses GPT-5-mini with batch requests at \$0.25/\$2.00 per 1M input/output tokens; the batch discount halves these rates to \$0.125/\$1.00. Table 3 summarizes per-sample token averages, current costs, and projections for the full SFT/preference corpus plus the RLVR targets. TÜLU-3 still requires filtering 900,000 SFT examples and 250,000 preference examples beyond the current subset; projected costs scale linearly with per-sample token counts. For prompts under 200k tokens, Gemini 3 Flash is priced at \$0.50/\$3.00 per 1M input/output tokens (similar to GPT-5-mini), while Gemini 3 Pro is \$2.00/\$12.00 (similar to GPT-5.2).

Downstream, we select shards with a year-bounded loader to enforce knowledge cutoffs (e.g., 2014).

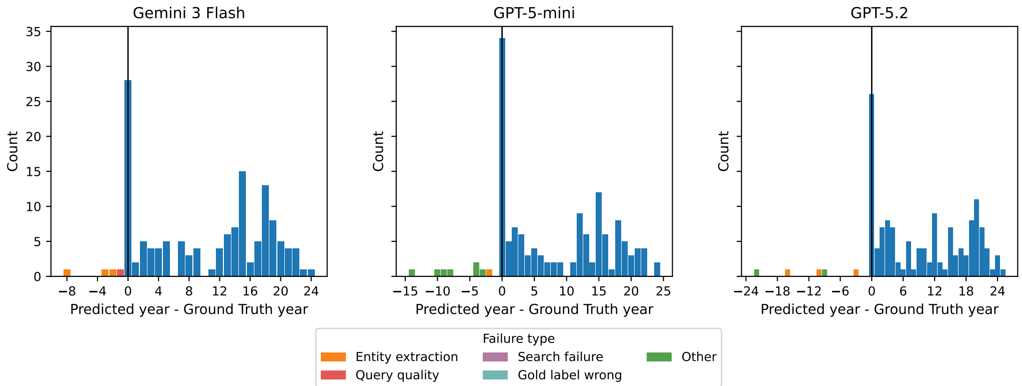
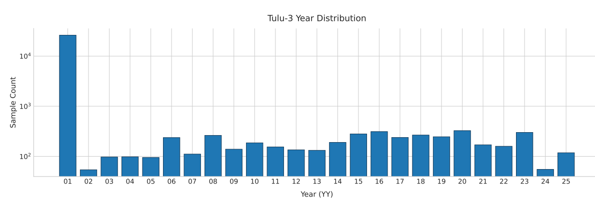
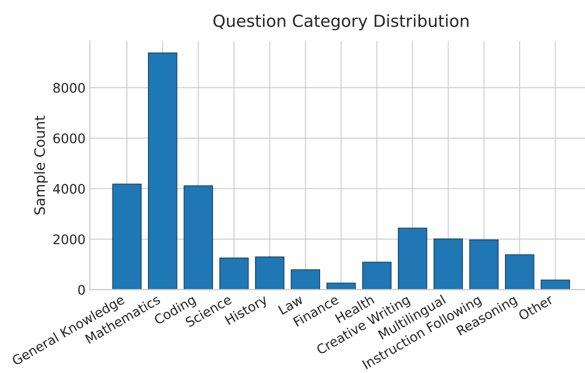


Figure 4: Search-aided prediction error (predicted year minus gold year) by model on the test set ($N = 141$), with negative deltas colored by failure type. Panels correspond to models.



(a) Year distribution (SFT).



(b) Category distribution (SFT).

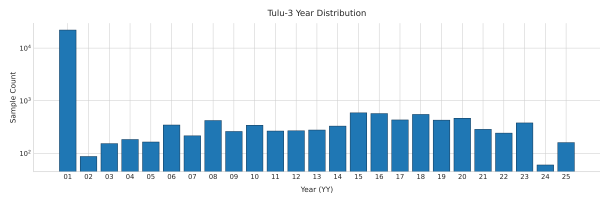
Figure 5: Filtering summary for the TÜLU-3 SFT mixture (session `2026-01-06_14-10PT`, $n =$ 30,549). Histograms are computed from the filtered sample set.

B SFT Filtering Prompt

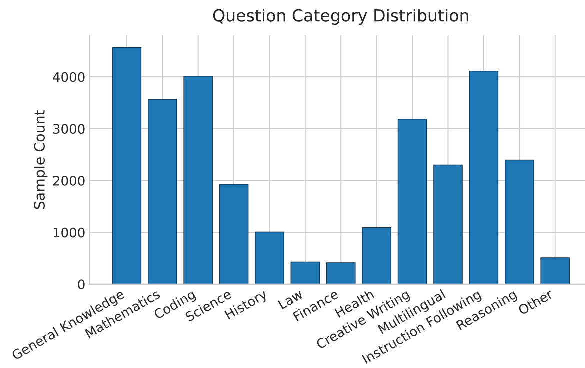
```
You label the minimum calendar year (between 2001 and 2025) required to answer
a question without temporal leakage. The label must never precede any fact
mentioned in the sample; when uncertain, err toward the later year so that no
future knowledge sneaks into earlier buckets.

You receive a dataset-specific question plus an answer bundle (which may contain
multiple sections).
These are supervised instruction-tuning pairs: treat the question as the user
prompt and the response as the assistant answer.
Pick the smallest year Y in [2001, 2025] so that a model with knowledge through
year Y could answer confidently, considering EVERYTHING in both the question and
the answer bundle. If no specific time-dependent knowledge is required, output
2001.

Rules:
```



(a) Year distribution (DPO).



(b) Category distribution (DPO).

Figure 6: Filtering summary for the TÜLU-3 preference mixture (session `2026-01-06_14-10PT`, $n = 29{,}510$). Histograms are computed from the filtered sample set.

Component	Setting
Base model	Qwen3-4B-Base
Tokenizer	Qwen3-4B-Base tokenizer
Training data	TÜLU-3 SFT, year $\leq$ 2007 (26,431 examples)
Sequence length	4,096 tokens
Batch size	1 sample per device
Gradient accumulation	16 steps (effective 16 samples per device)
Optimizer schedule	Linear decay with 3% warmup
Learning rate	1×10^{-4}
Weight decay	0.0
Epochs	2
LoRA configuration	rank 64, $\alpha = 16$, dropout 0.1
Memory optimizations	Flash attention and gradient checkpointing
Checkpoint cadence	Every 500 steps (keep last 3)

Table 1: Supervised fine-tuning configuration for the LoRA run.

```
- Identify all time-anchored entities in the question and answer bundle.
- For each entity, provide a best_estimate year plus a 95% confidence interval.
- Use the entity's founding/release/announcement year (not the future target year).
- Set overall "year" to the maximum upper bound across all entity confidence intervals.
- If a range is mentioned (e.g., "released between 2008 and 2015"), use that as
  the entity's interval.
- If information is older than 2001, still respond with 2001.
- Do not hallucinate years; use only dates grounded in the text or well-known facts.
- Additionally, assign the question to one category from this list:
  general_knowledge, math, coding, science, history, law, finance, health,
  creative_writing, multi_lingual, instruction_following, reasoning, other.

You may reason internally, but the final output must be a single JSON object only.
Do not include any extra text or code fences.
```

Task	Metric	Qwen3-4B Base	+SFT	+DPO	+RLVR	n
GSM8K	Exact match	83.00	83.00	–	–	100
DROP	F1	52.15	57.16	–	–	100
Minerva Math (avg)	Exact match	39.14	–	–	–	700
HumanEval	pass@10	95.84	97.30	–	–	100
HumanEval+	pass@10	94.80	93.28	–	–	100
IFEval	Prompt loose acc	40.00	43.00	–	–	100
PopQA	Accuracy	17.00	20.00	–	–	100
MMLU (mc)	Macro accuracy	74.46	74.44	–	–	5,700
AlpacaEval v2	Len-ctrl win rate	6.54	–	–	–	100
BBH (cot-v1)	Macro accuracy	–	–	–	–	–
TruthfulQA	MC2	54.48	45.59	–	–	100

Table 2: Primary metrics for the TÜLU-3 dev suite (Qwen3-4B-Base, 100-example subsets). The BBH run is still executing at this scale; results will be inserted once the evaluation completes.

Dataset	Current n	Tokens/sample	Current cost	Projected n	GPT-5-mini	GPT-5.2	GPT-5.2 Pro
SFT	30,549	1,661	$20.46	930,549	$623	$4.36k	$52.3k
Preference	29,510	2,437	$25.08	279,510	$238	$1.66k	$20.0k
RLVR GSM	7,358	2,167	$6.78	8,790	$8.10	$56.7	$680
RLVR MATH	7,372	1,653	$3.68	7,500	$3.74	$26.2	$315
RLVR IFEval	14,958	1,690	$10.81	15,000	$10.84	$75.9	$910

Table 3: Filtering costs (USD) under batch pricing. Projected counts use 900,000 and 250,000 additional SFT/preference samples and RLVR targets of 8.79k (GSM), 7.5k (MATH), and 15k (IFeval).

```
Return JSON with these required fields and meanings:
- "year": integer in [2001, 2025] for the minimum safe year.
- "confidence": "low" | "medium" | "high".
- "category": one of the allowed categories listed above.
- "justification": short reason for the chosen year.
- "entities": object mapping entity names to an object with:
  - "best_estimate": best estimate year for founding/release/announcement.
  - "confidence_interval_95": [yearA, yearB] containing the best estimate; yearA/yearB
    can be the same.
  - "search_query": a standalone query to verify the year estimate.
- If no entities are found, use an empty object for "entities".

Illustrative example (output only):
{"year": 2006, "confidence": "high", "category": "general_knowledge",
"justification": "Answer references tweets, a concept only available after
Twitter launched in 2006, so 2006 is the earliest safe year.",
"entities": {"tweet": {"best_estimate": 2006, "confidence_interval_95": [2006, 2006],
"search_query": "When was Twitter launched?"}}}
```

Dataset	Prompts	License
CoCoNot	10,983	ODC-BY-1.0
FLAN v2 (ai2-adapt-dev/flan_v2_converted)	89,982	–
No Robots	9,500	CC-BY-NC-4.0
OpenAssistant Guanaco	7,132	Apache 2.0
Tulu 3 Persona MATH	149,960	ODC-BY-1.0
Tulu 3 Persona GSM	49,980	ODC-BY-1.0
Tulu 3 Persona Python	34,999	ODC-BY-1.0
Tulu 3 Persona Algebra	20,000	ODC-BY-1.0
Tulu 3 Persona IF	29,980	ODC-BY-1.0
NuminaMath-TIR	64,312	Apache 2.0
Tulu 3 WildGuardMix	50,000	Apache 2.0
Tulu 3 WildJailbreak	50,000	ODC-BY-1.0
Tulu 3 Hardcoded	240	CC-BY-4.0
Aya	100,000	Apache 2.0
WildChat GPT-4	100,000	ODC-BY-1.0
TableGPT	5,000	MIT
SciRIFF	10,000	ODC-BY-1.0
Evol CodeAlpaca	107,276	Apache 2.0

Table 4: TÜLU 3 SFT mixture composition. Source details: Brahman et al. (2024), Longpre et al. (2023), Rajani et al. (2023), Kopf et al. (2024), Beeching et al. (2024), Han et al. (2024), Wildteaming (2024), Singh et al. (2024), Zhao et al. (2024), Zha et al. (2023), Wadden et al. (2024), Luo et al. (2023).

```
<question>
{sample.question}
</question>
<answer_bundle>
{sample.answer}
</answer_bundle>
Return JSON exactly in this schema:
{"year": 2001, "confidence": "low|medium|high",
"category": "one of the allowed categories",
"justification": "why year is required",
"entities": {"entityA": {"best_estimate": 2008, "confidence_interval_95": [2007, 2009],
"search_query": "When was entityA released?"},
"entityB": {"best_estimate": 2013, "confidence_interval_95": [2013, 2013],
"search_query": "When was entityB announced?"}}}
```

C SFT Mixture Data

The TÜLU 3 SFT mixture used for training contains 939,344 samples from the sources below.

References

Anonymous authors. Harnessing temporal databases for systematic evaluation of factual time-sensitive question-answering in llms. In *Under review (ICLR 2026 submission on OpenReview)*,

2026. URL https://openreview.net/forum?id=W7RNxsTKKZ.

Wenhu Chen, Xinyi Wang, and William Yang Wang. A dataset for answering time-sensitive questions. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/1f0e3dad99908345f7439f8ffabdffc4-Abstract-round2.html, 2021. NeurIPS 2021 Datasets and Benchmarks.

Jeffrey Cheng, Marc Marone, Orion Weller, Dawn Lawrie, Daniel Khashabi, and Benjamin Van Durme. Dated data: Tracing knowledge cutoffs in large language models, 2024. URL https://arxiv.org/abs/2403.12958.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 2022. URL https://aclanthology.org/2022.tacl-1.15/.

Robin Faro, Dongyang Fan, Tamar Alphaidze, and Martin Jaggi. Timoe: Time-aware mixture of language experts, 2025. URL https://arxiv.org/abs/2508.08827.

Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models, 2024. URL https://arxiv.org/abs/2308.08493.

Shahriar Golchin and Mihai Surdeanu. Data contamination quiz: A tool to detect and estimate contamination in large language models, 2025. URL https://arxiv.org/abs/2311.06233.

Ali El Lahib, Ying-Jieh Xia, Zehan Li, Yuxuan Wang, and Xinyu Pi. Temporal leakage in search-engine date-filtered web retrieval: A case study from retrospective forecasting, 2026. URL https://www.arxiv.org/abs/2602.00758.

Changmao Li and Jeffrey Flanigan. Task contamination: Language models may not be few-shot anymore, 2024. URL https://arxiv.org/abs/2312.16337.

Katerina Margatina, Shuai Wang, Yogarshi Vyas, Neha Anna John, Yassine Benajiba, and Miguel Ballesteros. Dynamic benchmarking of masked language models on temporal concept drift with multiple views, 2023. URL https://arxiv.org/abs/2302.12297.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. Dyknow: Dynamically verifying time-sensitive factual knowledge in llms, 2024. URL https://arxiv.org/abs/2404.08700.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. Torque: A reading comprehension dataset of temporal ordering questions. In *EMNLP 2020*, 2020. URL https://aclanthology.org/2020.emnlp-main.88/.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models, 2024. URL https://arxiv.org/abs/2310.16789.

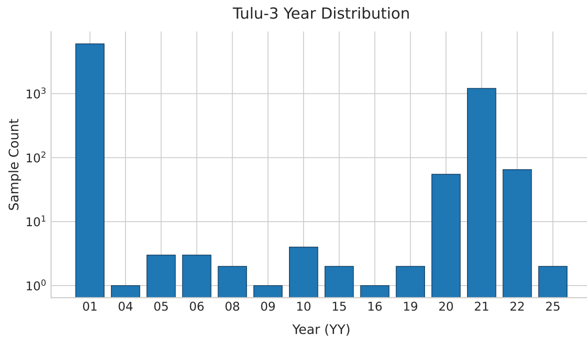
Md Nayem Uddin, Amir Saeidi, Divij Handa, Agastya Seth, Tran Cao Son, Eduardo Blanco, Steven R. Corman, and Chitta Baral. Unseentimeqa: Time-sensitive question-answering beyond llms' memorization. In *ACL 2025*, 2025. URL https://aclanthology.org/2025.acl-long.94/.

Michael J.Q. Zhang and Eunsol Choi. Situatedqa: Incorporating extra-linguistic contexts into qa. In *EMNLP 2021*, 2021. URL https://aclanthology.org/2021.emnlp-main.586/.

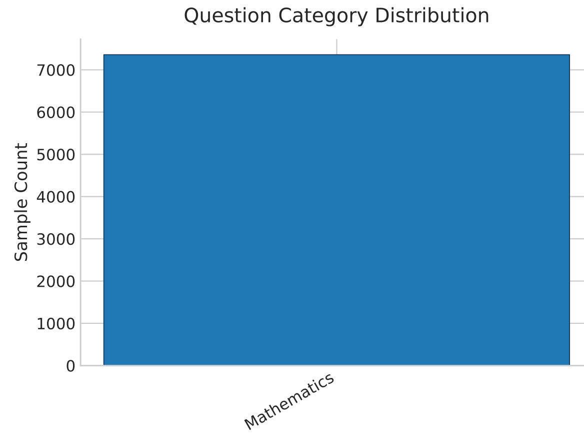
Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. Is your llm outdated? a deep look at temporal generalization. In *NAACL 2025 (Long Papers)*, 2025. URL https://aclanthology.org/2025.naacl-long.381/.

Qin Zhu, Qingyuan Cheng, Runyu Peng, Xiaonan Li, Tengxiao Liu, Ru Peng, Xipeng Qiu, and Xuanjing Huang. Inference-time decontamination: Reusing leaked benchmarks for large language model evaluation. In *Findings of EMNLP 2024*, 2024a. URL https://aclanthology.org/2024.findings-emnlp.532/.

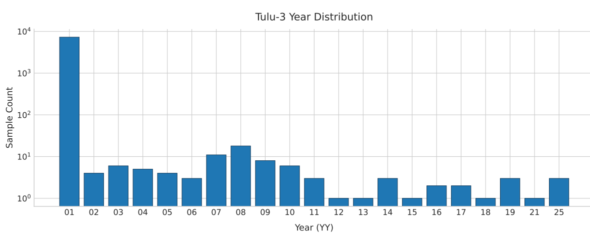
Wenhong Zhu, Hongkun Hao, Zhiwei He, Yunze Song, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. Clean–eval: Clean evaluation on contaminated large language models. In *Findings of NAACL 2024*, 2024b. URL https://aclanthology.org/2024.findings-naacl.53/.



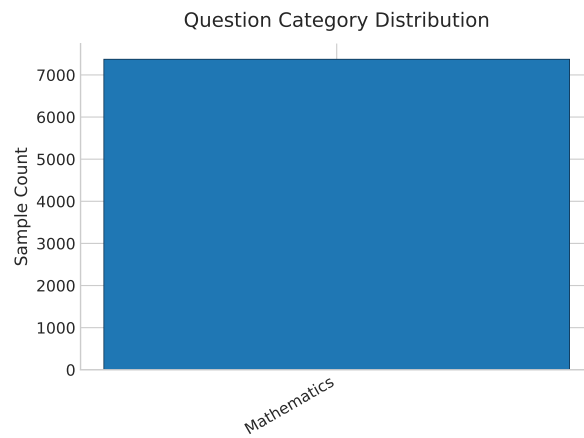
(a) Year distribution (RLVR-GSM).



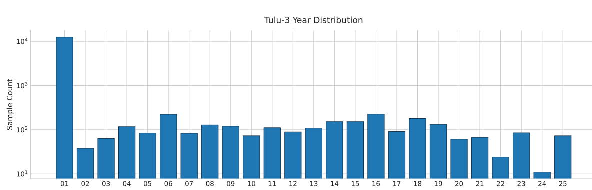
(b) Category distribution (RLVR-GSM).



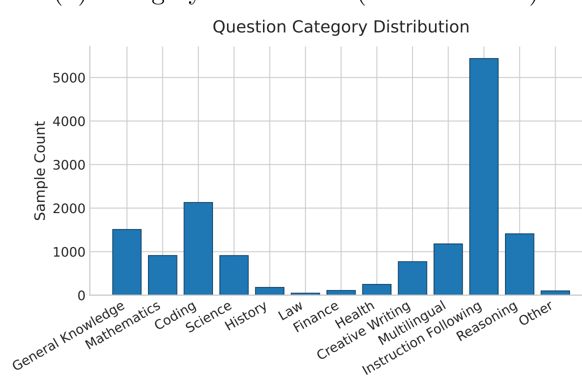
(c) Year distribution (RLVR-MATH).



(d) Category distribution (RLVR-MATH).



(e) Year distribution (RLVR-IFeval).



(f) Category distribution (RLVR-IFeval).

Figure 7: Filtering summary for the RLVR datasets (session `2026-01-06_14-10PT`, GSM $n = 7{,}358$, MATH $n = 7{,}372$, IFEval $n = 14{,}958$). Histograms are computed from the filtered sample sets.

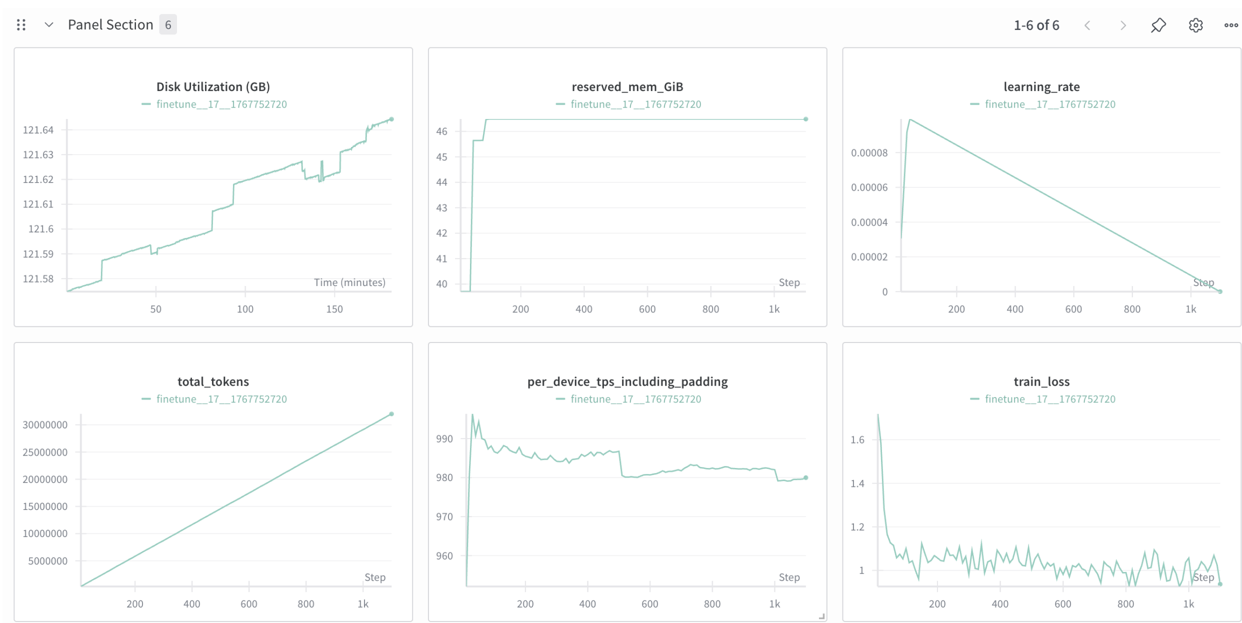


Figure 8: LoRA SFT training metrics: disk utilization (GB), reserved GPU memory (GB), learning rate, total tokens, per-device tokens per second, and train loss.