Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Training on Broken Contexts: Source Loss and Repair in Tool-Augmented LLM Post-Training

Amir Reza Peimani

amir.peimani@utoronto.ca

University of Toronto

Research Article

Keywords: LLM post-training, supervised fine-tuning, tool-using agents, sequence packing, training data, data auditing

Posted Date: October 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11116713/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Training on Broken Contexts: Source Loss and Repair in Tool-Augmented LLM Post-Training

Amir Reza Peimani[1*]

[1*]Institute of Biomedical Engineering, University of Toronto, Toronto, Canada.

Corresponding author(s). E-mail(s): amir.peimani@utoronto.ca;

Abstract

Tool-augmented supervised fine-tuning can keep assistant targets in the training loss after sequence construction has separated them from the tool observations that supplied their values. We audit this failure by tracking each rendered token back to its original conversation and testing whether matched tool observations remain visible after assistant masking, truncation, splitting, and packing. We analyze ToolACE, ReTool, and a 5,000-conversation sample of Glaive-FC across six token budgets and several sequence-construction policies. At 512 tokens, BFD-split separates 38.66%, 19.61%, and 6.43% of retained matched targets from their recorded sources in ToolACE, ReTool, and Glaive-FC, affecting 2.67%, 15.05%, and 8.60% of conversations. Default BFD produces no such cases among retained matched targets, but retains only 17.23% and 8.10% of the original assistant supervision in ToolACE and ReTool. We then reconstruct training examples so that complete targets remain with the tool observations they use. On ToolACE at 512 tokens, grouping targets from the same conversation retains 875/918 matched targets while reducing the retained input-token ratio from 0.325 to 0.275 and eliminating 1,244 duplicated source tokens compared with reconstructing one example per target. For tool-call supervision, reducing the retained schema to the referenced parameters increases ToolACE coverage from 28.60% to 73.15%. Implementation checks also uncover a multi-batch token-stream defect in TRL 1.9.2 wrapped packing. Sequence construction therefore affects both how much supervision reaches the optimizer and whether that supervision is paired with the context that produced it.

Keywords: LLM post-training, supervised fine-tuning, tool-using agents, sequence packing, training data, data auditing

1 Introduction

Tool-using language models are commonly fine-tuned on trajectories containing user requests, tool calls, tool observations, and assistant responses. The stored conversation, however, is not the object presented to the optimizer. A chat template converts messages into tokens, an assistant mask selects the tokens that contribute to the loss, and sequence-construction code truncates, splits, or packs the result to fit a fixed context length. These operations can leave an assistant answer in the loss after placing the tool observation that supplied part of that answer outside its attention group.

Consider a tool that returns an order identifier, inventory location, measurement, or error code, followed by an assistant response that uses the returned value. If sequence construction separates the observation from the answer, the answer tokens can still be optimized even though the recorded value is no longer present in their context. Counting retained sequences or loss tokens does not expose this case because the supervision itself has survived.

Existing work on sequence construction has largely focused on efficient context use, arbitrary truncation, batching, and interactions between examples packed into the same sequence. Best-fit packing reduces document truncation [3], LongAlign and hierarchical balance packing address long-context data construction and batching [1, 22], and recent SFT studies examine cross-example context effects and the conditions under which packing improves training [5, 18]. Tool-use trajectories add a different dependency: later supervised tokens may rely on specific observations, user constraints, or tool declarations earlier in the trajectory.

Recent agent-training work makes some of these dependencies explicit before fine-tuning. Graph Explorer executes knowledge-graph trajectories under the deployed context rules and retains action supervision only when the identifiers used by an action are visible at that point [2]. ACC recompiles agent trajectories so that observations collected across tool turns become direct long-context supervision [16], while WebClipper prunes trajectory structure before training [17]. Our question arises after these higher-level trajectory decisions have been made: once a conversation has been rendered, masked, truncated, split, and packed, does a supervised target still contain the recorded context from which it was produced?

We answer this question by tracking the origin of every token through sequence construction. For values that can be matched directly from a tool observation to the following assistant response, we test whether the original observation remains earlier in the same attention group as the supervised target. We report the fraction of matched targets that lose their source, the fraction of conversations affected, and the fraction of the original assistant objective contributed by the affected target tokens.

Across ToolACE, ReTool, and Glaive-FC, sequence policies that split conversations across training examples frequently retain targets after separating them from their recorded tool observations. At a 512-token budget, BFD-split affects 38.66% of retained matched targets in ToolACE, 19.61% in ReTool, and 6.43% in Glaive-FC. These cases occur in 2.67%, 15.05%, and 8.60% of conversations, respectively. Default BFD produces no such separations among retained matched targets in these audits, but retains only 17.23% of the original assistant supervision in ToolACE and 8.10%

in ReTool. The choice of sequence policy therefore changes both supervision retention and the context available to that supervision.

We also reconstruct shorter training examples that keep complete targets with the matched tool observations they use. Combining compatible targets from the same conversation reduces repeated context without reducing target coverage. On ToolACE at 512 tokens, grouping retains 875/918 matched targets, the same number as reconstructing a separate example for each target, while reducing the retained input-token ratio from 0.325 to 0.275 and eliminating 1,244 duplicated source tokens. For tool-call targets, large tool schemas create a second context-length problem. Keeping only the invoked tool or the parameters used in the call raises short-budget coverage substantially on ToolACE, whereas ReTool's compact schemas are better kept intact.

Finally, we test the sequence-construction implementations themselves against token-level fixtures. These tests uncover a multi-batch defect in the TRL 1.9.2 wrapped-packing path that changes the emitted token stream at Arrow map boundaries. This result is analyzed separately from the packing-policy comparison, which uses the single-batch implementation of the intended concatenate-then-split operation.

The paper makes three main contributions:

1. We develop a token-level audit that tests whether supervised targets in tool-augmented SFT still have access to the tool observations from which their matched values were obtained after rendering and sequence construction.
2. We measure this failure across three released tool-training corpora, six token budgets, and multiple packing and truncation policies, showing large differences across datasets and a clear interaction between target context and supervision retention.
3. We reconstruct training examples to retain matched observation-to-answer and user-to-tool-call relations under fixed token budgets. Grouping reduces duplicated context at unchanged target coverage, and reducing large tool schemas substantially increases short-context tool-call coverage.

2 Related Work

2.1 Training trajectories for tool-using agents

Tool-use trajectories contain dependencies between user requests, tool calls, tool observations, and later actions or answers. Recent work has increasingly used these dependencies when generating or filtering training data. Graph Explorer executes knowledge-graph trajectories under the same context rules used by the agent and retains action supervision only when the identifiers required by the action remain visible [2]. Anchor generates GUI-agent trajectories from verified branch points and filters actions using state-based checks [19]. WebClipper removes redundant and inefficient branches from web-agent trajectories [17], while ACC recompiles multi-turn agent trajectories into long-context question–answer training examples [16]. TRACER links generated claims to the tool observations used to produce them [23].

These methods operate during trajectory generation, filtering, or restructuring. We begin with an existing tool-use trajectory and examine what happens afterward,

when chat rendering, masking, truncation, splitting, and packing produce the token sequences used for supervised fine-tuning.

2.2 Sequence construction and packing

Sequence construction affects both training efficiency and the context presented to the model. Best-fit packing reduces arbitrary document truncation [3], while LongAlign and hierarchical balance packing address batching and data construction for long-context training [1, 22]. For supervised fine-tuning, Threshold Filtering Packing groups related examples within packs [5], and Packing Analysis studies when packing improves efficiency without degrading downstream performance [18].

We examine a different consequence of sequence construction within tool-use trajectories. A target can remain in the training loss after the tool observation containing its value has been placed in a different attention group. We measure this after the complete rendering, masking, and length-handling procedure, rather than from the stored conversation or its nominal length.

2.3 Tool-use training data and tool observations

ReAct established the widely used reason–act–observe interaction pattern [21], and ToolLLM, ToolACE, and ReTool developed large-scale training methods and datasets for tool use and function calling [7, 14, 15]. Data quality is consequential in this setting: filtering synthetic tool-use examples for correctness can improve downstream tool performance even with fewer training examples [12]. Models can also fail to extract and use information from tool outputs even when those outputs are present in the inference context [13].

Our analysis addresses the preceding training-data problem. Before asking whether a model learns to use a tool observation correctly, we test whether the observation is still present when the corresponding answer tokens are optimized.

3 Audit Definition

A stored conversation is rendered with the model's chat template and assistant loss mask and is then truncated, split, or packed into training sequences. We record the original conversation and rendered-token position of every output token throughout these operations. This allows a value in an assistant target to be traced back to the particular tool observation from which it was matched, even when the same string occurs elsewhere.

The primary analysis considers values copied from a tool observation into the immediately following assistant message. A value is included when it can be located in both messages after full chat rendering and has not already appeared in earlier non-tool text. Details of value extraction and matching are given in Section 4.1.

After sequence construction, a matched target is retained if its target-value tokens still contribute to the training loss. Its recorded tool value remains available if the

complete matched occurrence appears earlier in the same attention group as the supervised target-value tokens. An identical value from another conversation or another position does not count as the recorded source.

We count a retained target as *separated* when its matched tool value is no longer available under the final attention structure. If several matched values connect the same tool observation and assistant message, they are treated as one target unit, and the target is counted as separated if any of those values loses its recorded source.

We report three quantities. The separation rate is the fraction of retained matched targets that are separated from their recorded tool values. We also report the fraction of original conversations containing at least one separated target and the affected target-value tokens as a fraction of the original assistant loss tokens. These quantities measure the rate within the matched subset, its distribution across conversations, and its contribution to the full SFT loss.

The primary analysis covers direct copied values. Paraphrases, calculations, multi-step semantic dependencies, and other relations that cannot be tied to a specific occurrence in the recorded trajectory are not included. A separate tool-call analysis matches quoted or numeric user constraints to the following assistant tool call and includes the corresponding tool declaration when evaluating context-length constraints.

4 Methods

4.1 Matching tool values to assistant targets

We identify direct observation-to-answer relations by matching values in tool observations to the immediately following assistant message. Candidate values are extracted from JSON leaves, fenced JSON, standalone numeric values, and plain stdout lines. A match is retained when the value appears in the following assistant message, has not appeared in earlier non-tool text, and both occurrences can be located after applying the full chat template.

Text matching is case-insensitive in the primary analysis, while numeric values are matched using lexical boundaries. We repeat the corpus analysis with strict case-sensitive matching. Prespecified subsets include nonnumeric values, values containing at least 6, 10, and 20 characters, and opaque structured codes. Paraphrases, calculations, and other semantic transformations are not included.

For tool-call targets, we separately match quoted or numeric constraints in the user message to values copied into the following assistant tool call. We then evaluate three amounts of accompanying schema context: the full schema block, the complete declaration of the invoked tool, and a valid schema containing only the referenced parameters and their associated metadata.

4.2 Sequence construction policies

We evaluate TRL 1.9.2 sequence construction at maximum lengths of 256, 512, 1,024, 2,048, 4,096, and 8,192 tokens. The evaluated policies are non-packed keep-start

truncation, best-fit-decreasing (BFD) packing, BFD-split packing, wrapped packing, drop-overlength, and the deprecated keep-end truncation policy as a historical comparison [10, 11].

BFD packs sequences using best-fit decreasing and truncates tokens beyond the maximum sequence length. BFD-split first divides overlength sequences into fragments no longer than the token budget and then packs those fragments. Wrapped packing concatenates tokenized examples into a continuous stream and divides the stream into fixed-length blocks, so a block boundary may fall inside an original conversation. TRL uses BFD as its default packing strategy and exposes BFD-split and wrapped as alternatives [10]. Its paper index provides a BFD-split configuration associated with the best-fit packing method of Ding et al. [3], and a public LongAlign regression script uses wrapped packing [4, 9].

The analysis uses the causal sequence boundaries produced by each policy. For BFD and BFD-split, TRL emits `seq_lengths` describing the sequences or fragments contained within each packed row. Under the padding-free execution evaluated here, position IDs restart at these boundaries, so each fragment is treated as a separate causal sequence. A target can therefore use only earlier tokens within its own fragment. Wrapped packing does not emit corresponding fragment boundaries; each wrapped block is treated as one causal sequence.

4.3 Token tracking and implementation checks

For every token produced by sequence construction, we retain its original dataset row and position in the rendered conversation. We propagate the same information for loss labels. This lets us determine whether a target still contains the particular tool-value occurrence to which it was matched, rather than another occurrence of the same string.

We test our implementation against TRL using fixed synthetic examples. For each sequence policy, we compare the emitted token IDs, labels, packed boundaries, and `seq_lengths` with TRL outputs. We also compare the rendered source and target spans with token IDs produced directly by `apply_chat_template`.

Wrapped packing is tested both with the released TRL 1.9.2 implementation and with a single-batch implementation of the intended concatenate-then-split operation. The policy comparisons use the single-batch token stream; the difference found in the released multi-batch implementation is reported separately in Section 5.4.

4.4 Datasets and statistical analysis

We analyze the complete pinned releases of ToolACE and ReTool, containing 11,300 and 2,000 conversations, respectively. For Glaive-FC, we analyze 5,000 conversations selected from 39,591 examples satisfying the required multi-turn action–observation–answer structure in the Apache-2.0 release. Eligible conversations are ranked by SHA-256 hash, and the 5,000 lowest-ranked examples are retained. Dataset revisions, hashes, and the complete Glaive eligibility rule are reported in the appendix.

The observation-to-answer matcher yields 918 target units in ToolACE and 1,611 in ReTool. The Glaive-FC sample contains 6,713 target units at 512 tokens before policy-specific target loss. When several matched values connect the same tool observation and assistant message, they are treated as one target unit.

We report exact counts and percentages. For corpus-level proportions, 95% confidence intervals are obtained from 2,000 cluster-bootstrap replicates [6]. Each replicate resamples complete conversations with replacement and recomputes the statistic, keeping values from the same conversation together.

4.5 Training-example reconstruction

We reconstruct shorter training examples containing complete assistant targets and the matched tool observations used by those targets. For each target, the required content includes the complete target message, the matched value occurrences in its tool observations, and the chat-template tokens needed to serialize those source messages. After the required content has been selected, any remaining token budget is filled with complete message blocks from the same conversation, scanning backward from the latest selected target.

Targets from the same conversation are grouped when their combined required content fits within the token budget. The grouping procedure is deterministic. Each group begins with the remaining target requiring the largest number of tokens. Additional targets are then added in order of smallest marginal token cost while they fit within the budget, with ties broken by their position in the conversation. A new group is started when no remaining target fits.

Selected target messages are retained completely, tokens outside those targets receive label -100, and each matched tool value must occur before the corresponding supervised target tokens in the same causal sequence. Tokens from different original conversations are never combined. Each serialized example is checked again after tokenization using the same token tracking and visibility test as the corpus analysis.

We compare grouping with two alternatives. The first constructs one training example for each matched target. The second retains the complete tool-observation and assistant-response rounds associated with the target. We compare the number of matched targets retained, assistant target tokens retained, input tokens relative to the original rendered corpus, examples generated per source conversation, duplicated source tokens, sequence lengths, and changes in how often individual conversations appear in the reconstructed dataset.

4.6 Tool-call schema reduction

For each matched user-constraint-to-tool-call relation, we construct an example containing the relevant user constraint, the complete assistant tool call, and one of three forms of tool schema.

The *whole-schema* condition retains every tool declaration supplied with the conversation. The *invoked-tool* condition retains only the complete declaration of the function called by the assistant. The *parameter-slice* condition retains the invoked

function and the descriptions, types, enums, defaults, constraints, and required status of the referenced parameters.

Calls with ambiguous function identity, missing declarations, or schemas that cannot be reconstructed as valid structures are not counted as covered. At each token budget, coverage is the fraction of matched tool-call targets that fit together with the relevant user constraint, complete tool call, and specified schema content.

4.7 Controlled source-removal experiment

We also test whether removing tool values during training changes model behavior in a controlled exact-value task. The dataset contains six independently generated value families and 1,800 training examples. In separate training conditions, the tool observation containing the answer value is removed from 0%, 25%, 50%, or 100% of examples while the original assistant answer remains in the training loss.

We fine-tune Qwen2.5-0.5B-Instruct with LoRA [8, 20] using three optimization seeds. Evaluation uses held-out examples and measures exact target-value generation and target-value negative log-likelihood. The complete data-generation procedure, model configuration, evaluation sets, and family-level results are reported in the appendix.

5 Results

5.1 Sequence splitting separates supervised targets from their tool observations

At a 512-token budget, BFD-split separates 353/913 retained ToolACE targets from their matched tool observations (38.66%, 95% cluster-bootstrap CI 35.50–41.98), 315/1,606 ReTool targets (19.61%, CI 17.74–21.76), and 431/6,698 Glaive-FC targets (6.43%, CI 5.87–7.04). The ToolACE rate remains similar in the prespecified higher-specificity subsets: 297/773 nonnumeric targets (38.42%), 171/417 values containing at least 20 characters (41.01%), and 46/134 opaque identifiers (34.33%).

The rate falls as the token budget increases (Figure 1a). The median contiguous span needed to contain both the matched tool value and assistant target is 194 tokens for ToolACE, 32 for ReTool, and 65 for Glaive-FC. The corresponding 95th percentiles are 913, 125, and 144 tokens.

Default BFD produces no target–source separations among the matched targets it retains, but retains little assistant supervision on two of the three datasets. At 512 tokens, BFD retains 17.23% of the original assistant loss tokens in ToolACE and 8.10% in ReTool, compared with 91.99% in Glaive-FC. BFD-split retains 99.79%, 99.79%, and 99.93%, respectively (Table 1; Figure 1b).

The corpus-wide impact differs substantially from the rate among matched targets. In ToolACE, the 38.66% target-level rate affects 302/11,300 conversations (2.67%) and 2.089% of the original assistant loss tokens. In ReTool, 301/2,000 conversations are affected (15.05%), but the affected values contribute only 0.125% of the original assistant loss because the matched relations are dominated by short numeric values.

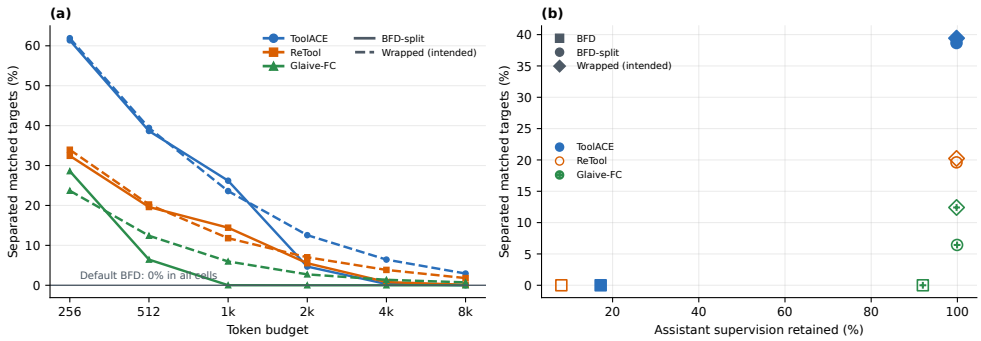


Fig. 1 Separation of matched assistant targets from their recorded tool observations across token budgets. (a) Fraction of retained matched targets whose tool value is no longer available after BFD-split or intended wrapped packing. Default BFD produces no such cases among retained matched targets in any corpus–budget combination. (b) Separation rate versus the fraction of original assistant supervision retained at 512 tokens

Table 1 Primary comparison at a 512-token budget. "Separated" counts retained matched targets whose recorded tool value is no longer available to the supervised target.

Corpus	Policy	Matched targets retained	Separated	Assistant supervision retained
ToolACE	BFD	10/918	0/10 (0.00%)	17.23%
	BFD-split	913/918	353/913 (38.66%)	99.79%
ReTool	BFD	6/1,611	0/6 (0.00%)	8.10%
	BFD-split	1,606/1,611	315/1,606 (19.61%)	99.79%
Glaive-FC	BFD	6,154/6,713	0/6,154 (0.00%)	91.99%
	BFD-split	6,698/6,713	431/6,698 (6.43%)	99.93%

Glaive-FC has 430/5,000 affected conversations (8.60%) and 1.224% of assistant loss tokens affected.

Intended wrapped packing gives similar or higher rates at 512 tokens: 39.43% in ToolACE, 20.24% in ReTool, and 12.41% in Glaive-FC. Strict case-sensitive matching changes the estimates very little. Under BFD-split, the corresponding rates are 38.80%, 19.84%, and 6.58%; under intended wrapped packing they are 39.62%, 20.36%, and 12.48%. The largest difference from the primary 512-token estimates is 0.22 percentage points. Results for all six token budgets are reported in the appendix.

5.2 Grouping retains the same targets with less duplicated context

On ToolACE at 512 tokens, grouping and one-example-per-target reconstruction both retain 875/918 matched targets (95.32%) and 82.19% of target supervision. Grouping reduces the retained input-token ratio from 0.325 to 0.275, reduces generated examples per original conversation from 1.291 to 1.025, and eliminates all 1,244 duplicated source tokens.

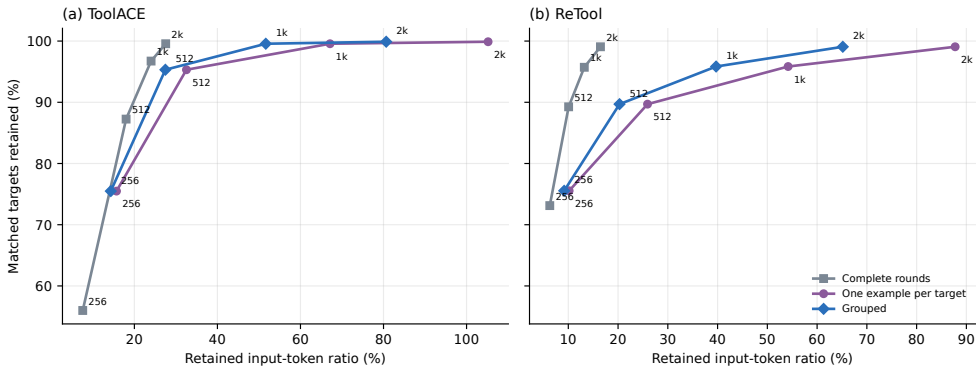


Fig. 2 Matched-target retention versus retained input-token ratio for (a) ToolACE and (b) ReTool at 256, 512, 1,024, and 2,048 tokens. Grouping targets from the same conversation generally matches one-example-per-target reconstruction using fewer input tokens. At 512 tokens, grouping reduces duplicated source tokens from 1,244 to zero in ToolACE and from 1,848 to 311 in ReTool

Retaining the complete tool-observation and assistant-response rounds uses fewer input tokens, with a ratio of 0.180, but retains only 801/918 targets (87.25%) and 65.35% of target supervision. The difference is larger at short budgets. At 256 tokens, grouping retains 693/918 targets (75.49%), compared with 514/918 (55.99%) when complete rounds are kept. At 1,024 tokens, grouping retains 914/918 targets (99.56%).

ReTool shows the same advantage over one-example-per-target reconstruction. At 512 tokens, both methods retain 1,445/1,611 matched targets (89.70%). Grouping reduces the input-token ratio from 0.259 to 0.203, examples per original conversation from 1.246 to 0.970, and duplicated source tokens from 1,848 to 311. Figure 2 shows the tradeoff between retained targets and input tokens across budgets.

5.3 Reducing large tool schemas increases short-context coverage

Among 7,591 ToolACE tool-call targets, retaining the full schema allows 28.60% to fit within 512 tokens. Retaining only the invoked tool increases coverage to 70.95%, and retaining only the referenced parameters increases it to 73.15%. At 256 tokens, the corresponding rates are 0.67%, 12.34%, and 15.89%. By 1,024 tokens, they rise to 83.70%, 88.86%, and 88.99%.

ReTool shows the opposite ordering because its trajectories generally contain a compact single-tool schema. At 512 tokens, the full schema fits with 34.42% of targets, compared with 27.33% for both reduced-schema variants. At 1,024 tokens, the corresponding rates are 65.44% and 54.80%. The benefit of reducing schema content therefore depends mainly on how much of the available token budget is occupied by tool declarations.

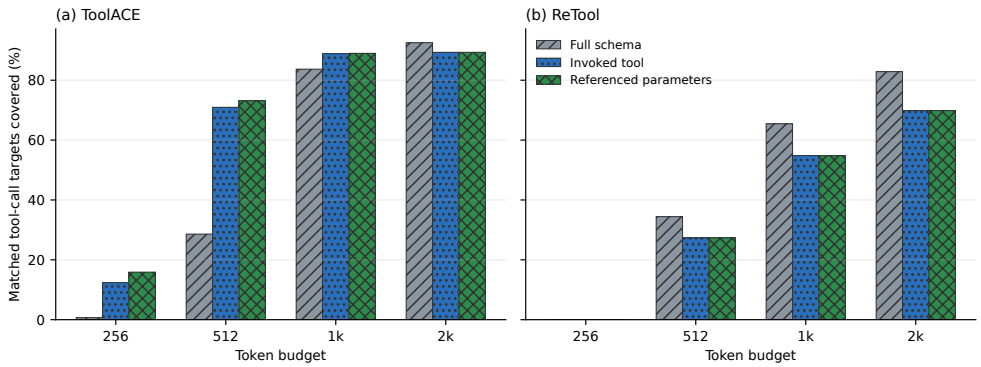


Fig. 3 Fraction of matched tool-call targets that fit within the token budget together with the relevant user constraint, complete tool call, and specified schema content for (a) ToolACE and (b) ReTool. Reducing the schema substantially increases short-budget coverage in ToolACE, whereas ReTool's compact schemas are better retained in full

5.4 TRL 1.9.2 wrapped packing changes the token stream across map batches

The released TRL 1.9.2 wrapped-packing implementation changes the intended token stream across Arrow map batches. We constructed a 2,001-row fixture containing two unique tokens per row so that every output position could be traced to its input row. The output matches the expected stream through position 1,999 and first diverges at position 2,000, exactly at the default 1,000-row map boundary. Overall, 2,002/4,002 positions come from the wrong input location. Processing the same fixture in a single map batch reproduces all 4,002 positions correctly.

Inspection of the implementation localizes the error to slice-local offsets being applied when reconstructing arrays from unsliced `column.values`. This causes parts of earlier Arrow storage to be repeated instead of preserving the intended concatenate-then-split stream.

The effect is also visible on ToolACE. At a 512-token budget, the released wrapped path contains assistant-token occurrences equal to 296.3% of the original assistant loss-token count, while unique coverage includes only 36.18% of the original assistant loss-token positions and 15.65% of source-token positions. The single-batch implementation of the intended wrapped operation retains 99.8% of assistant supervision. All wrapped results reported above therefore use that intended token stream.

6 Discussion

Sequence construction changes the training signal in tool-augmented SFT. Across ToolACE, ReTool, and Glaive-FC, BFD-split and wrapped packing frequently retain assistant targets after separating them from the tool observations containing their matched values. At 512 tokens, this occurs for 38.66% of retained matched targets in ToolACE, 19.61% in ReTool, and 6.43% in Glaive-FC. The corpus-wide impact differs substantially across datasets. ToolACE has the highest target-level rate but

only 2.089% of the original assistant loss tokens are affected. ReTool affects 15.05% of conversations but only 0.125% of assistant loss tokens because most matched values are short numeric fields. Reporting only one of these quantities would obscure an important part of the result.

Default BFD shows the other side of the tradeoff. It produces no target–source separations among the matched targets retained in our audits, but at 512 tokens it keeps only 17.23% of the original assistant supervision in ToolACE and 8.10% in ReTool. BFD-split retains almost all assistant supervision but separates some targets from their observations. Glaive-FC is less affected because most matched observation–target spans fit within the available context. Sequence construction therefore determines both the amount of supervision used for optimization and the context attached to that supervision. This extends previous analyses of SFT packing, which have focused primarily on efficiency, cross-example context effects, and downstream performance [18].

Grouping targets from the same conversation reduces the cost of reconstructing shorter examples. On ToolACE, grouping retains the same 875/918 matched targets as creating one example per target, while reducing the retained input-token ratio from 0.325 to 0.275 and eliminating 1,244 duplicated source tokens. ReTool shows the same pattern. Repeated source context increases training tokens and changes how often individual conversations appear in the reconstructed dataset, so combining compatible targets is a more efficient way to retain the same supervision.

The tool-call experiments show that schema size can dominate the context budget. In ToolACE, retaining the full schema allows only 28.60% of matched tool-call targets to fit within 512 tokens, compared with 70.95% when only the invoked tool is retained and 73.15% when only the referenced parameters are kept. ReTool behaves differently because its schemas are usually compact and contain a single tool. The full declaration is therefore preferable in that dataset. Schema reduction is useful when declarations are large enough to displace the user constraint or tool call that the model is being trained to reproduce.

The TRL 1.9.2 wrapped-packing result shows that the implementation itself can alter the training data beyond the intended packing rule. The released path first diverges from the expected token stream exactly at the Arrow map-batch boundary and assigns 2,002/4,002 positions to the wrong input locations in the controlled fixture. On ToolACE, the same behavior produces assistant-token occurrences equal to 296.3% of the original assistant loss-token count. Tracking the original row and position of each emitted token localizes the error to the reconstruction of batched Arrow columns. The result shows why packing should be checked on the emitted token stream instead of inferred from configuration settings.

The controlled source-removal experiment tests the learning consequence in a setting where the tool value determines the answer. Removing the value from every training example reduces held-out exact generation to zero and sharply increases target-value negative log-likelihood. Models trained with 25% or 50% of examples missing the value still recover the held-out values in this task. The experiment therefore shows that complete removal of the recorded tool information prevents learning of the exact-value mapping, while partial corruption is tolerated at the tested scale. The

corpus analysis addresses a different question: how often real sequence-construction policies produce such separations in released tool-training data.

Models already struggle to extract and use structured tool responses when those responses are present in context [13]. Removing the corresponding observation during training adds another failure before model reasoning becomes the limiting factor.

6.1 Limitations

The matcher covers values copied directly from a tool observation into the following assistant message. Paraphrases, calculations, multi-step dependencies, and other semantic relations are not counted. Some matched values may also be recoverable from other context or model knowledge. ToolACE, ReTool, and Glaive-FC contain generated tool-use trajectories, and Glaive-FC is evaluated on a deterministic 5,000-conversation sample. Reconstructing shorter examples changes some surrounding context and the frequency of examples derived from each conversation. The controlled experiment uses one 0.5B model and a synthetic exact-value task.

6.2 Practical implications

Training code can retain the original conversation and rendered position of each token through chat rendering, masking, truncation, splitting, and packing. These identifiers are sufficient to check whether a matched tool value remains available when its answer tokens contribute to the loss. Packing evaluations should report both assistant supervision retained and the fraction of matched targets separated from their observations. When long trajectories must be split, grouping compatible targets from the same conversation reduces repeated context, and large tool schemas can be reduced to the invoked tool or referenced parameters when the full declaration consumes too much of the token budget.

7 Conclusion

Tool-augmented SFT can retain assistant targets in the training loss after sequence construction has separated them from the tool observations containing their values. We find this behavior across three released tool-training corpora under policies that split conversations across training sequences. Default BFD avoids the problem among retained matched targets, but at short token budgets it discards most assistant supervision in ToolACE and ReTool.

Grouping targets from the same conversation retains the same supervision with fewer duplicated tokens, and reducing large tool schemas substantially increases tool-call coverage at short context lengths. We also identify a TRL 1.9.2 wrapped-packing defect that changes the emitted token stream across map batches. Sequence construction is therefore part of the effective training data in tool-augmented LLM fine-tuning and should be evaluated at the final token-sequence level.

Tables

Table 2 Glaive-FC dataset and sampling summary. Full revision and content hashes are included in the machine-readable manifest.

Dataset	Revision	License	Release rows	Eligible rows	Evaluation sample	Candidate relations
glaiveai/glaive-function-calling-v2	e7f4b6456019	Apache-2.0	112,960	39,591	5,000	15,541

Table 3 Complete 512-token results for the evaluated sequence policies. Separation rate is calculated among retained matched targets. Affected assistant tokens are source-separated target-value tokens divided by the original assistant loss-token count.

Corpus	Policy	Matched targets retained	Separated	Affected conversations	Affected assistant tokens	Assistant supervision retained
Glaive-FC	BFD	6,154/6,713	0/6,154 (0.00%)	0/5,000 (0.00%)	0.000%	91.99%
Glaive-FC	BFD-split	6,698/6,713	431/6,698 (6.43%)	430/5,000 (8.60%)	1.224%	99.93%
Glaive-FC	Keep-end (historical)	6,713/6,713	19/6,713 (0.28%)	19/5,000 (0.38%)	0.103%	99.79%
Glaive-FC	Wrapped (intended)	6,654/6,713	826/6,654 (12.41%)	826/5,000 (16.52%)	2.094%	99.81%
ReTool	BFD	6/1,611	0/6 (0.00%)	0/2,000 (0.00%)	0.000%	8.10%
ReTool	BFD-split	1,606/1,611	315/1,606 (19.61%)	301/2,000 (15.05%)	0.125%	99.79%
ReTool	Keep-end (historical)	1,259/1,611	153/1,259 (12.15%)	153/2,000 (7.65%)	0.055%	27.10%
ReTool	Wrapped (intended)	1,606/1,611	325/1,606 (20.24%)	308/2,000 (15.40%)	0.134%	99.81%
ToolACE	BFD	10/918	0/10 (0.00%)	0/11,300 (0.00%)	0.000%	17.23%
ToolACE	BFD-split	913/918	353/913 (38.66%)	302/11,300 (2.67%)	2.089%	99.79%
ToolACE	Keep-end (historical)	672/918	185/672 (27.53%)	185/11,300 (1.64%)	0.963%	88.20%
ToolACE	Wrapped (intended)	913/918	360/913 (39.43%)	314/11,300 (2.78%)	2.116%	99.80%

Table 4 Case-sensitive matching at a 512-token budget. Δ is the case-sensitive separation rate minus the primary case-insensitive rate, in percentage points.

Corpus	Policy	Separated / retained	Separation rate	Affected conversations	Affected assistant tokens	Primary rate	Δ
Glaive-FC	BFD-split	430/6,532	6.58%	429/5,000	1.193%	6.43%	+0.15 pp
Glaive-FC	Wrapped (intended)	810/6,489	12.48%	810/5,000	2.018%	12.41%	+0.07 pp
ReTool	BFD-split	313/1,578	19.84%	299/2,000	0.125%	19.61%	+0.22 pp
ReTool	Wrapped (intended)	321/1,577	20.36%	304/2,000	0.134%	20.24%	+0.12 pp
ToolACE	BFD-split	350/902	38.80%	300/11,300	2.079%	38.66%	+0.14 pp
ToolACE	Wrapped (intended)	357/901	39.62%	312/11,300	2.105%	39.43%	+0.19 pp

Table 5: Case-sensitive matching at every evaluated token budget.

Corpus	Policy	Budget	Separated / retained	Separation rate	Affected conversations	Affected assistant tokens	Primary rate	Δ
Glaive-FC	BFD-split	256	1,893/6,479	29.22%	1,711/5,000	4.810%	28.59%	+0.62 pp
Glaive-FC	BFD-split	512	430/6,532	6.58%	429/5,000	1.193%	6.43%	+0.15 pp
Glaive-FC	BFD-split	1,024	1/6,547	0.02%	1/5,000	0.010%	0.01%	+0.00 pp
Glaive-FC	BFD-split	2,048	0/6,547	0.00%	0/5,000	0.000%	0.00%	+0.00 pp
Glaive-FC	BFD-split	4,096	0/6,547	0.00%	0/5,000	0.000%	0.00%	+0.00 pp
Glaive-FC	BFD-split	8,192	0/6,547	0.00%	0/5,000	0.000%	0.00%	+0.00 pp
Glaive-FC	Wrapped (intended)	256	1,541/6,442	23.92%	1,409/5,000	3.921%	23.69%	+0.23 pp
Glaive-FC	Wrapped (intended)	512	810/6,489	12.48%	810/5,000	2.018%	12.41%	+0.07 pp
Glaive-FC	Wrapped (intended)	1,024	393/6,521	6.03%	393/5,000	0.966%	5.97%	+0.06 pp
Glaive-FC	Wrapped (intended)	2,048	180/6,535	2.75%	180/5,000	0.470%	2.73%	+0.02 pp
Glaive-FC	Wrapped (intended)	4,096	91/6,541	1.39%	91/5,000	0.260%	1.37%	+0.02 pp
Glaive-FC	Wrapped (intended)	8,192	48/6,545	0.73%	48/5,000	0.142%	0.73%	+0.00 pp
ReTool	BFD-split	256	514/1,575	32.63%	475/2,000	0.187%	32.44%	+0.20 pp
ReTool	BFD-split	512	313/1,578	19.84%	299/2,000	0.125%	19.61%	+0.22 pp
ReTool	BFD-split	1,024	231/1,580	14.62%	226/2,000	0.092%	14.43%	+0.19 pp
ReTool	BFD-split	2,048	88/1,581	5.57%	87/2,000	0.043%	5.53%	+0.04 pp
ReTool	BFD-split	4,096	13/1,582	0.82%	13/2,000	0.006%	0.81%	+0.01 pp

Continued on next page

Table 5: Case-sensitive matching at every evaluated token budget.

Corpus	Policy	Budget	Separated / retained	Separation rate	Affected conversations	Affected assistant tokens	Primary rate	Δ
ReTool	BFD-split	8,192	2/1,582	0.13%	2/2,000	0.001%	0.12%	+0.00 pp
ReTool	Wrapped (intended)	256	538/1,577	34.12%	492/2,000	0.190%	33.96%	+0.16 pp
ReTool	Wrapped (intended)	512	321/1,577	20.36%	304/2,000	0.134%	20.24%	+0.12 pp
ReTool	Wrapped (intended)	1,024	188/1,581	11.89%	183/2,000	0.086%	11.80%	+0.09 pp
ReTool	Wrapped (intended)	2,048	112/1,581	7.08%	111/2,000	0.059%	7.02%	+0.07 pp
ReTool	Wrapped (intended)	4,096	62/1,582	3.92%	62/2,000	0.025%	3.85%	+0.07 pp
ReTool	Wrapped (intended)	8,192	29/1,582	1.83%	29/2,000	0.011%	1.80%	+0.03 pp
ToolACE	BFD-split	256	551/896	61.50%	442/11,300	2.971%	61.37%	+0.13 pp
ToolACE	BFD-split	512	350/902	38.80%	300/11,300	2.079%	38.66%	+0.14 pp
ToolACE	BFD-split	1,024	239/905	26.41%	223/11,300	1.464%	26.20%	+0.21 pp
ToolACE	BFD-split	2,048	43/906	4.75%	40/11,300	0.417%	4.68%	+0.06 pp
ToolACE	BFD-split	4,096	3/906	0.33%	3/11,300	0.021%	0.33%	+0.00 pp
ToolACE	BFD-split	8,192	0/906	0.00%	0/11,300	0.000%	0.00%	+0.00 pp
ToolACE	Wrapped (intended)	256	561/899	62.40%	451/11,300	3.018%	61.91%	+0.49 pp
ToolACE	Wrapped (intended)	512	357/901	39.62%	312/11,300	2.105%	39.43%	+0.19 pp
ToolACE	Wrapped (intended)	1,024	214/903	23.70%	200/11,300	1.306%	23.61%	+0.09 pp
ToolACE	Wrapped (intended)	2,048	115/904	12.72%	114/11,300	0.598%	12.55%	+0.17 pp
ToolACE	Wrapped (intended)	4,096	59/905	6.52%	59/11,300	0.318%	6.43%	+0.09 pp
ToolACE	Wrapped (intended)	8,192	27/906	2.98%	27/11,300	0.101%	2.94%	+0.04 pp

Table 6 Comparison of training-example reconstruction methods at 512 tokens. Input-token ratio is the number of reconstructed input tokens divided by the number of tokens in the original rendered corpus. Examples per conversation use all original conversations as the denominator.

Corpus	Reconstruction	Matched targets retained	Retention	Target supervision retained	Input-token ratio	Examples / conversation	Duplicate source tokens
ReTool	Grouped	1,445/1,611	89.70%	51.05%	20.29%	0.970	311
ReTool	One example per target	1,445/1,611	89.70%	51.05%	25.92%	1.246	1,848
ReTool	Complete rounds	1,438/1,611	89.26%	50.37%	10.07%	1.240	0
ToolACE	Grouped	875/918	95.32%	82.19%	27.47%	1.025	0
ToolACE	One example per target	875/918	95.32%	82.19%	32.54%	1.291	1,244
ToolACE	Complete rounds	801/918	87.25%	65.35%	18.01%	1.181	0

Table 7 Fraction of matched tool-call targets that fit within the token budget together with the relevant user constraint, complete assistant tool call, and specified schema content.

Corpus	Schema content	256	512	1,024	2,048
ReTool	Full schema	0.00%	34.42%	65.44%	82.87%
ReTool	Invoked tool	0.00%	27.33%	54.80%	69.87%
ReTool	Referenced parameters	0.00%	27.33%	54.80%	69.87%
ToolACE	Full schema	0.67%	28.60%	83.70%	92.52%
ToolACE	Invoked tool	12.34%	70.95%	88.86%	89.33%
ToolACE	Referenced parameters	15.89%	73.15%	88.99%	89.33%

Table 8 Controlled-model results, reported as mean ± optimization-seed SD across three seeds. Exact generation requires the prespecified `VALUE=` field. NLL is computed over target-value tokens.

Condition	Clean exact	Clean value NLL	Conflict exact	Conflict stale copy
Unadapted base	0.00%	0.051	0.00%	0.00%
0% removed	77.28 ± 38.78%	0.001 ± 0.001	99.89 ± 0.10%	0.00 ± 0.00%
25% removed	100.00 ± 0.00%	0.000 ± 0.000	100.00 ± 0.00%	0.00 ± 0.00%
50% removed	100.00 ± 0.00%	0.000 ± 0.000	100.00 ± 0.00%	0.00 ± 0.00%
100% removed	0.00 ± 0.00%	1.245 ± 0.274	0.00 ± 0.00%	0.00 ± 0.00%

Table 9 Family-level results for the 0% and 100% source-removal conditions. The lower strict clean score at 0% includes the output-format variation observed in one optimization seed.

Family	Clean exact, 0% removed	Clean exact, 100% removed	Clean NLL, 0% removed	Clean NLL, 100% removed	Conflict exact, 0% removed	Conflict exact, 100% removed	Conflict stale copy, 100% removed
Order	66.00%	0.00%	0.001	1.157	100.00%	0.00%	0.00%
Error	66.33%	0.00%	0.000	1.189	100.00%	0.00%	0.00%
Inventory	75.67%	0.00%	0.000	1.435	100.00%	0.00%	0.00%
Route	99.67%	0.00%	0.001	1.055	99.67%	0.00%	0.00%
Ticket	66.67%	0.00%	0.000	1.230	100.00%	0.00%	0.00%
Checksum	89.33%	0.00%	0.004	1.406	99.67%	0.00%	0.00%

Statements and Declarations

Author contributions

Amir Reza Peimani conceived the study, developed the methods, performed the analyses, and wrote and revised the manuscript.

Funding

No funding was received for conducting this study.

Competing interests

The author has no relevant financial or non-financial interests to disclose.

Data and code availability

The code, configurations, recorded results, and data acquisition instructions are available at https://github.com/AmirRezaPeimani/when-supervision-outlives-support. Dataset and model revisions, licenses, and reproduction details are provided in the

appendix. A snapshot of the reproducibility package used for this submission is provided as Online Resource 1.

References

[1] Bai Y, Lv X, Zhang J, et al (2024) LongAlign: A recipe for long context alignment of large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp 1376–1395, https://doi.org/10.18653/v1/2024.findings-emnlp.74, URL https://aclanthology.org/2024.findings-emnlp.74/

[2] Chen Y, Wan S, Zhang T, et al (2026) Graph explorer: Training faithful KG agents with visibility-grounded supervision. In: Liakata M, Moreira VP, Zhang J, et al (eds) Findings of the Association for Computational Linguistics: ACL 2026. Association for Computational Linguistics, San Diego, California, United States, pp 7875–7890, https://doi.org/10.18653/v1/2026.findings-acl.387, URL https://aclanthology.org/2026.findings-acl.387/

[3] Ding H, Wang Z, Paolini G, et al (2024) Fewer truncations improve language modeling. In: Proceedings of the 41st International Conference on Machine Learning, pp 11030–11048, URL https://proceedings.mlr.press/v235/ding24f.html

[4] Diro A (2026) accelerate_pr4022_test_gist.py. URL https://gist.github.com/AmineDiro/9fd331214626b60e4d421264637b3828

[5] Dong J, Jiang L, Jin W, et al (2025) Threshold filtering packing for supervised fine-tuning: Training related samples within packs. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp 4422–4435, https://doi.org/10.18653/v1/2025.naacl-long.226, URL https://aclanthology.org/2025.naacl-long.226/

[6] Efron B, Tibshirani RJ (1993) An Introduction to the Bootstrap. Chapman & Hall

[7] Feng J, Huang S, Qu X, et al (2026) ReTool: Reinforcement learning for strategic tool use in LLMs. In: The Fourteenth International Conference on Learning Representations, URL https://openreview.net/forum?id=tRk1nofSmz

[8] Hu EJ, Shen Y, Wallis P, et al (2022) LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=nZeVKeeFYf9

[9] Hugging Face (2026) Paper Index. URL https://huggingface.co/docs/trl/paper_index, entry for "Fewer Truncations Improve Language Modeling"; accessed 2026-08-08

[10] Hugging Face (2026) SFT Trainer. URL https://huggingface.co/docs/trl/sft_trainer, accessed 2026-08-08

[11] Hugging Face (2026) TRL: Transformers reinforcement learning. URL https://github.com/huggingface/trl/releases/tag/v1.9.2, version 1.9.2

[12] Iskander S, Tolmach S, Shapira O, et al (2024) Quality matters: Evaluating synthetic data for tool-using LLMs. In: Al-Onaizan Y, Bansal M, Chen YN (eds) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Miami, Florida, USA, pp 4958–4976, https://doi.org/10.18653/v1/2024.emnlp-main.285, URL https://aclanthology.org/2024.emnlp-main.285/

[13] Kate K, Rizk Y, Ghosh P, et al (2026) How good are LLMs at processing tool outputs? In: Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pp 2918–2941, https://doi.org/10.18653/v1/2026.eacl-long.134, URL https://aclanthology.org/2026.eacl-long.134/

[14] Liu W, Huang X, Zeng X, et al (2025) ToolACE: Winning the points of LLM function calling. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=8EB8k6DdCU

[15] Qin Y, Liang S, Ye Y, et al (2024) ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=dHng2O0Jjr

[16] Su Q, Fang Z, Huang S, et al (2026) ACC: Compiling agent trajectories for long-context training. URL https://arxiv.org/abs/2605.21850, arXiv preprint

[17] Wang J, Xie Z, Yang D, et al (2026) WebClipper: Efficient evolution of web agents with graph-based trajectory pruning. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 21646–21666, https://doi.org/10.18653/v1/2026.acl-long.988, URL https://aclanthology.org/2026.acl-long.988/

[18] Wang S, Wang G, Wang Y, et al (2025) Packing analysis: Packing is more appropriate for large models or datasets in supervised fine-tuning. In: Findings of the Association for Computational Linguistics: ACL 2025, pp 4953–4967, https://doi.org/10.18653/v1/2025.findings-acl.256, URL https://aclanthology.org/2025.findings-acl.256/

[19] Wei J, Zhao Y, Ni K, et al (2026) Anchor: Branch-point data generation for GUI agents. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 17031–17047, https://doi.org/10.18653/v1/2026.acl-long.774, URL https://aclanthology.org/2026.acl-long.774/

[20] Yang A, Yang B, Zhang B, et al (2024) Qwen2.5 technical report. URL https://arxiv.org/abs/2412.15115, arXiv preprint

[21] Yao S, Zhao J, Yu D, et al (2023) ReAct: Synergizing reasoning and acting in language models. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=WE_vluYUL-X

[22] Yao Y, Tan J, Liang K, et al (2025) Hierarchical balance packing: Towards efficient supervised fine-tuning for long-context LLM. In: Advances in Neural Information Processing Systems, URL https://papers.nips.cc/paper_files/paper/2025/hash/410bbba8388369d8bb5875544d1d4428-Abstract-Conference.html

[23] Yu B, Jia C, Chi J, et al (2026) TRACER: Verifiable generative provenance for multimodal tool-using agents. URL https://arxiv.org/abs/2605.09934, arXiv preprint

Appendix A Reproducibility

Online Resource 1 contains the code, configurations, and recorded outputs used for conversation rendering, token tracking, sequence-policy evaluation, value matching, training-example reconstruction, tool-call schema reduction, and the controlled model experiment. Experiment records include dataset or model revisions, random seeds where applicable, raw outputs, derived results, runtime, and compute information. The test suite covers chat-template rendering, source and target span mapping, token tracking through packing, causal sequence boundaries, training-example reconstruction, schema validity, TRL output checks, and train/development/test separation.

The main reproduction commands are:

```
PYTHONPATH=src python scripts/run_phase1.py \
  --config configs/primary_audit.json \
  --output-dir outputs/raw/phase1_attention_corrected

PYTHONPATH=src python scripts/run_phase1_glaive.py \
  --config configs/glaive.json \
  --output-dir outputs/raw/glaive_attention_corrected

PYTHONPATH=src python scripts/run_phase1.py \
  --config configs/case_sensitive_toolace_retool.json \
  --output-dir outputs/raw/case_sensitive_tool_retool

PYTHONPATH=src python scripts/run_phase1_glaive.py \
  --config configs/glaive_case_sensitive.json \
  --output-dir outputs/raw/case_sensitive_glaive

PYTHONPATH=src python scripts/run_materialization.py \
  --config configs/materialization.json \
  --output-dir outputs/materialization_causal_validator

PYTHONPATH=src python scripts/run_action_repair.py \
  --config configs/action_repair.json \
  --output-dir outputs/schema_repair_reproduced

PYTHONPATH=src pytest -q
```

Environment specifications and complete command records accompany the released outputs.

Appendix B Datasets and licenses

ToolACE and ReTool are analyzed from their complete pinned releases, containing 11,300 and 2,000 conversations, respectively. Both releases declare the Apache-2.0 license. Glaive-FC uses the Apache-2.0 `glaiveai/glaive-function-calling-v2` release at revision

`e7f4b6456019f5d8bcb991ef0dd67d8ff23221ac`.

The analyzed Dataset Viewer artifact is 97,037,816 bytes with SHA-256

`7e7e32f0466699513b38923bc4991858e192c9274df22e243b9e77eb2775b6e8`.

Of 112,960 Glaive-FC rows, 39,591 satisfy the specified multi-turn action/observation/answer eligibility rule. Eligible conversations are ranked by SHA-256 hash, and the 5,000 lowest-ranked conversations form the evaluation sample. Table 2 summarizes the Glaive-FC release and evaluation sample.

Appendix C Complete 512-token policy results

Table 3 reports the complete 512-token policy results.

The released TRL 1.9.2 wrapped implementation is omitted from the policy comparison because its multi-batch output changes the intended token stream, as reported in Section 5.4. "Wrapped (intended)" denotes the single-batch implementation of the concatenate-then-split operation used for the main comparison. Keep-end is included only as a historical policy. BFD-split is a documented TRL packing strategy that can divide an overlength conversation across sequences [10].

Appendix D Token-tracking checks and case-sensitive matching

The audit requires the matched tool-value occurrence to precede the supervised target in the same causal sequence. Four regression cases test the implementation directly: a matched value earlier in the same sequence is available to the target; a value appearing after the target is not; a matched value placed in another sequence is not; and identical text originating from another row or position does not substitute for the recorded occurrence.

We repeat value matching with strict case-sensitive string identity. Table 4 reports the 512-token results. The largest change from the primary separation rate is 0.22 percentage points across the three corpora and two sequence-splitting policies.

Results for all six token budgets are reported in Table 5.

Appendix E Training-example reconstruction

For each matched assistant target, we identify the complete target message and the matched tool-value occurrences associated with that target. The required token set also includes the chat-template wrapper tokens of the corresponding source messages. Targets whose required content already exceeds the token budget are not retained.

Grouping is deterministic. The remaining target with the largest required token set starts a new group. Among the remaining targets that still fit, the algorithm

repeatedly selects the one that adds the fewest new tokens, breaking ties by target position in the conversation. When no additional target fits, the group is serialized and the procedure starts another group.

After a group has been selected, unused capacity is filled with complete message blocks from the same conversation, scanning backward from the latest selected target. Selected target messages retain their original assistant labels; all other retained tokens receive label -100.

Every serialized example is then checked for four properties: each selected target message is complete; its matched tool value is present earlier in the same causal sequence; labels are restricted to the selected targets; and the example contains tokens from only one original conversation. All 34,128 serialized examples generated in the reconstruction experiments pass these checks. Table 6 compares the three reconstruction methods at 512 tokens.

Appendix F Tool-call schema reduction

For each matched user-constraint-to-tool-call relation, the whole-schema condition retains every tool declaration supplied with the conversation. The invoked-tool condition retains only the complete declaration of the function called by the assistant. The parameter-slice condition constructs a valid JSON schema containing the invoked function and the types, enums, defaults, constraints, required status, and descriptions associated with the referenced parameters.

Calls with ambiguous function identity, missing declarations, or schemas that cannot be reconstructed as valid structures are not counted as covered. Table 7 reports tool-call coverage for the three schema conditions across token budgets.

ToolACE benefits strongly from reducing large declarations at short token budgets. ReTool generally contains a compact single-tool schema, so retaining the full declaration gives higher coverage.

Appendix G Statistical details

Corpus confidence intervals use 2,000 bootstrap replicates that resample complete original conversations with replacement. Numerators and denominators are recomputed within each replicate, so multiple matched values and targets from the same conversation remain grouped.

For each sequence policy we report the number and fraction of retained matched targets separated from their tool observations, the number and fraction of original conversations affected, the affected target-value tokens as a fraction of the original assistant loss tokens, and the fraction of original assistant supervision retained by the policy. Exact counts accompany percentages.

The nonnumeric, minimum-length, and opaque-code subsets used in the matching sensitivity analysis were defined before the corresponding policy results were examined.

Appendix H Controlled source-removal experiment

H.1 Data and training

The controlled experiment uses six independently generated families of tool-derived values: order confirmations, recovery codes, inventory locations, route authorizations, ticket handles, and checksums. Each family contributes 300 training examples, 60 development examples, 100 clean test examples, and 100 conflict examples. Training, development, and test sets use separate key namespaces and independently generated values, preventing key or answer overlap across splits.

Four training conditions remove the tool response containing the answer value from 0%, 25%, 50%, or 100% of the 1,800 training examples while leaving the original assistant answer in the loss. The 25% and 50% conditions are generated from a single nested SHA-256 ranking over the training set. The number of removed examples per family ranges from 68 to 84 of 300 at 25% and from 140 to 167 at 50%.

Training and development prompts also contain an explicitly marked stale-cache value before the tool call. The clean test set omits this value, while the conflict test set retains it. All examples fit within the 512-token budget. Rendered lengths are 285–308 tokens in training, 288–305 in development, 263–281 in the clean test set, and 293–310 in the conflict set.

Generation is greedy with a 20-token limit. The prespecified exact-generation endpoint requires the `VALUE=` field to contain the held-out target value exactly. We also compute target-value and whole-target negative log-likelihood on the same held-out examples.

The model is Qwen2.5-0.5B-Instruct at revision `7ae557604adf67be50417f59c2c2f167def9a775`. All linear projections are adapted with LoRA using rank 16, alpha 32, and dropout 0.05. Optimization uses AdamW with learning rate 2×10^{-4}, batch size 4, gradient accumulation over four batches, and one training epoch. The three optimization seeds are 20260808, 20260809, and 20260810. Training uses float16 on Apple MPS.

The training configuration was selected using the 360-example development set. In the 0% source-removal condition, it reached 360/360 exact development generations, produced no stale-value copies, and achieved mean target-value NLL of 0.0000844. The training configuration and clean and conflict test sets were then held fixed for the four-condition comparison. Dataset, configuration, training, evaluation, and analysis hashes are recorded in `configs/model_study_revision2_frozen_manifest.json`.

H.2 Results

Figure 4 and Table 8 summarize the controlled source-removal results.

The unadapted model produces 0% exact generation on both evaluation sets. With 0%, 25%, 50%, and 100% source removal during training, clean exact generation is 77.28%, 100%, 100%, and 0%, respectively. On the conflict set, the corresponding values are 99.89%, 100%, 100%, and 0%.

Complete source removal also increases target-value NLL from 0.000888 to 1.245163 on the clean set and from 0.000776 to 1.039038 on the conflict set.

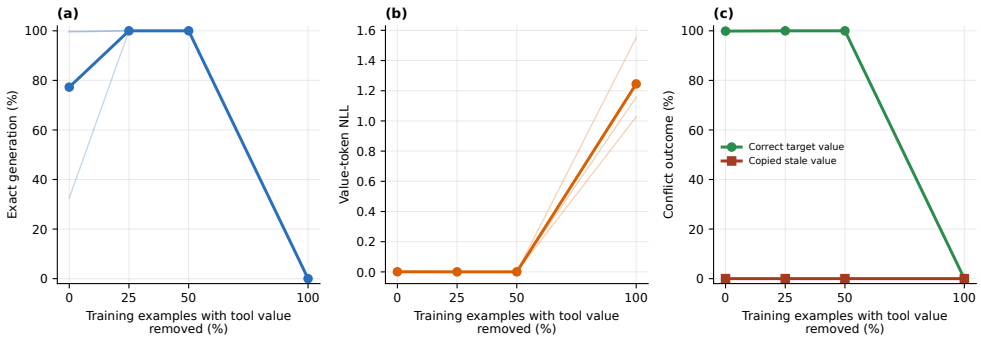


Fig. 4 Controlled source-removal experiment across three optimization seeds. (a) Clean exact generation. (b) Clean target-value negative log-likelihood. (c) Conflict-set outcomes. Exact generation is 100% at 25% and 50% source removal and falls to zero when the tool value is removed from all training examples. In the 0% condition, one seed frequently emits the correct value using `VALUE:` instead of the prespecified `VALUE=` format, reducing its strict exact score

For the 100%-minus-0% comparison, paired-record 95% bootstrap intervals are $[-78.56, -76.00]$ percentage points for clean exact generation and $[1.2165, 1.2710]$ for clean target-value NLL. The corresponding conflict intervals are $[-100.00, -99.72]$ percentage points and $[1.0188, 1.0597]$. These intervals resample evaluation records while retaining all three trained seeds within each replicate; seed-level SD is reported separately. Table 9 reports the corresponding family-level results for the 0% and 100% source-removal conditions.

The lower clean exact score at 0% source removal is largely a formatting effect. Seed 20260810 scores 195/600 under the strict parser because many correct values are emitted as `VALUE:target`; the other two seeds score 599/600 and 597/600. On the conflict set, the same three models score 599/600, 599/600, and 600/600. The low-scoring clean seed also retains low target-value NLL.

As a format diagnostic, searching the generated text for the held-out value finds the correct value in $99.39\% \pm 0.67$ of clean generations at 0% source removal, 100% at 25% and 50%, and 0% at 100%. No model copies the stale value on the conflict set.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1Reproducibility.zip