Agents for Temporal Grounding

Elliot Epstein

February 5, 2026

1 Introduction

We propose a methodology for assigning the earliest calendar year in which a question–answer bundle could likely have been written, and apply it to produce a large, time-grounded post-training dataset based on Tulu-3.

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 $P_{x,\mathrm{gt}}$:

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

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.

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

Figure 1 summarizes the end-to-end pipeline. The process follows 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/Synthetizer Agent: update entity confidence intervals based on the evidence and update the overall estimated year using the maximum upper bound across entities.

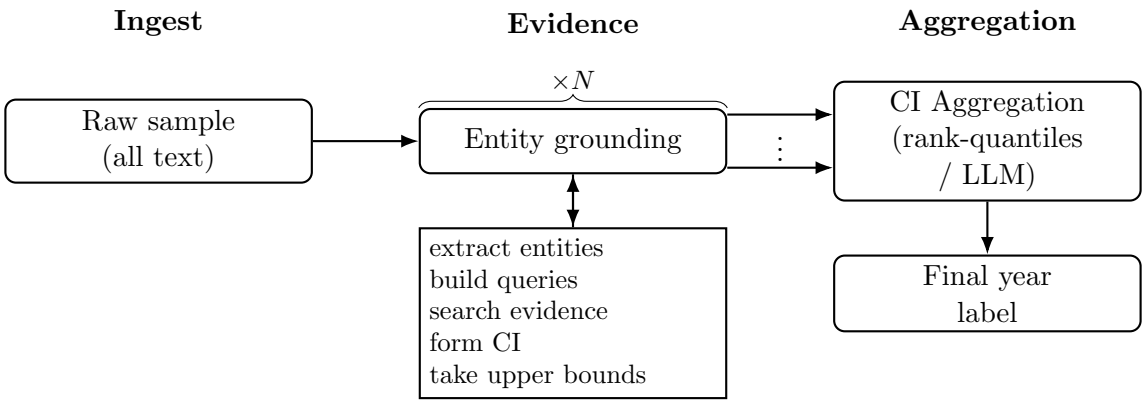


Figure 1: Filtering pipeline for a single sample. Evidence extraction and CI aggregation are repeated for N samples, then merged into a final year label.

Entity types. We group entities into three categories: explicit (an explicit date or event such as a product launch), implicit (a concept whose earliest admissible date could vary across observers, e.g., "there has been a lot of discussion about global warming on the news recently"), and timeless (no time-anchored entity is present).

We do not use a database agent because there is no internal database to cross-reference in this pipeline.

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.

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; 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 Evaluation

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

4 Results

Key plots for this analysis are Figure 3 and Figure 4. Figure 3 shows that search grounding improves no-leak accuracy by roughly 10 points across models, with Gemini models reaching 100% no-leak

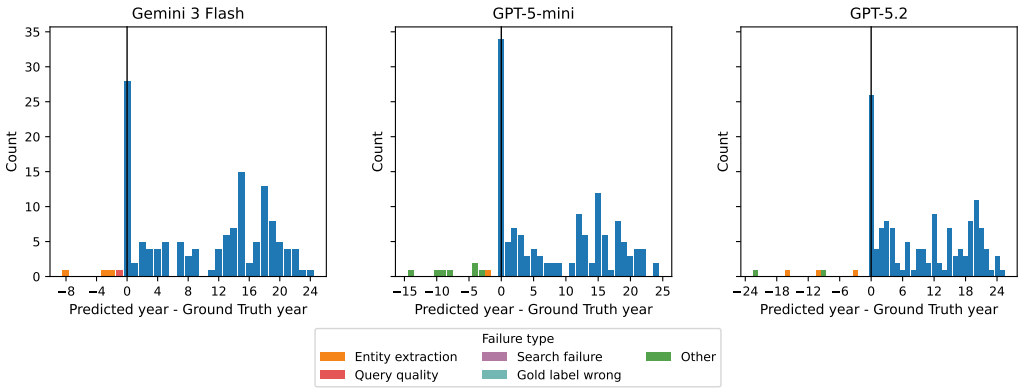


Figure 2: Search-aided prediction error (predicted year minus ground truth) by model on the held-out test set, with negative deltas colored by failure type.

accuracy on the dev set. Figure 4 shows the distribution of predicted year minus gold year; the most common failure is entity extraction, while search grounding appears reliable. We categorize error types using an LLM with access to the ground-truth answer. Figure 2 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; Figure 5 summarizes these counts and two max-ensembles.

4.1 Next Steps

Finalize the test-set plots and tables, then:

1. Measure inter-LLM consistency.
2. Measure inter-human-rater consistency.
3. Measure human/LLM correlation.
4. Add a simple LLM prompting baseline.
5. Quantify the gain from sampling N times rather than 1 time.
6. Update the test set with improved gold years once multiple annotations can be merged.

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.

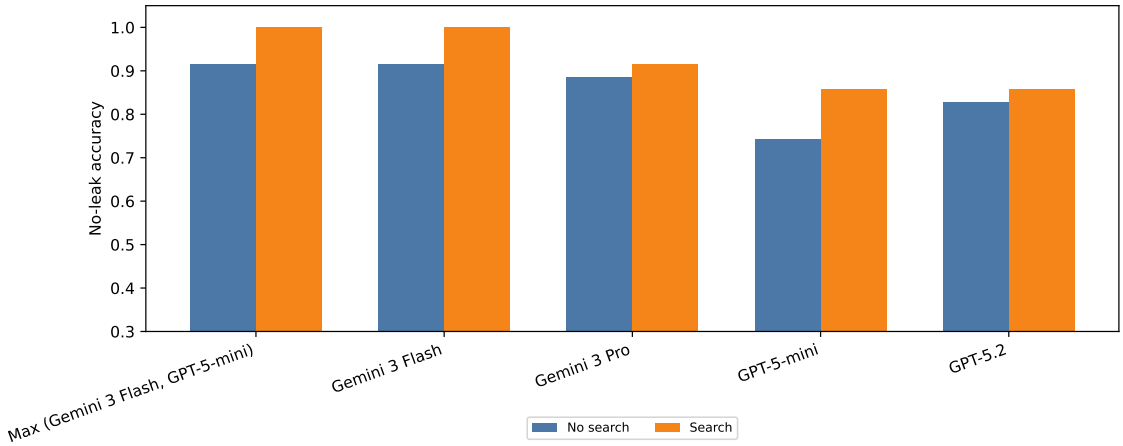


Figure 3: No-leak accuracy (predicted year $\geq$ gold) before vs. after search grounding.

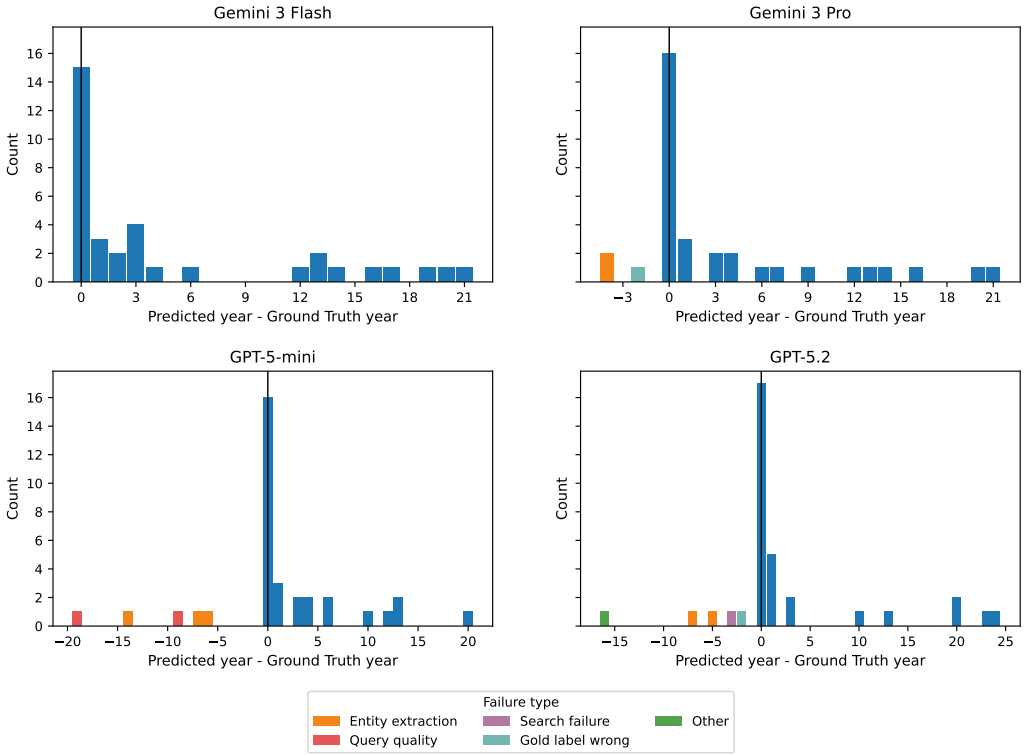


Figure 4: Search-aided prediction error (predicted year minus ground truth) by model, with negative deltas colored by failure type.

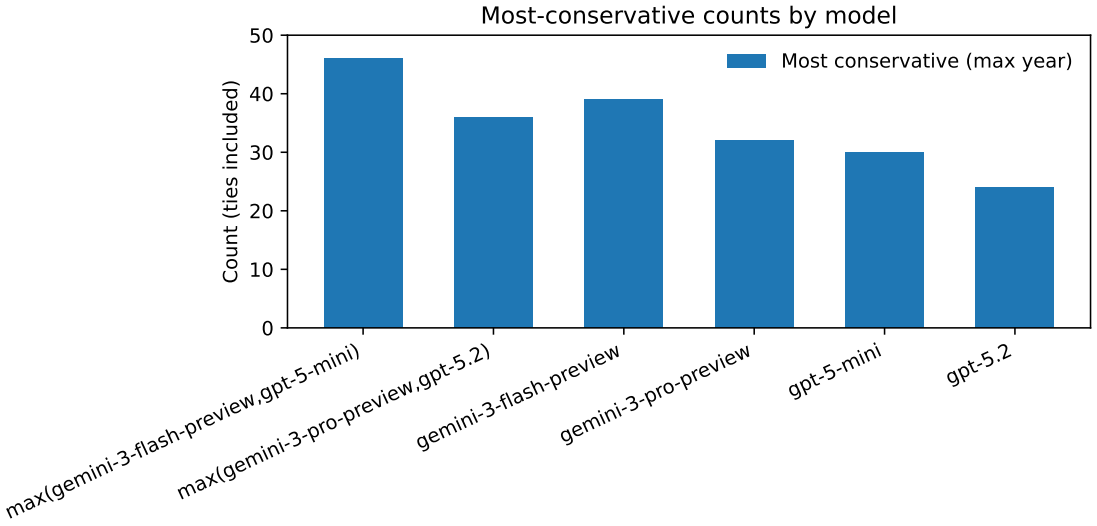
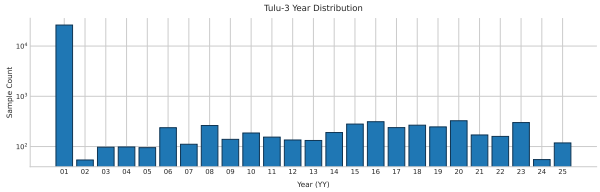
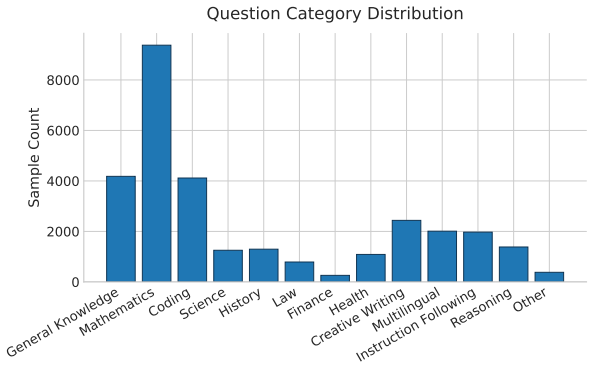


Figure 5: Most-conservative counts over 50 questions, with max-ensembles of Gemini 3 Flash + GPT-5-mini and Gemini 3 Pro + GPT-5.2 (ties count).

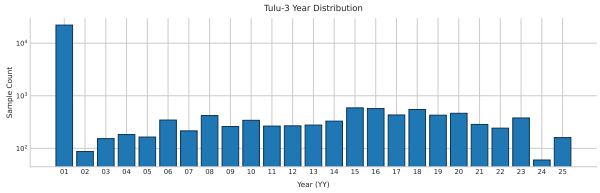


(a) Year distribution (SFT).

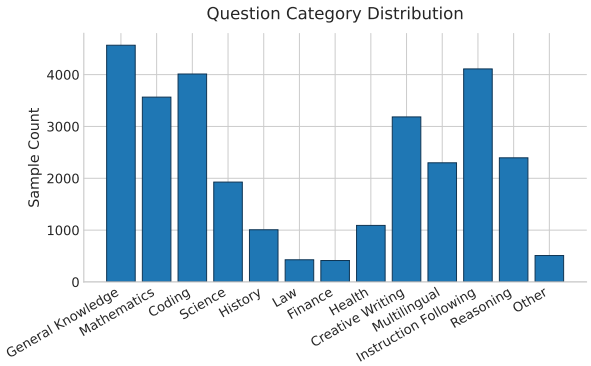


(b) Category distribution (SFT).

Figure 6: Filtering summary for the TÜLU-3 SFT mixture (session `2026-01-06_14-10PT`, $n = 30{,}549$).



(a) Year distribution (DPO).



(b) Category distribution (DPO).

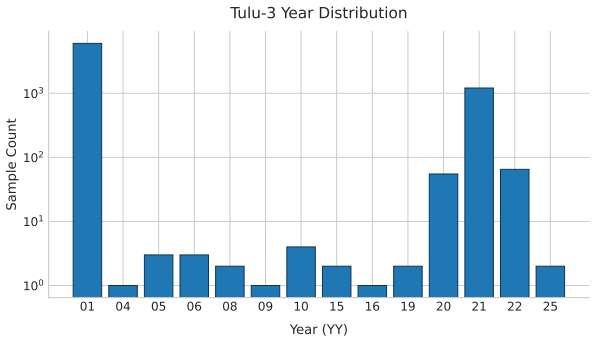
Figure 7: Filtering summary for the TÜLU-3 preference mixture (session `2026-01-06_14-10PT`, $n = 29{,}510$).

5 Discussion

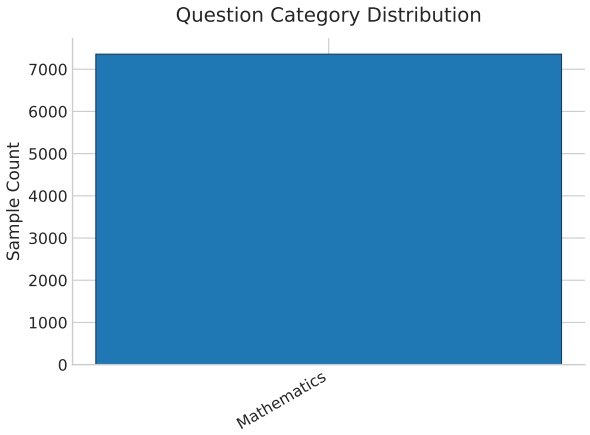
Two observations stand out: Gemini 3 Flash performs better than Gemini 3 Pro, and GPT-5-mini is similar to GPT-5.2. It would be valuable to explore whether this stems from smaller models producing more conservative intervals and from the asymmetric risk profile that penalizes underestimates more than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

6 Data Filtering

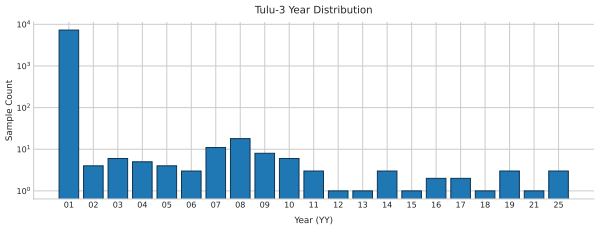
We label each supervised (SFT), preference, and RLVR sample with the minimum calendar year consistent with its question–answer bundle; the prompt structure is detailed in Appendix A. We considered deterministic filtering, but it was difficult to capture all edge cases with a rule-based approach. The 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 6–8 show year and category distributions for each dataset family and validate cutoff integrity.



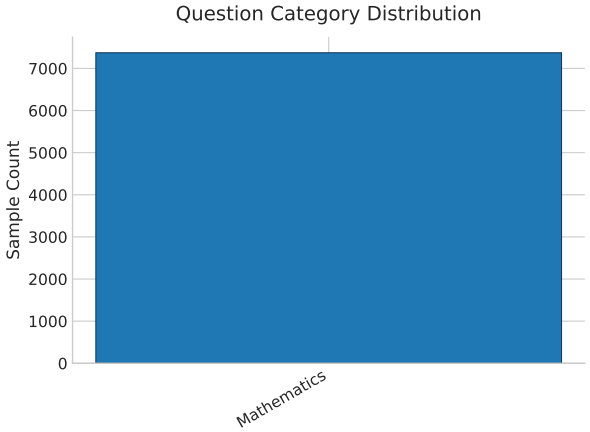
(a) Year distribution (RLVR-GSM).



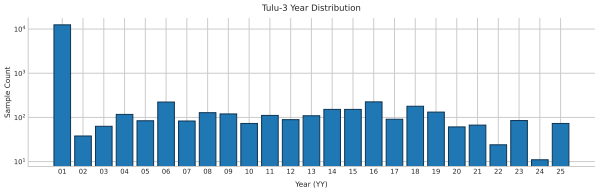
(b) Category distribution (RLVR-GSM).



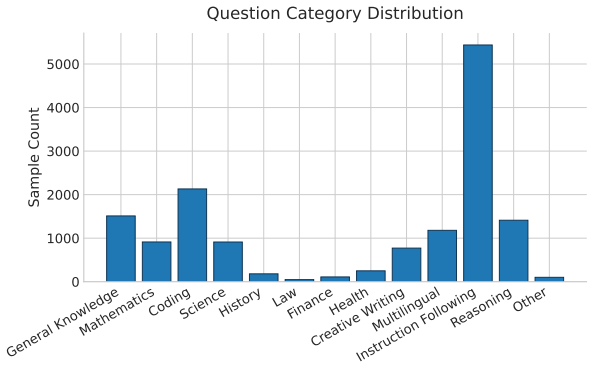
(c) Year distribution (RLVR-MATH).



(d) Category distribution (RLVR-MATH).



(e) Year distribution (RLVR-IFeval).



(f) Category distribution (RLVR-IFeval).

Figure 8: Filtering summary for the RLVR datasets (session `2026-01-06_14-10PT`, GSM $n = 7{,}358$, MATH $n = 7{,}372$, IFEval $n = 14{,}958$).

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.

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 two 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$\times$ 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 with `run_tulu3_dev_limit100.sh`, 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 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 summa-

Figure 9: LoRA SFT training metrics: disk utilization (GB), reserved GPU memory (GB), learning rate, total tokens, per-device tokens per second, and train loss.

rizes per-sample token averages, current costs, and projections for the full SFT/preference corpus plus the RLVR targets. TÜLU-3 still requires filtering 900k SFT examples and 250k 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).

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 900k/250k additional SFT/preference samples and RLVR targets of 8.79k (GSM), 7.5k (MATH), and 15k (IFeval).

A 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:
- 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.

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.",
```

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).

```
"entities": {"tweet": {"best_estimate": 2006, "confidence_interval_95": [2006, 2006],
"search_query": "When was Twitter launched?"}}}

<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?"}}}
```

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