Measuring the Limits of Continual Learning for LLMs

Nimit Kalra [*1] **Narutatsu Ri** [*2] **Zerzar Bukhari** [3] **Ang Li** [4] **Sanae Lotfi** [4] **Liam Fowl** [2] **Micah Goldblum** [1]

Abstract

Language models are trained in stages but deployed as mostly static artifacts, leaving them poorly matched to a world that continually produces novel information. This trait has motivated a broad class of *continual learning* systems that adapt models to new information through weight updates, retrieval, memory, long-context inference, or hybrid mechanisms. Yet, existing evaluations do not tell us whether such systems have truly *internalized* new information: whether they can go beyond memorizing new information and update stale beliefs, resolve indirect references, compose new facts with prior knowledge, surface facts even when only implicitly relevant, and confidently recognize gaps in its own knowledge. We construct IMPRINTBENCH, a benchmark of realistic settings that expose these systematic shortcomings. IMPRINTBENCH consists of a refreshable pipeline that automatically constructs evaluations across three domains: news events, open-source API changes, and evolving personalization histories, with queries spanning six capability families: acquisition, temporal update, referential resolution, composition, implicit relevance, and boundary awareness. Across in-the-wild update scenarios, we find common systematic failures in both retrieval-based and training-based methods, showing that current systems still fall short of robustly learning from new experience.

1. Introduction

Language models acquire their knowledge base during pre- and post-training, and are then often deployed without further weight updates for months to service traffic. However, during deployment, relevant world information continues to evolve: APIs change, libraries deprecate functions, user preferences adapt, and world events unfold. Keeping a deployed model in sync with a constantly changing world has motivated an increasingly broad family of *continual learning* methods. These span tactics like retrieval-augmented generation (RAG), parametric updates, self-distillation, and context compression, among others.

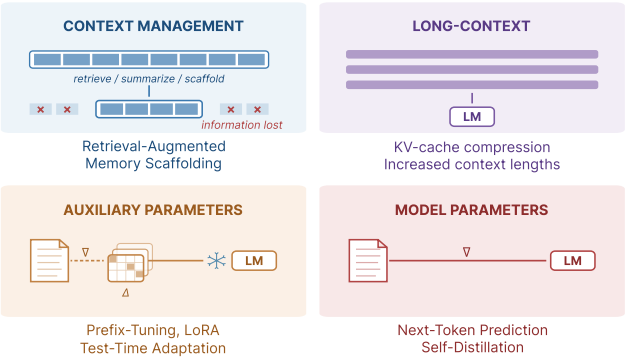

Figure 1. We study four broad families of continual learning approaches. *Context management* retrieves or summarizes content into the prompt; *long-context* approaches aim to efficiently support ever-growing corpus lengths; *auxiliary parameters* compress the corpus into adapters or a KV cache that a model reads from; *model parameters* update model weights directly. Each family has a distinct structural limitation that becomes visible with corpus size.

What it actually means for a model to *internalize* this new knowledge, however, is rarely pinned down by current evaluations. Most existing evaluations only delve as deep as testing direct recall: after exposure to new information, e.g., a document, news article, or API, can the model regurgitate this information when explicitly asked for it? This barometer is clearly insufficient, as truly learning new information requires seamlessly integrating it into a model's internal beliefs and understanding about related areas. For example, a change in a major world political leader should not just affect an intelligent model's answer to a simple question about the identity of the current leader, but also affect its understanding of what policies that nation might be more or less likely to pursue under the new leadership. Worse, evaluations typically test a single capability, domain, and method at a time, yielding little signal about which approaches work best across a range of tasks.

In order to address this lack of signal, we introduce IM-

[*]Equal contribution [1]Columbia University [2]Princeton University [3]Marble.ai [4]New York University. Correspondence to: Nimit Kalra <nimit@utexas.edu>, Narutatsu Ri <nr3764@princeton.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

PRINTBENCH, a benchmark designed to test a wide range of continual learning capabilities. We make two design commitments that distinguish IMPRINTBENCH: breadth of measurement, as every method is evaluated against a spectrum of difficult questions and each method is given the same new knowledge corpus, allowing for an apples-to-apples comparison across approaches, and refreshability, as our evaluation questions are produced by an automatic pipeline that can be run on-demand to capture new information like world events and code library releases, preventing contamination and allowing for longitudinal comparisons across models and methods. Our pipeline focuses on three settings that mirror these design principles: dynamic world knowledge sourced from real news articles, tool/library updates drawn from popular open-source API changelogs, and long-horizon personalization built from streaming user histories. Across these settings, we stress-test different continual learning methods across six capabilities spanning recall (direct acquisition, reference resolution, implicit relevance), reasoning over updates (temporal update, compositional reasoning), and abstention (boundary awareness), with questions carefully designed to be grounded in real source material and automatically screened so that they cannot be answered from prior knowledge or the prompt alone, leaving genuine internalization as the only route to a correct answer.

We use IMPRINTBENCH to evaluate methods spanning a handful of design families for continual learning (cf. Figure 1). *In-context learning* simply places all new information directly in the prompt, but realistic update streams exceed any finite context window, and recall degrades over the long inputs that fit within it (Liu et al., 2024). *Retrieval-augmented* generation (Lewis et al., 2020; Gao et al., 2024) sidesteps this limit by injecting only the most relevant documents, but it inherits a coverage gap, surfacing only what the retriever can find. *Memory and summarization* scaffolds (Packer et al., 2024) extend this reach by compressing history into a fixed-size buffer, so the aggressive ratios required for large corpora should trade away the fine-grained detail that continual learning aims to preserve. *Context compression* methods such as KV-cache compression (Kim et al., 2025) and trained prefixes (Eyuboglu et al., 2025) aim to preserve a frozen model's behavior at a fraction of the tokens, a promising direction whose fidelity we expect to be strained at the high compression ratios large corpora demand. *Parametric training* instead writes updates into the weights through next-token prediction, supervised fine-tuning, or self-distillation (Shenfeld et al., 2026), the only family whose inference cost is independent of corpus size but one we expect to risk overwriting prior knowledge (Kalajdzievski, 2024). Each is a well-established approach to keeping models current, one we expect to reveal a distinct structural limitation as the corpus grows.

Evaluating these methods under matched exposure budgets across $\sim$4,300 questions, we find that no method internalizes updates across all capabilities. Notably, for training-based methods, direct recall accuracy is a weak predictor of performance on harder capabilities. Performance also falls sharply with corpus size; the compression that large corpora force costs context-management methods 20–40 points and leaves dedicated compression schemes no better than compute-matched baselines, while parametric training stays flat but still trails in-context methods wherever the corpus fits. Retrieval recovers facts well but rarely abstains when the answer is absent, whereas parametric training confidently asserts new facts but struggles when a new fact must supersede an old one. Among parametric methods, supervised fine-tuning and self-distillation perform comparably given matched training data, differing mainly on this temporal axis, where fine-tuning commits to the new fact and self-distillation preserves the old.

Concretely, our contributions are:

- IMPRINTBENCH, an auto-refreshing benchmark and pipeline that decomposes knowledge internalization into six testable capabilities beyond direct recall and measures them across three real-world settings.
- A unified evaluation of several inference-time and training-time continual learning methods in controlled compute- and data-matched conditions.
- Analysis of where and how current continual learning methods fail, and how their accuracy degrades as the update corpus grows.

2. Beyond Recall: Measuring *Internalization*

2.1. Types of Capabilities

We say a model has *internalized* an update when it can not only recall the new information but also surface it from indirect cues, reason with it, and recognize its limits. We accordingly design IMPRINTBENCH around six capabilities that test for these facets.

Direct and Indirect Recall. A natural floor that we must test is *direct acquisition* (**DirAcq**): asked directly, the model should report a fact it was just exposed to. Yet clearing DirAcq is necessary but not sufficient, as a model can reproduce a fact verbatim while being unable to recognize it phrased any other way. *Reference resolution* (**RefRes**) targets exactly this gap, querying the same fact through aliases, paraphrases, descriptions, or role-based identifiers; since real queries rarely echo the exact wording of the source, it separates a model that has learned a fact from one that has memorized a string. A step beyond this rephrasing is *implicit relevance* (**ImpRel**), which examines whether the model surfaces an update when a query invokes it only in-

World News	
DirAcq	Did an Israeli airstrike kill Iran's late Supreme Leader Ayatollah Khamenei in early March 2026?
TempUpd	As of 2026-03-12, who holds the role of Supreme Leader of Iran?
CompRel	Was the current Supreme Leader of Iran born after the end of World War II?
ImpRel	What major geopolitical events happened in early March 2026 in the Middle East?
BoundAbs	Did the UN Security Council adopt Resolution 2715 in late February 2026 demanding a ceasefire?
Code Changelogs	
DirAcq	Does NumPy have a function at `numpy.strings.slice`?
RefRes	What function in NumPy extracts substrings from an array of strings using start, stop, and . . . ?
CompRel	Which function fits in the masked snippet below to extract a substring of specific . . . ?
ImpRel	What functions are in `numpy.strings`?
Personalization	
RefRes	*(after the user mentioned solving crosswords and logic puzzles in passing)* What sort of brain-teasing pastime does the user enjoy in their spare time?
TempUpd	*(after the user expressed enthusiasm for African history documentaries earlier and later said they were tired of the genre)* Which documentary gift option should be avoided when narrowing down the shortlist?
CompRel	*(after the user said they enjoy action-adventure games and prefer VR over flat-screen play)* Which game suits this user best — VR action-adventure, non-VR action-adventure, VR puzzle, or non-VR puzzle?
ImpRel	*(after the user mentioned a low tolerance for strong spice)* Which dinner spot should be suggested: a gently seasoned Mediterranean bistro or a chili-forward Sichuan restaurant?
BoundAbs	*(the user has never said anything about gardening)* What kind of gardening setup do I keep at home?

Figure 2. **Selected test queries.** Each row is a representative item generated by our pipeline for a given setting and question type. Each setting emphasizes a different subset of the question types based on the structure of its updates. We provide detailed statistics of each setting in Section C.

directly (such as its timing, topic, or salience) rather than by naming it (e.g., "what happened around the world this week?"). Without ImpRel, a model is useful only to a user who already knows which fact to ask for.

Reasoning over Updates. Recall asks whether a fact can be retrieved, but knowledge also moves along a temporal axis. *Temporal update* (**TempUpd**) captures this and tests whether the model treats a new fact as superseding the one it replaces—not by discarding and overwriting the old fact, but by ordering the two in time and answering from whichever currently holds. Besides the temporal axis, a model should also be able to integrate multiple facts at once; *Compositional reasoning* (**CompRel**) tests this, asking whether the model can reason *with* an update rather than only about it, combining it with other new facts or prior knowledge to reach conclusions that were never stated (e.g., in personalization, recommending a gift might require combining a hobby the user mentioned with a budget they set weeks earlier).

Knowing When Not to Answer. *Boundary awareness* (**BoundAbs**) complements the capabilities above; rather than asking what a model can answer, BoundAbs tests whether the model abstains when no exposed evidence supports a confident answer. Update streams are always incomplete, so many queries cannot be answered from the exposed evidence, and a model that answers anyway will fabricate facts it was never given. A good method should therefore keep a model well-calibrated about both what it knows and where its knowledge ends.

2.2. Domain Settings

We construct IMPRINTBENCH around three realistic settings: dynamic world knowledge (World News), tool/library updates (Code Changelogs), and evolving user states (Personalization). Each comes with a refreshable pipeline that automatically builds a corpus and evaluation questions for the capabilities above; we defer scraping, filtering, and validation details to Section B and statistics to Section C. See Figure 2 for example questions in each setting.

World News. Updates consist of real-world events that postdate a model's knowledge cutoff, requiring the model to continually revise the entities and relationships it must reason about. We anchor updates to spikes in high-volume prediction markets on Polymarket, collect contemporaneous news coverage, and ground relevant entities in a Wikipedia-derived relationship graph.

Code Changelogs. Updates are concrete, verifiable changes to software APIs. We extract these from the changelogs of five widely-used Python libraries (`numpy`, `pandas`, `polars`, `pytorch`, and `scipy`) and pair each with real usage found via GitHub code search.

Personalization. Updates are facts about an individual's preferences and life events that accumulate and change over a long history. We build our questions from long-horizon personalization data (Jiang et al., 2025; Li et al., 2026) by casting each dialogue as a stream of user-state updates and tracking the current state for evaluation purposes.

3. Experimental Setup

We compare all methods under matched conditions so that differences in accuracy reflect the method rather than discrepancies in data or compute. Each method is supervised by one of three sources: the raw update corpus (C), synthetic QA pairs generated from it following prior work (Padmanabhan et al., 2026) (Q), or both (C+Q).[1] In contrast to prior self-distillation evaluations (Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026), we include C+Q to compute- and data-match for methods such as self-distillation, which itself trains on both the corpus and its QA pairs. We further sweep effective compression ratios for in-context and parametric methods alike, and run our main evaluations on Qwen3-30B-A3B-Instruct-2507 and targeted ablations on Qwen3-4B-Instruct-2507, both instruct-tuned with native context windows long enough to host the in-context baselines below.

3.1. Baselines

We describe each baseline below, with hyperparameters in Section D and further implementation details in Section D.2.

Retrieval-Augmented Generation (RAG). We evaluate three retrieval-augmented generation variants spanning a spectrum of retrieval fidelity. **Oracle RAG** prepends the exact context used to generate each query, providing an upper bound against which the other variants can be calibrated. **Hybrid RAG** retrieves the top-k de-duplicated documents into the prompt using BM25 (Robertson & Zaragoza, 2009), dense retrieval, and cross-encoder reranking. **Agentic RAG** equips the model with a retrieval tool that invokes the same hybrid retriever up to k times. We sweep $k \in \{3, 5, 10\}$ for Hybrid and Agentic RAG to characterize the relationship between per-step retrieval budget and answer quality.

Summarization. As the full corpus in each setting exceeds standard context windows, we use a fixed chunk size of 32,768 tokens and prompt the model to summarize into {512, 1,024, 2,048} tokens to simulate 64×, 32×, and 16× compression ratios along the token dimension. In practice, these ratios vary due to prompting variance; see Table 1 for further details. As a finer-grained alternative, we also produce one summary per document (per-feature on Code Changelogs, per-article on World News); see Table 11.

[1] We define C and Q for each setting in Table 8.

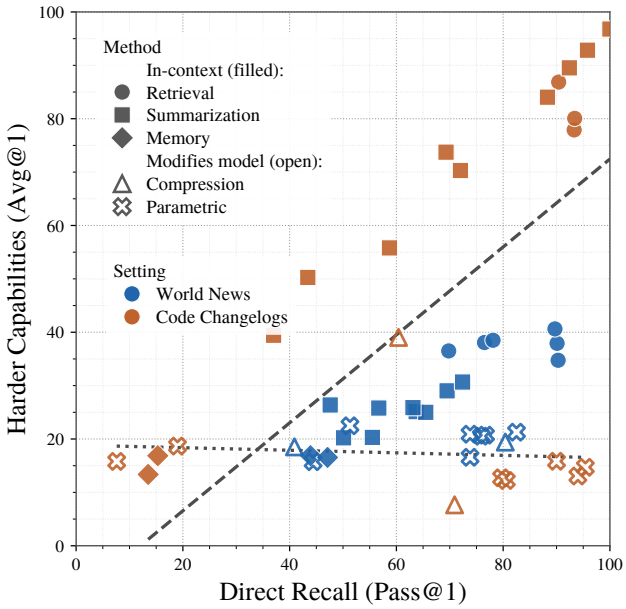

Figure 3. **DirAcq is a weak predictor of internalization for parametric methods.** Each marker is one method pooled over World News and Code Changelogs (personalization has no DirAcq and is omitted). Color denotes setting, marker shape method family, and fill style whether the method is in-context (filled) or modifies models weights or cache (open). x-axis is DirAcq Avg@1, and y-axis is Avg@1 over all other question types. Dashed and dotted lines are least-squares fits over the in-context and parametric methods. In-context methods gain on the harder capabilities as recall rises, while parametric methods reach comparable recall yet stay low.

Memory Scaffolding. We implement a core-memory-only variant of **MemGPT** (Packer et al., 2024), in which the backbone LLM processes update chunks sequentially and rewrites a setting-specific structured memory under a fixed token budget, with sections for active, superseded, and uncertain facts. We evaluate two compression levels, 32× and 64× smaller than the original corpus. At test time, the model answers using only this structured memory.

Context Compression. We study three distinct mechanisms of context compression: **textual summarization via prompting**, **KV-cache compression**, and **prefix-tuning**. We select KVzip (Kim et al., 2025) and Cartridges (Eyuboglu et al., 2025) as our latter two methods. To scale beyond the context window, we follow each method's original recipe: KVzip uses its default fixed chunk size of 2,048 tokens, and Cartridges is trained on the corpus chunked into its constituent documents. We use a compression ratio of 16× for KVzip. For Cartridges, we train a fixed 4,096-token prefix with a different compression ratio for each corpus.

Parametric Training. We evaluate three parametric training objectives: **next-token prediction (NTP)** on the exposed corpus (C), **supervised fine-tuning (SFT)** on gener-

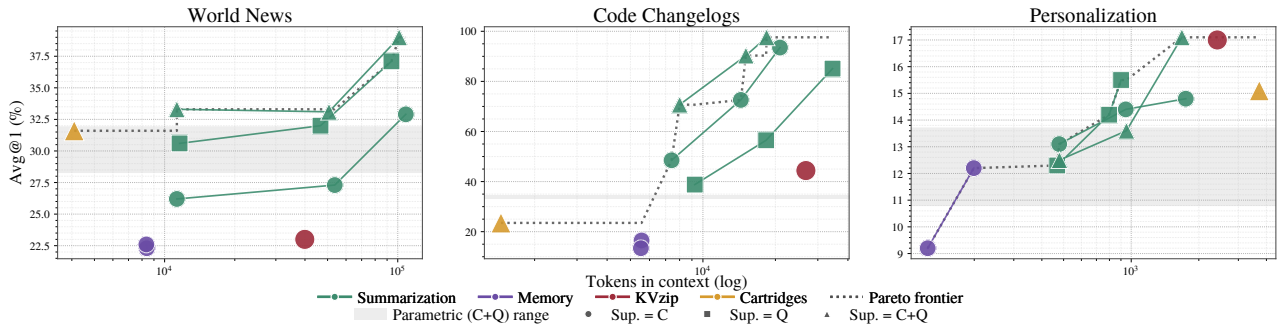

Figure 4. **Context management scales poorly.** Higher compression (moving left) simulates the regime where corpora no longer fit in context; Pass@1 falls sharply as the in-context budget shrinks. Gray band: SFT/SDFT (C+Q) Pass@1 range.

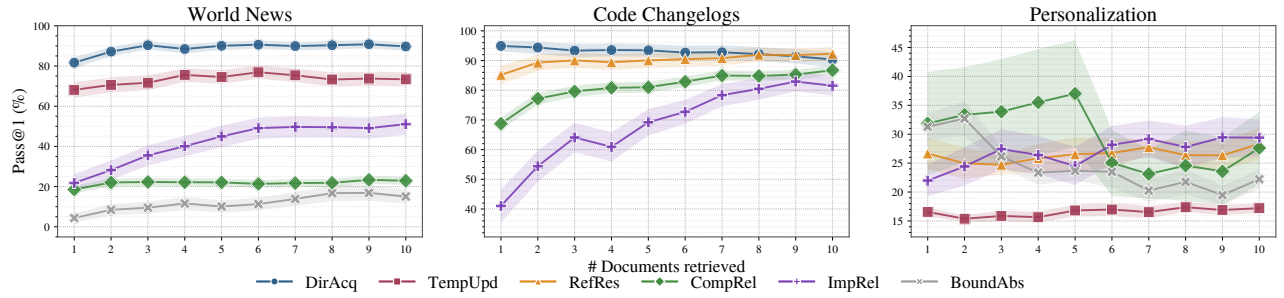

Figure 5. **Pass@1 vs. retrieval depth, per question type and setting.** As the number of retrieved documents grows from 1 to 10, coverage-bound capabilities improve (ImpRel in every setting, CompRel on Code Changelogs), direct and reference recall are already saturated at a single document, and BoundAbs stays low throughout.

ated question-answer pairs (Q), and **self-distillation (SDFT)** (Shenfeld et al., 2026). For the first two objectives, we consider full-parameter fine-tuning and low-rank adaptation (LoRA) (Hu et al., 2022). To provide a matched baseline for self-distillation, we also train a C+Q variant that combines next-token prediction on the corpus with supervised fine-tuning on the generated questions. Following Shenfeld et al. (2026), we train with dense per-token supervision from the same model with privileged information dependent on the setting. We train the student on its answer tokens by minimizing reverse KL to the teacher's next-token logits at each position. On Code Changelogs, we additionally compare against finer-grained LoRA adapters (one per library) in Table 11.

4. Results

We evaluate every method on IMPRINTBENCH against the six capabilities of Section 2 under the matched conditions of Section 3, scoring each (method, setting, question type) by Avg@1 (average accuracy over 8 rollouts) and the stricter AllCorrect@8. We analyze these results by capability and method family below. Full per-method, per-question-type Avg@1/AllCorrect@8 numbers across all three settings on Qwen3-30B-A3B-Instruct-2507 are reported in Table 1. We include a smaller-base ablation for Qwen3-4B-Instruct-2507 in Table 2.

DirAcq Predicts Internalization Only for In-Context Methods. Figure 3 plots each method's DirAcq recall against its aggregate Avg@1 over the remaining (non-DirAcq) capabilities. For in-context methods the two rise together (Spearman $\rho \approx 0.9$), while for parametric methods recall is largely uninformative ($\rho = 0.31$ and -0.25, both insignificant). The gap is starkest at matched recall, where methods scoring 88–100 on Code Changelogs DirAcq still span 13–97% on the harder capabilities, with training-based methods at the bottom (SDFT recalls 94.0% yet averages 13.1, against 96.8 for a compute-matched summary at 100%). This indicates shallow parametric encoding—in context a fact sits verbatim and serves every downstream use, whereas a weight update records its surface form without a level of integration that other forms of reasoning require.

In-Context Methods Prevail Only at Large Token Budgets. Figure 4 sweeps the in-context token budget by varying corpus compression rates. Every in-context family behaves similarly, with accuracy tracking the token budget and falling 20–40 points as compression tightens from $\sim$15$\times$ to $\sim$80$\times$, and the compression mechanism mattering little.

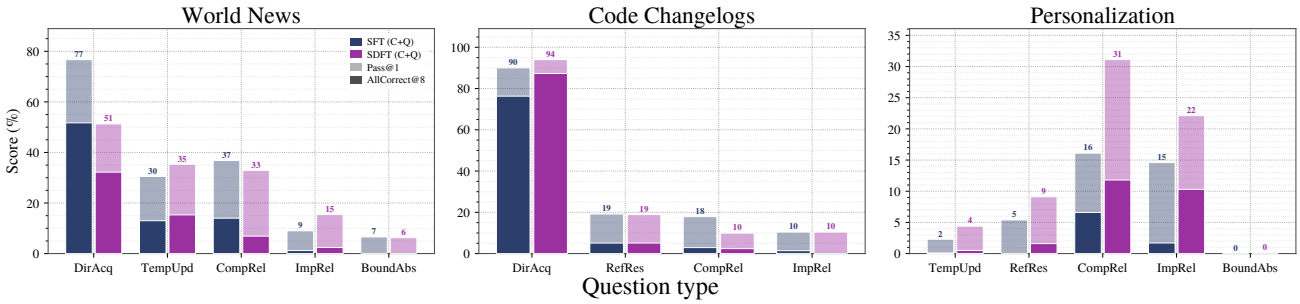

Figure 6. **SFT (C+Q) vs SDFT (C+Q) per question type.** Solid: Pass@1; dashed: AllCorrect@8. Error bars are ±1 SE on World News and Code Changelogs.

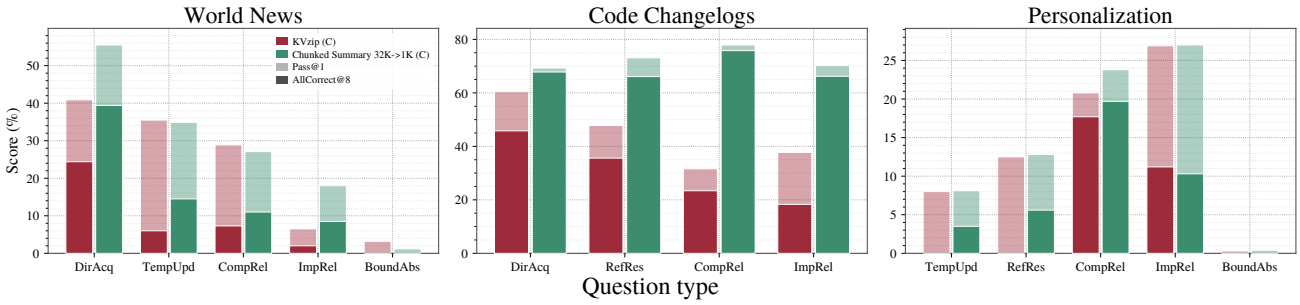

Figure 7. **KVzip underperforms Chunked Summarization at a higher token-budget.** To compute-match, we use a 32× chunked summarization against a ∼16× KVzip compression.

Parametric methods store the update in weights and stay flat across corpus size. Notably, the two regimes cross: in-context methods win only at large token budgets, and their gains erode as compression tightens until the frontier meets the parametric band, with weaker in-context methods falling below it on World News and Personalization. Method performance is thus strongly dependent on corpus size, and the in-context lead seen on corpora that fit does not survive the compression that realistic corpora force.

Retrieval Depth Helps Coverage But Not Abstention. Retrieval recovers facts well but stumbles on several capabilities (Table 1). Although a natural remedy is to widen the retrieval budget, we ask whether retrieving more documents truly closes these gaps. Figure 5 sweeps retrieval depth from one to ten documents. The question types respond in three distinct ways: DirAcq and RefRes are already saturated at a single document, as one relevant passage contains the fact, while ImpRel climbs in both settings and CompRel climbs on Code Changelogs—these queries are coverage-bound, requiring many documents to surface all the relevant evidence (World News ImpRel rises from 22 to 51). Also note that BoundAbs never improves with depth and stays near zero on World News, as confirming that a fact does not exist would require retrieving essentially the whole corpus. Retrieval's failures are thus structural rather than budget-limited: more documents help only where the bottleneck is coverage, and no budget buys abstention.

Dedicated Compression Does Not Beat Naive Baselines. KVzip and Cartridges are engineered to preserve a model's behavior at a fraction of the tokens, so we ask whether they beat naive baselines at a matched budget. We compare KV-cache compression (KVzip, Figure 7) and auxiliary-parameter compression (Cartridges, Figure 8) against compute-matched baselines that share the same supervision and a comparable token budget: Chunked Summary 32K→1K (C) for KVzip (∼16×), full-parameter SFT (C+Q) for Cartridges on World News/Code Changelogs (fixed 4096-token prefix), and Chunked Summary 32K→2K (C+Q) for Cartridges on Personalization (where the per-persona cartridge is only ∼10×). Neither method shows an advantage: on World News and Personalization both land within a few points of the naive baseline in either direction, and on Code Changelogs they collapse, where the harder capabilities dominate (Cartridges averages 23.5 against SFT's 34.4, and KVzip 44.4 against summarization's 72.6). At a matched budget, then, these specialized methods offer little over a summary or SFT. As with context management, what a method achieves is set by its token budget, not by the mechanism that produces it.

Self-Distillation vs. SFT Under Matched Data. Self-distillation is proposed as a stronger way to write updates into weights, especially for continual learning. We test whether self-distillation retains this advantage under matched supervision. We do so via SFT on the same corpus

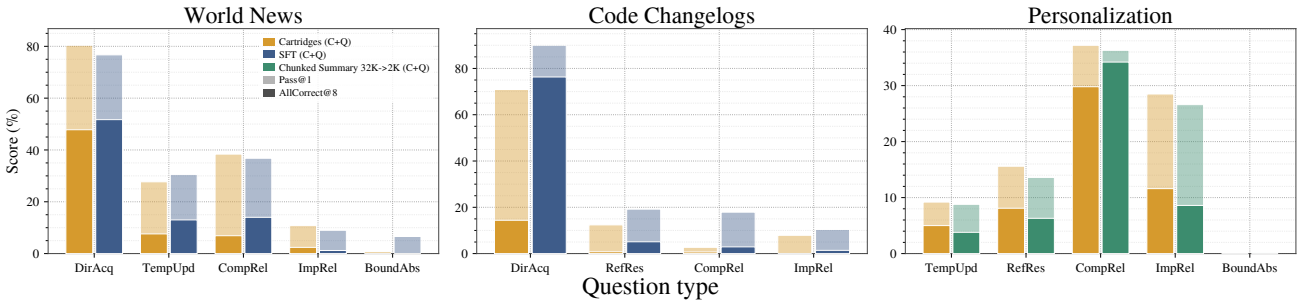

Figure 8. **Cartridges underperforms full-parameter SFT (C+Q).** Cartridges' fixed 4096-token prefix is compute-matched to SFT zero-budget on the larger settings. On Personalization, where the per-persona cartridge is only $\sim$10$\times$ compression, we use a more aggressive 16$\times$ summary baseline that still wins.

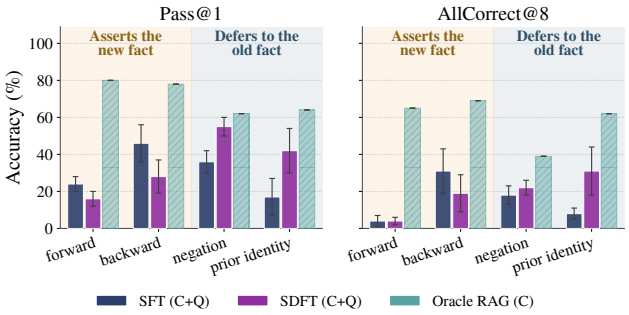

Figure 9. **World News TempUpd split into four temporal axes.** *Forward*: "who holds role X?" answer is the new entity. *Backward*: "what role does the old entity hold?" answer is the new role. *Negation*: "does the old entity still hold X?" answer is no. *Prior identity*: "what role did the old entity hold before?" Forward and backward axes assert the new fact; negation and prior-identity axes defer to the old fact. Oracle RAG (C) is an in-context reference. Left: Pass@1; right: AllCorrect@8.

and QA pairs (C+Q) that self-distillation consumes, and compare Avg@1 performance per question type (Figure 6). Under this matched supervision, the two track within a few points on World News and Code Changelogs—SDFT is even 26 points worse on World News DirAcq (51 vs 77)—and SDFT keeps a 5–15 point edge only on small-corpus Personalization (CompRel 16→31, ImpRel 15→22). We thus hypothesize that self-distillation's reported advantage largely reflects the extra data it trains on rather than the objective itself, and holds only when the corpus is small.

Further, although the two methods score similarly on average, we show that they differ in how they handle old versus new knowledge. We examine this by splitting World News TempUpd questions according to whether the answer is the new fact or the one it replaced and find a clean trade-off shown in Figure 9. SFT wins when the model must commit to the new entity (forward, backward), while SDFT wins when it must still operate on the old one (negation, prior identity). SFT therefore overwrites toward the update while SDFT protects what came before, and no single objective does both, as both trail the in-context Oracle RAG reference on every axis.

Backbone Scale Matters Most for Reasoning-Heavy Methods. Although we hold the backbone fixed throughout, a natural question is how much each method's performance depends on the backbone's own capabilities. To test this, we rerun World News with the smaller Qwen3-4B-Instruct-2507 in place of Qwen3-30B-A3B-Instruct-2507 and include our results in Figure 10. The drop is uneven. In-context retrieval is nearly backbone-invariant (Vanilla RAG barely moves), while the largest gaps fall on Agentic RAG and parametric SFT/SDFT, the methods that lean on the backbone's own reasoning and capacity rather than on retrieved text. Method comparisons therefore shift with backbone scale, and the methods that do their own reasoning are the ones that most need a capable base.

5. Discussion

We introduce IMPRINTBENCH, an evaluation suite for continual learning across three real-world update regimes (world news, code APIs, and personalized assistants) with six question families that probe how a model treats new information beyond rote recall. For training-based methods, direct recall is a weak predictor of internalization; such methods memorize a new fact yet fail to integrate it for indirect reference, composition, or temporal reasoning, so recall alone should never be used to rank methods. Retrieval-based methods on the other hand fail to scale for negative existence queries, context management approaches (such as summarization and memory) lose 20–40 points of Pass@1 at realistic compression ratios, and context compression methods (KVzip, Cartridges) fail to beat compute-matched baselines. Parametric methods (SFT, SDFT) trail in-context methods at the largest token budgets but become competitive only as compression tightens. Within parametric methods, SFT matches self-distillation given matched training data, though the two diverge along temporal axes: SFT commits to the new entity, while SDFT preserves the old.

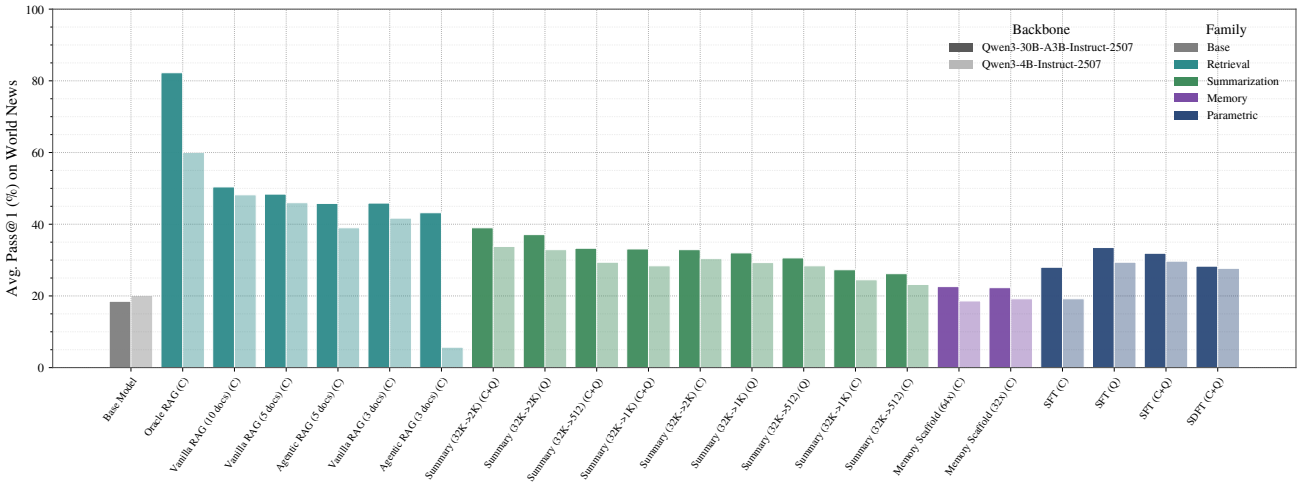

Figure 10. **Backbone comparison on World News.** Qwen3-30B-A3B-Instruct-2507 (solid) vs Qwen3-4B-Instruct-2507 (faded), per method (averaged across question types). The smaller backbone trails on most methods, with the largest gaps on Agentic RAG and parametric SFT/SDFT.

In-context methods dominate while the update corpus still fits the context window, but as it grows and forces aggressive compression, parametric training becomes the better choice, since its accuracy is independent of corpus size. Abstention remains weak across all families, with most methods answering rather than abstaining when the evidence is absent. Closing the abstention gap and moving beyond surface-level memorization remain the central open problems. We hope IMPRINTBENCH serves as a stress test for the next generation of update-time learning systems.

Related Work. Keeping a deployed model current spans several families. Parametric updates write new knowledge into weights, but supervised fine-tuning forgets prior tasks (Kalajdzievski, 2024), LoRA limits this at a cost to capacity (Hu et al., 2021; Biderman et al., 2024), and self-distillation reduces forgetting further (Shenfeld et al., 2026). In-context approaches instead place new information in the prompt, from few-shot prompting (Brown et al., 2020) to retrieval-augmented generation (Lewis et al., 2020; Gao et al., 2024), though they degrade over long contexts (Liu et al., 2024; Li et al., 2024). Between these extremes, memory scaffolds (Wang et al., 2024; Das et al., 2024; Tack et al., 2024; Chen et al., 2026) and KV-cache or adapter compression (Kim et al., 2025; Charakorn et al., 2026; Eyuboglu et al., 2025) store updates compactly, while test-time training adapts the model on the fly (Sun et al., 2024; Tandon et al., 2025). Recent benchmarks (Li & Goyal, 2025; Dou et al., 2026; Wu et al., 2024; Maharana et al., 2024) find that these methods still struggle to internalize new facts and reason over them, which we aim to address via IMPRINTBENCH.

Limitations and Future Work. Our corpora exceed a typical context window but not by orders of magnitude, so we do not stress the months- or years-long accumulation regimes where in-context families would be further disadvantaged and parametric or hybrid approaches may dominate outright. Our study also centers on two open Qwen3 backbones, and trends may shift for larger or differently trained models. The regime boundary we observe motivates hybrid methods that retrieve while the corpus is small and fold updates into weights as it grows. Progress will also require training objectives that integrate new facts rather than record their surface form, and that keep models calibrated to abstain when evidence is absent. Because our pipeline is refreshable, IMPRINTBENCH can serve as a contamination-free, longitudinal testbed as new models and methods appear.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Biderman, D., Portes, J., Ortiz, J. J. G., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Charakorn, R., Cetin, E., Uesaka, S., and Lange, R. T. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902*, 2026.

Chen, Y., Chen, R., Yi, S., Zhao, X., Li, X., Zhang, J., Sun, J., Hu, C., Han, Y., Bing, L., et al. Msa: Memory sparse attention for efficient end-to-end memory model scaling to 100m tokens. *arXiv preprint arXiv:2603.23516*, 2026.

Das, P., Chaudhury, S., Nelson, E., Melnyk, I., Swaminathan, S., Dai, S., Lozano, A., Kollias, G., Chenthamarakshan, V., Jiří, Navrátil, Dan, S., and Chen, P.-Y. Larimar: Large language models with episodic memory control, 2024. URL https://arxiv.org/abs/2403.11901.

Devoto, A., Jeblick, M., and Jégou, S. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

Dou, S., Zhang, M., Yin, Z., Huang, C., Shen, Y., Wang, J., Chen, J., Ni, Y., Ye, J., Zhang, C., et al. Cl-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026.

Eyuboglu, S., Ehrlich, R., Arora, S., Guha, N., Zinsley, D., Liu, E., Tennien, W., Rudra, A., Zou, J., Mirhoseini, A., et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jiang, B., Hao, Z., Cho, Y. M., Li, B., Yuan, Y., Chen, S., Ungar, L., Taylor, C. J., and Roth, D. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=6ox8XZGOqP.

Kalajdzievski, D. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*, 2024.

Kim, J.-H., Kim, J., Kwon, S., Lee, J. W., Yun, S., and Song, H. O. Kvzip: Query-agnostic kv cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*, 2025.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, A. O. and Goyal, T. Memorization vs. reasoning: Updating llms with new knowledge. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25853–25874, 2025. doi: 10.18653/v1/2025.findings-acl.1326. URL https://aclanthology.org/2025.findings-acl.1326/.

Li, S. S., Paranjape, B., Oktar, K., Ma, Z., Zhou, G., Guan, L., Zhang, N., Park, S., Chen, L., Yang, D., Tsvetkov, Y., and Celikyilmaz, A. Horizonbench: Long-horizon personalization with evolving preferences, 2026. URL https://arxiv.org/abs/2604.17283.

Li, T., Zhang, G., Do, Q. D., Yue, X., and Chen, W. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Padmanabhan, S., Gul, M. O., and Goyal, T. Updating parametric knowledge with context distillation retains post-training capabilities, 2026. URL https://arxiv.org/abs/2602.16093.

Robertson, S. and Zaragoza, H. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3 (4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/

1500000019. URL https://doi.org/10.1561/1500000019.

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Tack, J., Kim, J., Mitchell, E., Shin, J., Teh, Y. W., and Schwarz, J. R. Online adaptation of language models with a memory of amortized contexts. *Advances in Neural Information Processing Systems*, 37:130109–130135, 2024.

Tandon, A., Dalal, K., Li, X., Koceja, D., Rød, M., Buchanan, S., Wang, X., Leskovec, J., Koyejo, S., Hashimoto, T., et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Wang, Y., Gao, Y., Chen, X., Jiang, H., Li, S., Yang, J., Yin, Q., Li, Z., Li, X., Yin, B., et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2024. URL https://arxiv.org/abs/2410.10813.

Ye, T., Dong, L., Wu, X., Huang, S., and Wei, F. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Zhao, S., Xie, Z., Liu, M., Huang, J., Pang, G., Chen, F., and Grover, A. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

A. Full Results Tables

We report all main results for Qwen3-30B-A3B-Instruct-2507 below in a single table. We re-run the World News pipeline with Qwen3-4B-Instruct-2507 as the base model and parametric finetune target (Table 2). Compare against the Qwen3-30B-A3B-Instruct-2507 numbers in Table 1.

B. Pipeline Design

We describe the pipeline in detail for each of our settings.

B.1. World News

In World News, each test instance is a corpus of recent Polymarket-spike news articles spanning a fixed time window, paired with held-out questions about the events and entities described in those articles. A method is trained on the corpus and is then evaluated on the held-out questions. Adapter methods receive only the question at evaluation time, so success requires internalizing corpus facts rather than retrieving them from an explicit test context.

Corpus Construction. Articles are sourced by scraping the URLs flagged in Polymarket spike events: for each market spike, the referenced URLs are pulled through the Exa fetch API and stored verbatim under `articles/<market_id>/spike_<i>/<url_hash>.txt`. A Qwen3-32B cleaning pass strips navigation, ads, sidebars, and metadata while preserving article body text. After cleaning, the corpus is semantically deduplicated to the canonical 678-article set used by all evaluations.

Question Generation. Each generator runs on the cleaned, deduplicated corpus.

1. **DirAcq.** For each article, the generator produces ten boolean (Yes/No) questions: roughly five anchored to true article claims and five plausible falsehoods that swap exactly one element (entity, date, quantity, or relationship). Each question is required to be self-contained, with an explicit date anchor, fully named entities, and an unambiguous scope so it cannot be answered from the question alone.

2. **TempUpd.** A pre-extraction pass over each article identifies (entity_before, entity_after) update pairs (e.g., a role transfer) and classifies the update as either a named-entity swap (TYPE-A) or a state/value flip (TYPE-B). Per TYPE-A pair, four temporal axes are produced from the canonical article: T1 *forward* ("As of `<date>`, who holds `<role>`?" — answer is the new entity); T2 *negation* ("As of `<date>`, is `<old>` still `<role>`?" — answer is No); T3 *prior identity* ("Beyond `<new>`'s `<role>`, what is `<new>` otherwise known for?" — answer is the prior role; emitted only when the prior role is grounded in corpus text, to prevent fabrication); and T4 *backward* ("As of `<date>`, who replaced `<old>` as `<role>`?" — answer is the new entity). TYPE-B pairs receive only T1 and T2.

3. **CompRel.** Compositional questions are constructed over groups of markets that spiked at the same timestamp from different events. Five sub-families are generated: 1-hop compositional (combining a corpus fact with pretraining knowledge); counterfactual ("If `[event]` had not happened, would `[consequence]`?" — answer flips); indirect probing (broad topical questions where the new fact must surface unprompted); temporal coherence (before/during/after consistency for the same event); and subject aliasing (the same fact under alternative entity names or paraphrastic surface forms).

4. **ImpRel.** Per spike, the generator emits broad-strokes indexing questions of the form "What major `<plural category>` happened in `<scope>` `<time>`?". Gold answers are required to be a single high-level named event arc (not a list, not a sub-fact). Across the corpus, candidate answers are then embedded with `bge-large-en-v1.5` and clustered at cosine similarity 0.82; one canonical question is retained per cluster, and the cluster size is carried forward as a salience signal.

5. **BoundAbs.** For each of the largest indexing clusters, the generator produces plausible "No"-answer boundary questions using four strategies: actor-action absence, subevent absence, quote absence, and consequence absence. Each candidate is then embedded, the top-K most similar articles are retrieved, and a judge labels every retrieval as *affirm*, *deny*, or *unrelated*. The verdict counts are stored alongside the question as a difficulty annotation.

	Method	Sup.	Compr.	DirAcq	TempUpd	RefRes	CompRel	ImpRel	BoundAbs	Avg.
World News	Base Model		–	30.9 15.6	30.3 12.2	–	24.7 9.9	0.0 0.0	6.8 0.0	18.5 7.5
	Oracle RAG	C	–	85.6 73.9	80.2 58.8	–	71.7 58.2	91.5 65.9	–	–
	Vanilla RAG (3 docs)	C	54.4×	79.9 71.1	59.7 39.7	–	38.8 18.6	39.5 26.8	3.0 0.0	44.2 31.3
	Vanilla RAG (5 docs)	C	32.5×	82.8 72.2	61.2 39.7	–	39.1 22.7	48.0 28.0	3.5 0.0	46.9 32.5
	Vanilla RAG (10 docs)	C	16.5×	85.2 72.2	60.7 38.9	–	39.9 23.2	52.3 37.8	4.2 0.0	48.5 34.4
	Agentic RAG (3 docs)	C	226.8×	69.8 48.3	62.2 42.7	–	38.1 13.8	42.7 25.6	3.0 0.0	43.2 26.1
	Agentic RAG (5 docs)	C	135.9×	76.5 53.9	62.8 41.2	–	36.5 15.1	48.8 23.2	4.2 0.0	45.8 26.7
	Agentic RAG (10 docs)	C	67.9×	78.1 56.7	62.8 36.6	–	34.1 11.7	52.7 28.0	4.4 0.0	46.4 26.6
	Memory Scaffold (32×)	C	76.2×	43.9 30.0	28.5 12.2	–	26.2 11.0	12.3 3.7	0.5 0.0	22.3 11.4
	Memory Scaffold (64×)	C	76.5×	47.1 32.2	28.8 9.2	–	25.3 8.7	11.9 7.3	0.2 0.0	22.6 11.5
	KVzip	C	16×	40.9 24.4	35.5 6.0	–	28.9 7.3	6.5 2.0	3.2 0.0	23.0 8.0
	Cartridges	C+Q	156.0×	80.4 47.8	27.7 7.6	–	38.4 6.9	10.8 2.4	0.8 0.0	31.6 13.0
	Chunked Summary (32K→2K)	C	5.9×	63.7 48.3	42.9 17.6	–	28.2 8.2	28.8 12.2	0.6 0.0	32.9 17.2
	Chunked Summary (32K→1K)	C	11.9×	55.5 39.4	34.9 14.5	–	27.1 11.0	18.0 8.5	1.2 0.0	27.3 14.7
	Chunked Summary (32K→512)	C	56.6×	50.1 37.2	32.1 15.3	–	29.2 13.5	19.1 14.6	0.5 0.0	26.2 16.1
	SFT	C	–	73.8 50.6	24.6 8.4	–	35.2 9.2	4.7 0.0	1.8 0.0	28.0 13.6
	+ LoRA	C	–	44.4 35.0	30.5 16.8	–	31.8 14.3	0.5 0.0	0.2 0.0	21.5 13.2
	Chunked Summary (32K→2K)	Q	6.8×	69.5 52.8	45.3 21.4	–	35.6 13.3	32.9 22.0	2.3 0.0	37.1 21.9
	Chunked Summary (32K→1K)	Q	13.7×	56.7 41.7	42.1 19.1	–	33.5 11.7	26.4 17.1	1.2 0.0	32.0 17.9
	Chunked Summary (32K→512)	Q	55.1×	47.6 33.9	43.2 26.7	–	33.3 14.0	26.7 18.3	2.3 0.0	30.6 18.6
	SFT	Q	–	82.5 57.8	33.5 14.5	–	39.3 17.6	7.2 0.0	5.1 0.0	33.5 18.0
	+ LoRA	Q	–	75.9 38.3	33.9 15.3	–	39.6 9.4	5.6 0.0	3.0 0.0	31.6 12.6
	Chunked Summary (32K→2K)	C+Q	6.3×	72.4 53.3	46.8 24.4	–	32.8 10.5	40.5 25.6	2.6 0.0	39.0 22.8
	Chunked Summary (32K→1K)	C+Q	12.6×	65.5 47.2	39.6 16.0	–	31.0 9.2	27.9 19.5	1.5 0.0	33.1 18.4
	Chunked Summary (32K→512)	C+Q	56.6×	63.1 48.9	40.3 18.3	–	32.2 13.8	29.6 22.0	1.4 0.0	33.3 20.6
	SFT	C+Q	–	76.7 51.7	30.5 13.0	–	36.8 14.0	9.0 1.2	6.6 0.0	31.9 16.0
	+ LoRA	C+Q	–	73.8 36.1	30.0 11.5	–	41.1 11.0	7.8 0.0	4.7 0.0	31.4 11.7
	SDFT	C+Q	–	51.3 32.2	35.3 15.3	–	32.9 6.9	15.4 2.4	6.3 0.0	28.3 11.4
Code Changelogs	Base Model		–	7.8 5.1	–	12.2 5.1	1.4 0.5	7.2 0.0	–	7.2 2.7
	Oracle RAG	C	245.2×	98.3 96.0	–	90.5 72.0	99.6 99.3	99.8 98.6	–	97.0 91.5
	Vanilla RAG (3 docs)	C	54.4×	91.5 92.4	–	90.7 82.2	80.3 78.9	50.0 35.2	–	78.1 72.2
	Vanilla RAG (5 docs)	C	32.5×	93.2 93.2	–	90.7 82.2	82.9 80.3	53.4 38.0	–	80.1 73.4
	Vanilla RAG (10 docs)	C	16.5×	90.7 89.8	–	93.2 85.6	88.2 85.6	67.8 64.8	–	85.0 81.5
	Memory Scaffold (32×)	C	77.5×	13.5 5.9	–	7.8 0.0	28.6 25.1	3.7 0.0	–	13.4 7.8
	Memory Scaffold (64×)	C	77.0×	15.3 1.7	–	11.9 0.0	36.8 33.7	1.9 0.0	–	16.5 8.9
	KVzip	C	16×	60.4 45.8	–	47.8 35.6	31.6 23.4	37.7 18.3	–	44.4 30.8
	Cartridges	C+Q	295.4×	70.9 14.4	–	12.4 0.8	2.7 0.7	7.9 0.0	–	23.5 4.0
	Chunked Summary (32K→2K)	C	20.5×	95.8 95.8	–	95.7 93.2	95.8 94.7	87.0 77.5	–	93.5 90.3
	Chunked Summary (32K→1K)	C	29.8×	69.3 67.8	–	73.1 66.1	77.9 75.8	70.2 66.2	–	72.6 69.0
	Chunked Summary (32K→512)	C	57.8×	43.4 42.4	–	57.1 50.0	45.8 44.0	47.9 45.1	–	48.5 45.4
	SFT	C	–	19.0 5.1	–	33.0 13.6	16.5 6.0	6.5 0.0	–	18.8 6.2
	+ LoRA	C	–	7.6 5.9	–	34.4 20.3	1.4 0.0	11.6 5.6	–	13.8 8.0
	Chunked Summary (32K→2K)	Q	12.4×	88.3 87.3	–	88.9 83.9	79.2 76.8	84.0 62.0	–	85.1 77.5
	Chunked Summary (32K→1K)	Q	23.4×	58.7 58.5	–	44.1 10.2	60.3 52.4	63.0 57.7	–	56.5 44.7
	Chunked Summary (32K→512)	Q	46.3×	37.0 31.4	–	32.5 11.0	42.7 25.8	43.1 33.8	–	38.8 25.5
	SFT	Q	–	95.4 84.7	–	14.5 5.1	20.8 9.3	8.8 1.4	–	34.9 25.1
	+ LoRA	Q	–	80.6 44.1	–	20.7 5.9	9.7 3.1	6.3 0.0	–	29.3 13.3
	Chunked Summary (32K→2K)	C+Q	23.3×	100.0 100.0	–	98.0 94.9	96.6 95.0	95.8 84.5	–	97.6 93.6
	Chunked Summary (32K→1K)	C+Q	28.4×	92.4 92.4	–	91.8 89.0	90.9 89.7	85.9 69.0	–	90.3 85.0
	Chunked Summary (32K→512)	C+Q	53.5×	72.0 72.0	–	76.4 72.9	61.8 59.8	72.7 60.6	–	70.7 66.3
	SFT	C+Q	–	90.0 76.3	–	19.2 5.1	17.9 2.9	10.4 1.4	–	34.4 21.4
	+ LoRA	C+Q	–	79.7 39.8	–	20.1 5.9	10.9 2.4	7.2 1.4	–	29.5 12.4
	SDFT	C+Q	–	94.0 87.3	–	18.9 5.1	9.9 2.4	10.4 0.0	–	33.3 23.7
Personalization	Base Model		–	–	1.2 –	4.0 –	17.1 –	13.8 –	0.0 0.0	16.3 –
	Oracle RAG	C	–	–	18.9 –	34.1 –	44.7 –	41.4 –	23.1 –	32.4 –
	Vanilla RAG (10 docs)	C	–	–	21.5 7.1	27.0 7.9	27.6 17.1	32.8 10.3	22.3 0.0	23.5 6.8
	Agentic RAG (10 docs)	C	–	–	8.7 –	19.8 –	27.6 –	19.8 –	6.7 –	11.4 –
	Memory Scaffold (32×)	C	194.2×	–	5.7 3.0	13.3 10.3	18.3 17.1	21.2 6.0	2.7 0.0	12.2 7.3
	Memory Scaffold (64×)	C	311.5×	–	3.2 1.6	4.9 3.2	15.8 13.2	19.3 4.3	2.7 0.0	9.2 4.4
	KVzip	C	16×	–	8.0 0.0	12.5 0.0	20.8 17.7	26.9 11.2	0.3 0.0	17.0 10.4
	Cartridges	C+Q	10.4×	–	9.2 5.0	15.6 8.1	37.2 29.8	28.5 11.6	0.0 0.0	15.1 9.1
	Chunked Summary (32K→2K)	C	22.1×	–	9.6 4.0	14.8 5.6	22.0 19.7	27.7 12.1	0.0 0.0	14.8 8.3
	Chunked Summary (32K→1K)	C	40.9×	–	8.1 3.5	12.8 5.6	23.8 19.7	27.0 10.3	0.4 0.0	14.4 7.8
	Chunked Summary (32K→512)	C	80.9×	–	6.1 2.3	7.4 4.8	23.4 17.1	27.9 11.2	0.7 0.0	13.1 7.1
	SFT	C	–	–	1.0 0.0	3.0 0.0	19.7 2.6	12.9 1.7	0.0 0.0	10.3 0.9
	+ LoRA	C	–	–	1.1 0.1	3.4 0.0	20.1 10.5	13.1 4.3	0.0 0.0	7.5 3.0
	Chunked Summary (32K→2K)	Q	42.9×	–	10.5 6.6	17.5 14.3	21.2 18.4	28.0 11.2	0.5 0.0	15.5 10.1
	Chunked Summary (32K→1K)	Q	48.5×	–	9.3 5.2	14.5 10.3	26.0 22.4	20.7 7.8	0.6 0.0	14.2 9.1
	Chunked Summary (32K→512)	Q	82.6×	–	6.0 3.2	8.2 5.6	24.2 22.4	21.9 7.8	1.2 0.0	12.3 7.8
	SFT	Q	–	–	2.8 0.0	9.2 0.8	21.7 7.9	15.2 1.7	0.0 0.0	11.4 2.1
	+ LoRA	Q	–	–	3.5 0.3	10.6 0.8	26.5 13.2	10.9 1.7	0.0 0.0	10.3 3.2
	Chunked Summary (32K→2K)	C+Q	23.0×	–	8.8 3.8	13.6 6.3	36.3 34.2	26.6 8.6	0.2 0.0	17.1 10.6
	Chunked Summary (32K→1K)	C+Q	40.6×	–	6.8 2.6	12.5 6.3	20.7 19.7	27.8 12.9	0.3 0.0	13.6 8.3
	Chunked Summary (32K→512)	C+Q	80.7×	–	5.0 1.8	7.1 4.0	22.5 17.1	26.5 8.6	1.1 0.0	12.5 6.3
	SFT	C+Q	–	–	2.3 0.1	5.4 0.0	16.1 6.6	14.6 1.7	0.0 0.0	10.8 1.7
	+ LoRA	C+Q	–	–	3.0 0.8	7.1 0.8	15.6 9.2	15.5 5.2	0.0 0.0	8.3 3.2
	SDFT	C+Q	–	–	4.4 0.5	9.1 1.6	31.1 11.8	22.1 10.3	0.1 0.0	13.7 4.9

Table 1. Pass@1 / AllCorrect@8 per question type on Qwen3-30B-A3B-Instruct-2507, across the three settings of IMPRINTBENCH. **Sup.** indicates the supervision source (C=raw corpus, Q=synthetic QA pairs, C+Q=both interleaved); see Table 8 for per-setting definitions. **Compr.** is the effective compression of the corpus into context tokens. For each (method, supervision) we report the hyperparameter with the highest average score (excluding DirAcq).

	Method	Sup.	Compr.	DirAcq	TempUpd	RefRes	CompRel	ImpRel	BoundAbs	Avg.
World News	Base Model		–	32.4 18.3	27.9 13.0	–	26.6 11.2	0.0 0.0	13.9 0.0	20.1 8.5
	Oracle RAG	C	–	76.2 66.1	67.3 53.4	–	36.5 19.4	–	–	60.0 46.3
	Vanilla RAG (3 docs)	C	54.4×	70.6 57.8	64.9 43.5	–	34.0 16.8	32.0 19.5	7.2 0.0	41.7 27.5
	Vanilla RAG (5 docs)	C	32.5×	76.9 62.8	63.3 45.8	–	36.3 18.1	46.0 34.1	7.4 0.0	46.0 32.2
	Vanilla RAG (10 docs)	C	16.5×	79.6 63.3	65.6 44.3	–	35.5 18.1	52.9 37.8	7.5 0.0	48.2 32.7
	Agentic RAG (3 docs)	C	226.8×	10.1 0.0	9.1 0.0	–	6.8 0.0	2.3 0.0	0.2 0.0	5.7 0.0
	Agentic RAG (5 docs)	C	135.9×	64.2 40.6	55.5 35.9	–	32.6 14.3	39.8 19.5	2.9 0.0	39.0 22.0
	Memory Scaffold (32×)	C	76.2×	39.7 27.8	23.2 9.9	–	21.1 6.4	9.6 4.9	2.4 0.0	19.2 9.8
	Memory Scaffold (64×)	C	76.5×	40.8 31.1	20.0 5.3	–	18.9 6.6	11.4 4.9	2.0 0.0	18.6 9.6
	Chunked Summary (32K→2K)	C	5.9×	58.6 45.0	38.2 22.9	–	26.4 9.2	23.9 9.8	4.8 0.0	30.4 17.4
	Chunked Summary (32K→1K)	C	11.9×	47.8 37.2	30.7 16.8	–	24.8 9.7	16.0 12.2	3.5 0.0	24.5 15.2
	Chunked Summary (32K→512)	C	56.6×	44.2 33.9	28.0 14.5	–	23.9 7.4	16.0 9.8	3.9 0.0	23.2 13.1
	Chunked Summary (32K→2K)	Q	6.8×	64.0 45.0	42.8 27.5	–	27.8 11.5	26.4 15.9	3.6 0.0	32.9 20.0
	Chunked Summary (32K→1K)	Q	13.7×	51.0 36.7	39.2 20.6	–	27.7 12.5	24.4 18.3	4.4 0.0	29.3 17.6
	Chunked Summary (32K→512)	Q	55.1×	44.2 33.3	40.7 28.2	–	28.8 10.2	23.0 19.5	5.3 0.0	28.4 18.3
	Chunked Summary (32K→2K)	C+Q	6.3×	65.9 48.9	39.8 20.6	–	28.8 8.9	28.5 19.5	5.9 0.0	33.8 19.6
	Chunked Summary (32K→1K)	C+Q	12.6×	58.8 41.7	31.8 16.8	–	25.5 7.9	22.4 11.0	3.5 0.0	28.4 15.5
	Chunked Summary (32K→512)	C+Q	56.6×	52.5 39.4	36.8 18.3	–	27.1 9.7	26.4 19.5	4.4 0.0	29.4 17.4
	SFT	C	–	35.3 21.1	24.1 7.6	–	26.6 6.9	0.0 0.0	9.9 0.0	19.2 7.1
	SFT	Q	–	74.9 46.1	24.1 9.9	–	39.5 10.2	5.5 0.0	3.0 0.0	29.4 13.2
	SFT	C+Q	–	62.6 33.3	34.6 12.2	–	37.6 7.4	9.0 0.0	4.8 0.0	29.7 10.6
	SDFT	C+Q	–	57.8 29.4	24.0 8.4	–	34.2 6.6	7.5 0.0	15.1 0.0	27.7 8.9

Table 2. **World News results, Qwen3-4B-Instruct-2507 base.** Mean Pass@1 / AllCorrect@8 (gray) per question type. Oracle RAG omits ImpRel and BoundAbs (no sensible cluster-level oracle / no abstention oracle). `pending` = generation done, judge in flight; – = generation not yet started.

After generation, all non-BoundAbs candidates pass a two-pass cheatability filter on top of an over-generated 3× candidate budget. A blind no-context judge first removes items that the judge can answer correctly without seeing the article (i.e., items that leak through pretraining knowledge); a with-context judge then removes items that the judge cannot answer correctly even with the source article in context. A per-article question budget proportional to article length controls how many surviving items are kept. BoundAbs candidates are not filtered: every candidate is retained alongside its retrieval verdict counts, so the difficulty distribution is preserved.

All generated answers are judged with a fixed MATCH/NONMATCH/REFUSAL rubric. DirAcq, TempUpd, CompRel, and ImpRel items are correct iff the verdict is MATCH. BoundAbs is scored by abstention behavior: explicit refusals or insufficient-information responses are correct, while a substantive claim — affirming or denying — counts as a false claim. Per-question Pass@1 averages over decoded samples; AllCorrect@8 reports the fraction of items for which all 8 sampled responses are correct. World News has no RefRes family.

B.2. Code Changelogs

In Code Changelogs, each test instance is a newly added Python API in a major numerical/ML library release, paired with one or more questions that probe whether a method has internalized the new identifier. A method is trained on the post-release documentation and changelog corpus for the affected libraries (NumPy, pandas, Polars, PyTorch, SciPy) and is then evaluated on held-out questions. Adapter methods see only the question at evaluation time, so success requires internalizing release content rather than retrieving from an explicit test context.

Corpus Construction. Release notes are pulled from each library's GitHub releases endpoint and split into a full body, a "New Features" section, and a "Backwards Incompatible Changes" section. A markdown bullet parser walks the New Features sections and emits one candidate per leaf bullet, recording bullet text, the enclosing section heading, the cited PR numbers, an extracted primary identifier (preferring backticked tokens, with heuristic fallback for dotted, snake_case, and CamelCase), an optional parent (for parameter additions like "add `weights_only` parameter to `torch.load`"), and a `prompt_kind` of *function* or *parameter*. A heuristic callable filter drops obvious non-callables (env vars, build flags, screaming-case acronyms).

For every surviving candidate, the pipeline fetches each cited PR's metadata, body, and changed-file list, and greps the locally checked-out library source tree for documentation snippets mentioning the identifier. An evidence-grounded LLM judge (Qwen3.6-35B-A3B, multi-trial consensus) then decides, given bullet + PR evidence + doc snippets, whether the

candidate truly introduces a new Python-callable identifier and what its canonical full import path is; passing candidates are renamed to their canonical path, rejected candidates are moved to `_rejected/`. A subsequent leaf-novelty judge (5/5 unanimous required) discards identifiers whose leaf word is merely a port, rename, or alias of an existing API in a sibling namespace.

For each accepted feature, the pipeline runs a paginated GitHub code search (`"<leaf>" language:Python NOT is:fork`), drops library-owned and forked repositories, drops vendored paths, fetches each surviving file directly from `raw.githubusercontent.com`, and rejects files where the only mention of the leaf is its own `def`/`class` site (i.e., the implementation rather than a usage). Each fetched file then passes two LLM audits: a usage-relevance judge that the snippet really uses the new library identifier (not a homograph in a different package), and a self-contained judge that a small illustrative snippet can be cleanly extracted from this file. A final extraction pass produces a 5–30 line minimal snippet per surviving file, preserving the target call site verbatim. A snippet-quality judge with three properties (the snippet actually uses the target library; the masked identifier really refers to the new API rather than a name collision; the gold leaf does not appear verbatim in surrounding comments, docstrings, or variable names) gates the final snippet pool, requiring 5/5 unanimous yes on each property.

Question Generation. The accepted features feed five question families.

1. **DirAcq.** A templated recall question of the form "Does `<library>` have a `<kind>` at `<canonical>`?" is emitted for every non-parameter feature, with gold answer "Yes". This probes whether the model knows the new identifier exists at its canonical path.

2. **RefRes.** A referential question is generated by asking Qwen3.6-35B-A3B to write a description of *what* the new identifier does without naming it, then templating the description into "What `<kind>` in `<library>` does `<description>`?". The gold answer is the leaf identifier; this probes whether the model can recover the identifier from a paraphrase of its intent.

3. **CompRel.** For every quality-passed snippet, the target identifier is masked deterministically (`\b<primary>\b` for callables, `\b<primary>\s*=` for parameters; zero substitutions rejects the snippet). A natural-language question is then generated by an LLM that sees only the masked snippet and the local doc page; system rules prohibit naming the masked identifier or thinly paraphrasing its leaf word, while explicitly permitting irreducible technical terms (e.g., "XCCL", "ONNX"). A faithfulness validator (same model, given masked snippet + question + docs) rejects questions that fabricate operations not present in the code, give the answer away, or fail to correspond to this snippet. Surviving (snippet, question, gold) triples become candidates. A *golden* subset is then assembled by intersecting (i) validator-pass with (ii) Qwen3-30B-A3B-Instruct-2507 scoring 0/8 on the no-context pass@8 baseline (the local backbone genuinely does not know the answer) with (iii) gpt-5.4-mini + `web_search_preview` scoring 8/8 on a pass@8 oracle (the question has a single uniquely correct answer). A harder *no-import* variant is also produced by deterministically stripping namespace prefixes and dead import lines so the leaf identifier must be recovered from surrounding logic alone.

4. **ImpRel.** An indexing question of the form "What `<plural-of-kind>` are in `<parent_namespace>`?" is emitted per non-parameter feature, with the leaf as gold. The parent is the canonical path with the leaf stripped (e.g., `pandas.DataFrame.cummin` → parent `pandas.DataFrame`). This probes cluster-level recall: given a containing module or class, can the model name one of its new members?

5. **BoundAbs.** A boundary set is constructed from synthesized non-existent identifiers under each library's namespace (plausible-sounding leaves that never appear in any monitored release). The corresponding recall question ("Does `<library>` have a `<kind>` at `<fictitious-canonical>`?") has gold answer "No"/abstention.

Filters are applied per family. The CompRel golden filter (validator-pass ∩ baseline-0/8 ∩ oracle-8/8) is the strictest and is the source of the published Code Changelogs evaluation set; weaker variants (validator-pass only, or validator-pass ∩ oracle-8/8) are retained for sensitivity analyses. DirAcq and ImpRel items are templated and have no LLM filter; RefRes items are gated by the same canonical-path and leaf-novelty judges that admit features into the corpus.

All generated answers are judged with a fixed MATCH/NONMATCH/REFUSAL rubric. DirAcq, RefRes, CompRel, and ImpRel items are correct iff the verdict is MATCH; for CompRel a trailing-component equality check is applied first

(e.g., `F.conv2d` ≡ `torch.nn.functional.conv2d`) before falling through to the LLM judge for any remaining ambiguity. BoundAbs is correct iff the model abstains or denies the existence of the fictitious identifier. Pass@1 averages over decoded samples; AllCorrect@8 reports the fraction of items for which all 8 sampled responses are correct. Code Changelogs has no TempUpd family.

B.3. Personalization

In Personalization, each test instance is one PersonaMem (Jiang et al., 2025) user. A method is trained on that user's personalization data and is then evaluated once on held-out questions about the same user. Adapter methods receive only the question at evaluation time, so success requires internalizing the user's facts rather than retrieving them from an explicit test context.

Question Generation. The generator starts from PersonaMem's user facts and conversation evidence. Each usable fact is converted into a semantic frame using Prompt E.4; the frame contains the normalized claim, tracked attribute, value, optional rejected value, polarity, domains, constraint dimension, task affordances, decision levers, and banned surface forms. This frame is the input to the family-specific question-generation prompts.

1. **DirAcq/RefRes.** For each PersonaMem fact, the pipeline decides whether the paired user turn states the fact directly. Lexical overlap ≥ 0.8 is direct; otherwise an LLM labels the pair as direct, nearby, implicit, or unrelated using Prompt E.4. Direct facts produce a DirAcq question asking for the tracked attribute; indirect facts produce an RefRes question asking for the same normalized claim through paraphrastic wording. Direct facts may also receive an RefRes paraphrase probe. Both DirAcq and RefRes questions are generated with Prompt E.4, which returns a question and expected answer.
2. **TempUpd.** For PersonaMem update records, the generator creates questions from the old and new preference values. A NEW-INFO question asks whether the model knows the newly introduced user fact and is generated with Prompt E.4; a CHANGED-INFO question asks for the current value, is constructed so the old and new preferences would lead to different answers, and is generated with Prompt E.4.
3. **BoundAbs.** The generator samples PersonaMem facts that should not be available to the learner under the evaluated training view, including privacy/forget-style facts where applicable, and uses Prompt E.4 to ask a natural question that would require that fact. The stored gold behavior is abstention rather than a factual answer.
4. **ImpRel.** The generator builds a pool of semantically framed PersonaMem facts with task affordances such as meal choice, schedule choice, trip choice, writing assistance, or support strategy. It groups facts by affordance. For each candidate target fact, it selects distractor facts from the same user's fact set, preferring distractors that share the affordance, constraint dimension, or domain, so that the decision is plausibly confusable. It then calls Prompt E.4 with the target frame, distractor frames, affordance, and banned surface forms. The LLM returns a realistic assistant scenario, a decision variable, one user-facing question, and a gold answer. The question must ask what should be suggested, chosen, planned, avoided, or kept in mind; it must not name the user fact or any distractor fact. The gold answer is the downstream recommendation or fit judgment, not the fact itself.
5. **CompRel.** The generator groups the user's framed facts by shared task affordance and enumerates two- or three-fact tuples with distinct constraint dimensions and distinct tracked attributes. For each tuple, it uses a generic stem, typically "Which option is the best fit for this user?", and calls Prompt E.4 to construct a multiple-choice decision. The emitted options must instantiate a truth table: for two facts, $[T, T]$, $[F, T]$, $[T, F]$, $[F, F]$; for three facts, $[T, T, T]$, $[F, T, T]$, $[T, F, T]$, $[T, T, F]$. The gold option is the only one satisfying all required facts, and each distractor violates a different required fact while remaining plausible for an unknown user.

After generation, all non-BoundAbs candidates are filtered for answer leakage and generic answerability. The blind no-context check in Prompt E.4 and the forced-commit check in Prompt E.4 reject items that can be answered without user state. Retrieval items also pass the DirAcq/RefRes contract audit in Prompt E.4. ImpRel items pass the fact-list oracle in Prompt E.4, the application-family audit in Prompt E.4, and the universality audit in Prompt E.4. CompRel items pass the fact-list oracle in Prompt E.4, the no-context option-bias audit in Prompt E.4, and the matrix/family audit in Prompt E.4.

All generated answers are judged with the fixed MATCH/NONMATCH/REFUSAL rubric in Prompt E.4. DirAcq, RefRes, TempUpd, ImpRel, and CompRel are correct iff the verdict is MATCH; for multiple-choice items the reference answer is the

gold option letter. BoundAbs is scored by abstention behavior: explicit refusals or insufficient-information responses are correct, while user-specific claims are false claims. Metrics are computed per family and per persona, then averaged over personas with at least one item in the family. We also retain subtype scores for TempUpd, ImpRel, and CompRel.

C. Benchmark Statistics

C.1. Corpus Statistics

Table 3 summarizes the corpus statistics for both settings. All token counts are computed with the Qwen3-30B-A3B-Instruct-2507 tokenizer. For Code Changelogs, "Article" refers to the human-readable documentation aggregated for an accepted feature (release-note bullet, full docs page, PR metadata, summary), excluding the raw patches and diffs that are still appended during NTP training. For World News, "Article" is the cleaned news-article body (`article.txt`).

	World News	Code Changelogs	Personalization
Articles	678	176	50
Total corpus tokens	638,828	428,282	1,933,409
Article tokens (mean)	942	2,433	38,668
Self-study training Q+A	16,667	18,400	5,000
Q+A pairs per article	24.6	104.5	100.0
Total question tokens	563,555	395,090	77,666
Total answer tokens	200,831	180,429	79,167
Question / Answer tokens (mean)	33.8 / 12.0	21.5 / 9.8	15.5 / 15.8
Total C+Q tokens	1,403,214	1,003,801	2,090,242
C+Q tokens (mean)	2,070	5,703	41,805

Table 3. Training corpus statistics, all token counts via the Qwen3-30B-A3B-Instruct-2507 tokenizer. "Article" = one news article (World News) / one accepted feature with its bullet, docs, PR metadata and summary, excluding raw patches (Code Changelogs) / one persona-chunk pair (Personalization, 10 PersonaMem-v2 personas with K=1 chunk each + 10 HorizonBench personas with K=4 chunks each = 50 chunks total).

Month	Articles	Article Tokens	Tok / Art.
2026-02	446	454,835	1020
2026-03	1,023	1,069,595	1046

Table 4. World News corpus broken down by month of publication.

Library	Features	Article Tokens	Tok / Art.
`numpy`	4	27,507	6877
`pandas`	21	96,641	4602
`polars`	52	223,409	4296
`pytorch`	52	555,396	10681
`scipy`	47	276,448	5882

Table 5. Code Changelogs corpus broken down by library. "Article" = one accepted feature with bullet, docs, PR metadata, and patch.

C.2. World News: by month

Table 4 breaks the World News corpus down by month of publication.

C.3. Code Changelogs: by library

Table 5 breaks the Code Changelogs corpus down by library.

Source		Chunks	Article Tokens	Tok / Art.
PersonaMem-v2	(10 personas, K=1)	10	424,910	42,491
HorizonBench	(10 personas, K=4)	40	1,508,405	37,710

Table 6. Personalization corpus broken down by source. "Article" = one persona-chunk pair: a self-contained span of conversational history that the model is supposed to internalize. PersonaMem-v2 personas are static (K=1 chunk per persona, all chats packed into one chunk); HorizonBench personas are streaming (K=4 chunks per persona).

Question type	# Questions	Avg. supporting tokens
World News		
DirAcq	180	1007
ImpRel	82	—
TempUpd	131	572
CompRel (2-hop)	149	591
CompRel (3-hop)	150	582
CompRel (4-hop)	93	535
BoundAbs	83	—
Code Changelogs		
DirAcq	118	7285
RefRes	118	7285
CompRel	1,155	10228
ImpRel	118	7285
Personalization		
DirAcq	100	38,666
TempUpd	1,111	38,666
RefRes	126	38,666
CompRel	129	38,666
ImpRel	171	38,666
BoundAbs	255	38,666

Table 7. Number of evaluation questions per question type, with the average number of corpus tokens needed to answer each question (sum over the article(s) the question depends on, mean across questions of that type). BoundAbs questions reference no article and are excluded from the supporting-tokens average.

C.4. Evaluation Statistics

Table 7 reports the number of evaluation questions per question type for each setting, along with the mean number of corpus tokens needed to answer each question (sum of tokens across all articles the question depends on, averaged within question type). ImpRel questions reference cluster-level information rather than a single article and are reported with a dash; BoundAbs questions reference no article in the corpus by construction. Figure 11 visualizes the same per-question-type counts.

C.5. Generation Statistics

Figure 12 reports the average number of tokens pulled into the model's context window per question, per method, on Code Changelogs. Figure 13 reports the per-method distribution of generation lengths on Code Changelogs, broken out per question type with the corresponding Pass@1 shown as a faint bar in the background.

D. Experiment Details

D.1. Supervision Sources (C and Q) per Setting

The **Sup.** column in every results table encodes each method's supervision: C = supervised by the raw *Corpus*, Q = supervised by the synthetic SFT *QA pairs*, and C+Q = both (interleaved). Table 8 defines what C and Q are concretely for each setting; per-setting token statistics are in Table 3.

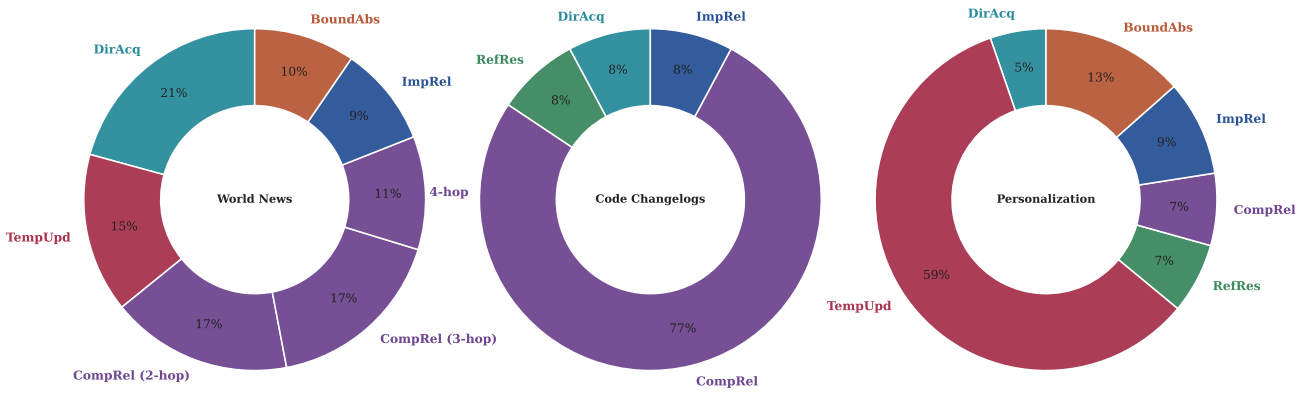

Figure 11. Question-type distribution of the evaluation set: World News (left), Code Changelogs (middle), and Personalization (right). Shorthand labels: DirAcq, TempUpd, RefRes, CompRel, ImpRel, BoundAbs. CR is split by hop count for World News.

	C — Corpus	**Q — Synthetic QA pairs**
World News	Cleaned news-article body for each Polymarket-linked event.	Q+A pairs distilled from each article.
Code Changelogs	Per-feature aggregation: release-note bullet, full documentation page, introducing PR metadata, and a natural-language summary; raw patches/diffs are additionally appended for NTP.	Q+A pairs about the API change, covering direct-recall, reference-resolution, and downstream-usage formats.
Personalization	Persona-chunk pair: a self-contained span of one user's conversational history.	Q+A pairs distilled from the facts observed in the chunk (preferences, life events, sensitive information, relations, interests).

Table 8. Per-setting definitions of the supervision sources C and Q used in the **Sup.** column of every results table. C+Q interleaves both.

D.2. Training Hyperparameters

Common training recipe. All learnable methods (NTP, SFT, NTP+SFT, SDFT, Cartridges) share the following recipe across all three backbones ({Qwen3-30B-A3B-Instruct-2507, Qwen3-4B-Instruct-2507 }): AdamW with β_1=0.9, β_2=0.999, ϵ=10^{-8}, and zero weight decay; a cosine learning-rate schedule with no warmup; 2 training epochs; gradient clipping at maximum norm 1.0; and bfloat16 precision throughout.

Per-method differences. Table 9 lists only the hyperparameters that actually vary across methods.

	NTP	**SFT**	**NTP + SFT**	**SDFT**	**Cartridges**
Effective batch size		64		64	16
LR (full FT)		{1e-5, 5e-6, 1e-6}		{5e-6, 1e-5, 5e-5}	2e-2
LR (LoRA)		{1e-5, 5e-5, 1e-6}		—	—
Corpus chunked length	512	—	512	—	1024
Max completion length	—	—	—	256	256
LoRA config	r=16, α=32, dropout 0.1, all-linear			—	—
Cartridge prefix length	—	—	—	—	4,096 tokens
Teacher distillation targets	—	—	—	top-100 logits	top-100 logits

Table 9. Per-method training hyperparameters; all other settings follow the common recipe in §D.2. The three SFT-family supervisions (NTP on the corpus, SFT on synthetic QA pairs, and the interleaved NTP+SFT variant over the entire dataset) share the same LR sweep, with both full-parameter and LoRA configurations evaluated for each. { } denotes a swept set.

D.3. Evaluation Hyperparameters

We use the same decoding configuration for every method. Per question, we sample n_{trials}=8 generations with temperature 0.7, top-p 0.95, top-k 20, and `max_new_tokens` = 4096. Generation-length statistics will be reported in a forthcoming

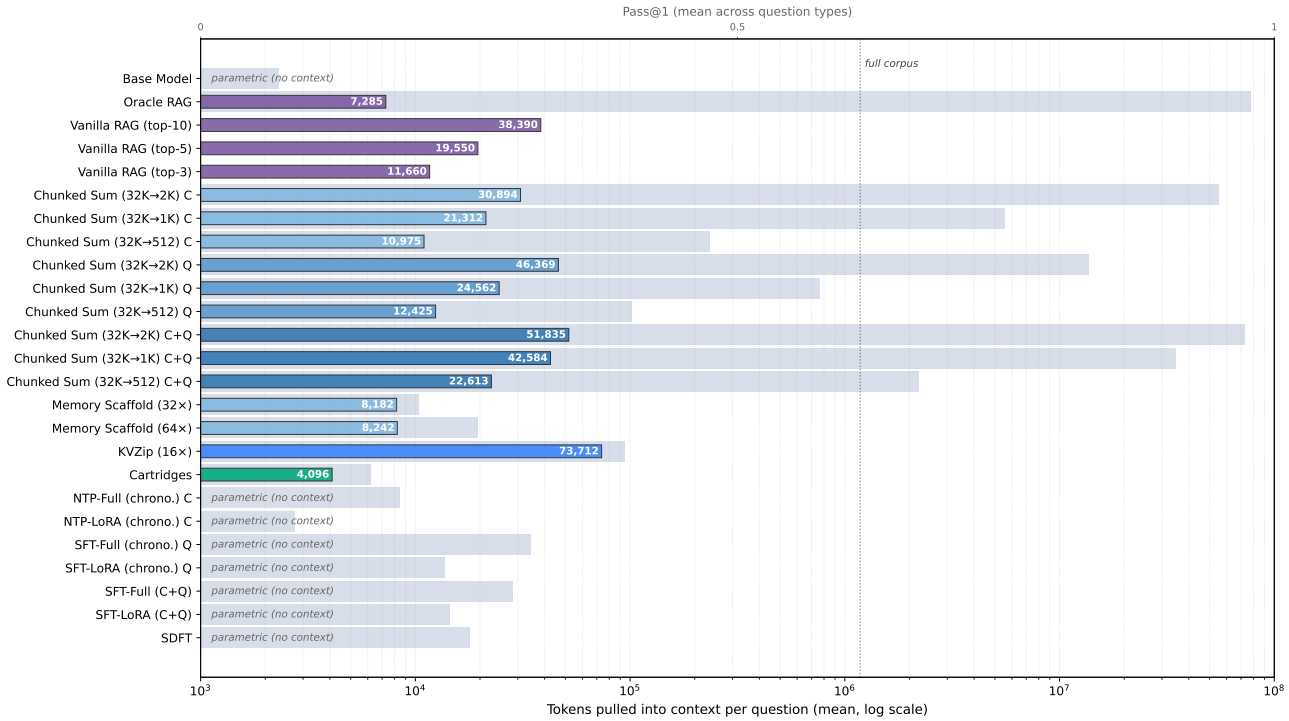

Figure 12. Mean tokens pulled into context per question on Code Changelogs (log scale). For most context-based methods the prefix is constant across question types (a single global summary, memory, or KV cache); for Oracle RAG and Vanilla RAG it varies per question and we report the mean. Parametric methods (Base Model, NTP/SFT/SDFT) inject nothing into context and are shown only via their accuracy band. The light-gray background bars show mean Pass@1 across question types (top axis).

revision.

Each generation is judged against the gold answer; the judge model, prompt, and decision protocol differ by question type. Per-question-type definitions, example items, and full judge prompts are deferred to `<question-type appendix placeholder>`. We report:

- **Pass@1**: mean fraction of correct generations across the 8 trials.
- **Pass@8**: fraction of questions for which at least one of the 8 generations is correct.
- **AllCorrect@8**: fraction of questions for which all 8 generations are correct.

D.4. Compression Hyperparameters

Chunked Summary and Memory Scaffold both lean on a frontier API model to compress the corpus into a global text prefix that is reused at evaluation time, but differ in granularity (per-chunk vs. accumulating-global) and budget interpretation. Table 10 lists their settings side-by-side.

KVzip. We use the KVPress repository (Devoto et al., 2025) and use the default chunk-size of 2048.

D.5. Per-feature / per-library baselines on Code Changelogs

The default Summarization and Parametric baselines on Code Changelogs in Table 1 train one model (or one summary) on the entire corpus, batched into 32K-token packed chunks (Tables 9 and 10). Table 11 reports finer-grained alternatives: a per-feature summary (one summary per accepted feature, instead of per packed chunk) and per-library LoRA adapters (one adapter per library, vs. one adapter trained chronologically on the unified corpus).

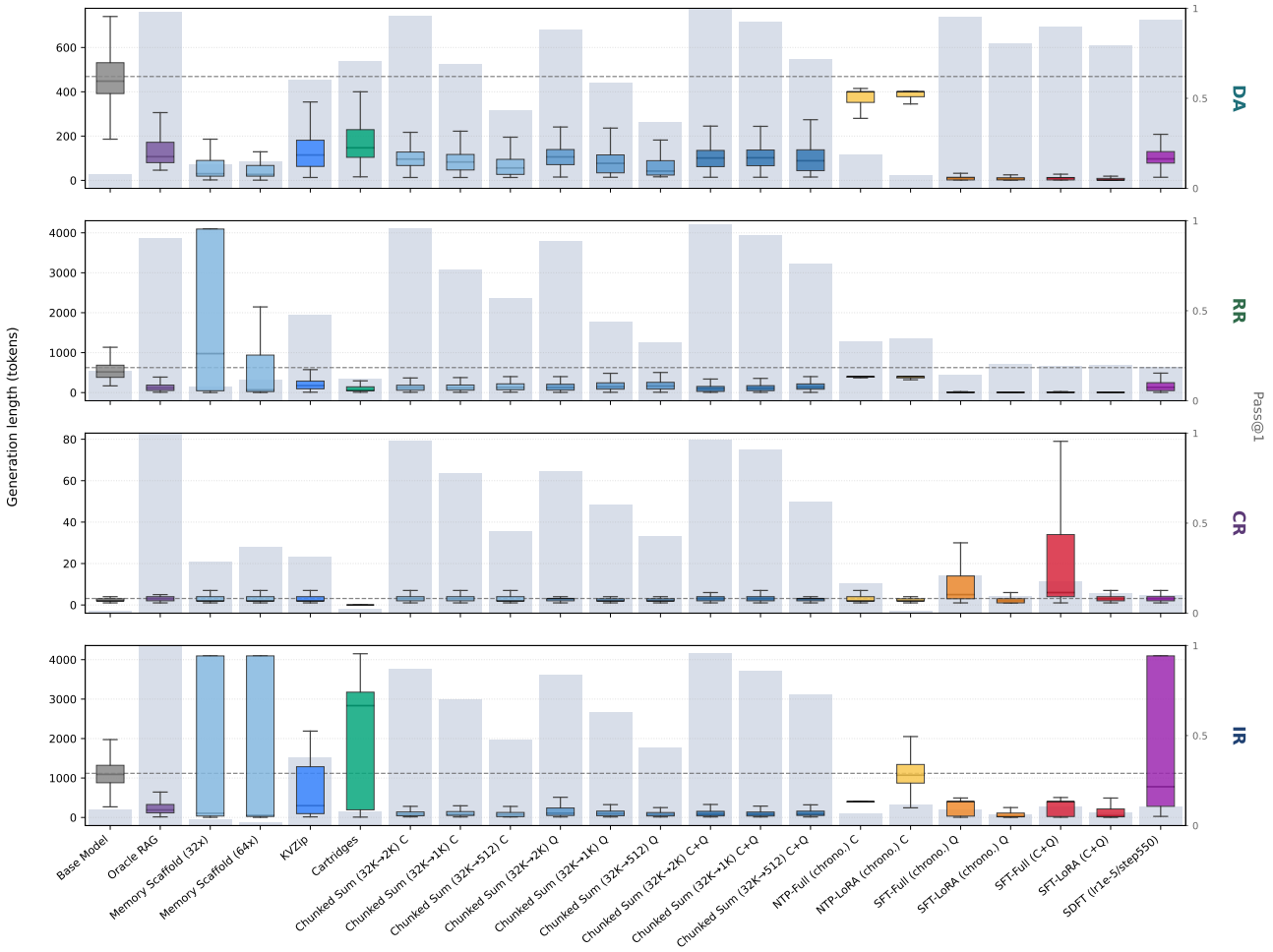

Figure 13. Per-method generation length on Code Changelogs (Qwen3-30B-A3B-Instruct-2507), one row per question type (DirAcq, RefRes, CompRel, ImpRel). Box = IQR, whiskers = 1.5× IQR, median bar in black, outliers omitted. The dashed horizontal line marks the base-model mean for that question type; the light-gray background bars show Pass@1 (right axis). Methods are ordered to mirror Table 1

E. Prompt Templates

This section contains prompts for question-answer generation and for the LLM-judge prompt used for each (setting, question-type) cell in our evaluation. Slots populated at evaluation time are written as ⟨*name*⟩; gold fields differ by question type (e.g. World News TempUpd scores against `answer_after_update` rather than `answer`). Each box title names the judge model.

E.1. Question Generation

E.2. World News

World News uses a hybrid judge: DirAcq and BoundAbs items have deterministic structure (binary or abstention), so we use a *regex judge* that doesn't call any LLM. The remaining three families (TempUpd, CompRel, ImpRel) use a shared Y/N LLM judge.

We initially used the LLM judge for all five families; spot-checking revealed that on DirAcq specifically (where gold is always Yes/No) the LLM judge over-penalized responses that began with the correct yes/no but then meandered into off-topic ramble — a common failure mode for compressed-context methods like Chunked Summary and KVzip. The regex judge, which inspects only the model's first standalone yes/no token, is both deterministic and a fairer reflection of

	Chunked Summary	Memory Scaffold
API model	`gpt-5.4-mini`	`gpt-4.1-mini`
Granularity	per 32K-token packed chunk	global memory, sequential updates
Update unit	one chunk → one summary	per-feature/article
Target budget	{2048, 1024, 512} per chunk	total_tok / {32, 64} (global)
Hard cap on output	1.25× target	—
Max completion tokens	1.5× target	$\min(2 \cdot \text{budget}, 8192)$

Table 10. API-based compression baselines. Both methods produce a static text prefix that is concatenated as the global system context at evaluation time. Empirical compression ratios are reported in Table 1.

	Method	Sup.	Compr.	DirAcq	TempUpd	RefRes	CompRel	ImpRel	BoundAbs	Avg.
Code Changelogs	Base Model		–	7.8 5.1	–	12.2 5.1	1.4 0.5	7.2 0.0	–	7.2 2.7
	Summary (per feat.)	C	15.1×	98.7 95.2	–	97.4 96.8	96.5 95.0	99.8 98.6	–	98.1 96.4
	Summary (per feat.)	Q	10.6×	100.0 100.0	–	98.8 95.8	93.1 92.1	100.0 100.0	–	98.0 97.0
	SFT	C	–	19.0 5.1	–	33.0 13.6	16.5 6.0	6.5 0.0	–	18.8 6.2
	+ LoRA (per lib)	C	–	6.1 4.0	–	33.5 22.2	1.7 0.2	12.0 5.6	–	13.3 8.0
	+ LoRA (all)	C	–	7.6 5.9	–	34.4 20.3	1.4 0.0	11.6 5.6	–	13.8 8.0
	SFT	Q	–	95.4 84.7	–	14.5 5.1	20.8 9.3	8.8 1.4	–	34.9 25.1
	+ LoRA (per lib)	Q	–	82.5 69.0	–	18.7 7.9	14.8 7.3	6.2 1.4	–	30.5 21.4
	+ LoRA (all)	Q	–	80.6 44.1	–	20.7 5.9	9.7 3.1	6.3 0.0	–	29.3 13.3
	SDFT	C+Q	–	94.0 87.3	–	18.9 5.1	9.9 2.4	10.4 0.0	–	33.3 23.7

Table 11. **Granular (per-feature, per-library) baselines.** Code Changelogs on Qwen3-30B-A3B-Instruct-2507. **Sup.**: C = raw Corpus, Q = synthetic QA pairs, C+Q = both; see Table 8. "(per lib)" LoRA = one adapter per library; "(all)" = one adapter trained chronologically on the unified all-library corpus. We report mean Pass@1 / AllCorrect@8 (strict).

binary-classification accuracy. The same fix applies to BoundAbs, where every gold begins with "No, the corpus contains no record of…" and the model passes whenever it abstains (says "no", or any of a fixed set of abstention phrases).

World News — DirAcq judge (regex; no model call)

DirAcq golds are exclusively in {`Yes`, `No`} (180/180 items). For each generated response we scan for the first standalone yes/no word (the regex `\b(yes|no)\b`, case-insensitive) and the response scores *Y* iff that token matches the gold. If no yes/no token appears in the response, the response scores *N*.

World News — BoundAbs judge (regex; no model call)

Every BoundAbs gold begins with "No, the corpus contains no record of…" (83/83 items) — the desired model behavior is principled abstention. We score *Y* if either: (i) the response contains any of a fixed list of abstention phrases in its first 600 characters — "insufficient information", "no record of", "not in my reference", "not mentioned", "not covered", "no information about", "cannot find", etc.; or (ii) the first standalone yes/no token in the response is "no". Otherwise *N*.

World News — shared LLM judge (`gpt-5.4-nano`); used for TempUpd/ CompRel/ ImpRel

System.
```
You judge whether an LLM's RESPONSE is consistent with the GOLD answer to a
QUESTION. Reply with ONLY one letter:
  Y = consistent (model's answer is the same as gold modulo phrasing)
  N = inconsistent or no answer attempted
Reply with ONLY the single letter Y or N.
```

User.
```
QUESTION: ⟨question⟩
GOLD: ⟨gold⟩
RESPONSE: ⟨response⟩
Y or N?
```

World News — TempUpd judge

Uses the shared LLM judge above. `gold` is populated from `answer_after_update` so that responses reflecting the *old* state score N.

World News — CompRel judge

Uses the shared LLM judge above; `gold` is populated from `answer_after_update`. Applied uniformly across 2/3/4-hop compositional items.

World News — ImpRel judge

Uses the shared LLM judge above; `gold` is populated from `answer` (cluster-level canonical summary).

E.3. Code Changelogs

Code Changelogs — DirAcq judge (gpt-5.4-nano)

```
You are an evaluation judge. Given a question, a reference answer, and a model's
response, classify the model's response into exactly one category.

Question: ⟨question⟩
Reference answer: ⟨expected_answer⟩
Model response: ⟨trained_answer⟩

Categories:
- MATCH: The model's response contains the key factual claim from the reference answer.
Different wording is fine, but the core fact must be present and correct. Partial
correctness does not count.
- NONMATCH: The model's response is factually wrong, vague, off-topic, or missing the
key claim from the reference answer.
- REFUSAL: The model explicitly declines to answer, says it doesn't know, lacks
information, or cannot answer the question.

Think and then reply with exactly one word: MATCH, NONMATCH, or REFUSAL.
```

Code Changelogs — RefRes judge (gpt-5.4-nano)

System. `You are a strict identifier-matching judge for Python library APIs. Given a GOLD identifier and a MODEL prediction, answer YES iff the prediction refers to the SAME Python identifier as the gold. Allow trailing-component dotted matches (e.g.` torch.foo `matches` foo`), trailing call notation (e.g.` foo `matches` foo() `or` foo(x)`), and minor whitespace differences. Otherwise answer NO. Reply with ONLY the single word YES or NO.`

User. `GOLD:` ⟨*gold*⟩
`PREDICTION:` ⟨*prediction*⟩
`YES or NO?`

Code Changelogs — CompRel judge (gpt-5.4-nano)

`Same identifier-matching rubric as the Code Changelogs RefRes judge above. CompRel presents the model with a masked Python snippet (one identifier replaced by` <MASK>`) and a natural-language description of intent; the judge scores whether the model's filled-in identifier resolves to the same canonical identifier as gold under the matching rules above. Generation prompt:`

System. `You are a Python completion tool. Given a question and a Python snippet with` <MASK> `in place of a single library identifier, reply with ONLY the identifier that should go in` <MASK>`. No code fences, no explanation, no quotes -- just the identifier as it would appear in code.`

User. `Question:` ⟨*question*⟩
`Code with one missing identifier:`
“‘python
⟨*masked*⟩
“‘
`The identifier that belongs at` <MASK> `is:`

Code Changelogs — ImpRel judge (gpt-5.4-nano)

System. You are checking whether an LLM's free-form list answer contains a specific Python identifier. Given a target leaf identifier and the model's response, answer YES iff the response explicitly names the target identifier (or a trailing-dotted-component match like module.target for target) as one of the listed items / functions / classes / methods. Reply with ONLY YES or NO.

User. TARGET: ⟨*gold*⟩
RESPONSE: ⟨*response*⟩
Does the response explicitly list the TARGET identifier?

E.4. Personalization

Personalization — fact-frame annotation

You are compiling a user fact into a semantic frame for a personalization benchmark.

Fact: "⟨*preference*⟩"
Category: "⟨*pref_type*⟩"
Topic: "⟨*topic*⟩"

Build a compact frame that can support both retrieval questions and downstream application questions.

Return valid JSON with: normalized_claim, attribute, value, rejected_value, polarity, domains, constraint_dimension, task_affordances, decision_levers, and surface_forms.

Personalization — explicitness classifier

You are labeling a user turn from a personalization dataset. A preference record is paired with the user turn in which the preference first appears in a conversation.

Preference: "⟨*preference*⟩"
User's message: "⟨*user_turn*⟩"

Classify how the user's message reveals this preference.
A -- DIRECT: the user states or closely paraphrases the preference as their own.
B -- NEARBY: the user mentions the subject/topic but does not assert it as their own.
C -- IMPLICIT: the preference is only inferable from behavior, writing style, or produced content.
D -- UNRELATED: the user's message does not meaningfully reveal this preference.

Reply with exactly one letter: A, B, C, or D.

Personalization — DirAcq/RefRes question generation

You are compiling a personalization benchmark question from a semantic user-fact frame.

Source fact frame:
⟨*fact_frame*⟩

Capability contract:
⟨*contract*⟩

Family-specific guidance:
⟨*family_guidance*⟩

Generate one candidate that satisfies the contract exactly. DirAcq/RefRes questions ask what is true about the user, not what to recommend or do. The expected answer must be equivalent to the normalized claim. Do not state the correct value in the question. DirAcq should be direct about the tracked attribute or domain; RefRes should be paraphrastic and avoid surface forms. Keep the question one natural sentence.

Reply with only valid JSON: {"question": "...", "expected_answer": "..."}.

Personalization — TempUpd new information

```
You are generating a test question for a personalization benchmark.

A PersonaMem user has the following preference record:
"⟨preference⟩"

Category: ⟨pref_type⟩
Topic: ⟨topic⟩

Generate one natural, concise question that tests whether a system knows this
information about the user. The question must be answerable only from this preference
and must not hint at the answer. Provide a concise expected answer.

Reply with only valid JSON: {"question": "...", "expected_answer": "..."}.
```

Personalization — TempUpd changed information

```
You are generating a test question for a personalization benchmark.

A user's preference has changed:
OLD preference: "⟨old_preference⟩"
NEW preference: "⟨new_preference⟩"
Topic: ⟨topic⟩

Generate one natural, concise question that tests whether the system knows the current
preference. The question should have clearly different answers under the old and
new preferences and must not hint at either preference. Provide the expected answer
reflecting the new preference.

Reply with only valid JSON: {"question": "...", "expected_answer": "..."}.
```

Personalization — BoundAbs question generation

```
You are generating a test question for a personalization benchmark.

A user has a preference or life event that their personal assistant does not know
about:
"⟨fact⟩"

Generate one natural, concise question that would require knowing this fact to answer
correctly. A system without this knowledge should say that it does not know or lacks
enough information.

Reply with only valid JSON: {"question": "..."}.
```

Personalization — blind no-context check

```
Answer the following question about a user. If you don't have enough information, say
so. Be concise.

Question: ⟨question⟩
```

Personalization — forced-commit no-context check

```
You are answering a question about an unknown user. You have no information about this
user, but you must still provide a specific best-guess answer.

Question: ⟨question⟩

Give a single concrete answer, even if highly uncertain. Do not say that you do not
know, cannot determine, have no information, or need more information. Be concise.
```

Personalization — DirAcq/RefRes contract audit

```
You are auditing one personalization benchmark question.

Source fact frame:
⟨fact_frame⟩

Desired family: ⟨family⟩
Candidate question: ⟨question⟩
Candidate expected answer: ⟨answer⟩

DirAcq asks what is true about the user in direct attribute/domain language. RefRes
asks what is true about the user in paraphrastic or abstract language. ImpRel asks for
a downstream consequence, not the fact. CompRel asks an applied decision that requires
combining multiple facts.

Return whether this candidate satisfies the desired family contract. Reply with only
valid JSON containing passes, label, question_intent, answer_matches_fact, and reason.
```

Personalization — ImpRel generation

```
You are compiling an Implicit Relevance question for a personalization benchmark.

Target fact frame, which the question must require applying:
⟨target_frame⟩

Distractor fact frames, known about the user but irrelevant:
⟨distractor_frames⟩

Downstream task affordance: ⟨affordance⟩
Banned words/phrases: ⟨banned_forms⟩

Invent a realistic assistant scenario in which the target fact changes the best action.
Write one natural user-facing question about what should be suggested, chosen, planned,
avoided, or kept in mind. The stem must not mention any user attribute, condition,
hobby, schedule, identity, or preference, and must not use scaffolding preambles
such as "given the user's" or "based on". The gold answer must be the downstream
recommendation or fit judgment, not a paraphrase of the fact.

Reply with only valid JSON containing scenario, decision, question, gold_answer, and
explanation.
```

Personalization — ImpRel fact-list oracle

```
You are answering a question about a user. The user has the following facts:

⟨facts_text⟩

Question: ⟨question⟩

Answer concisely. If the facts above do not determine the answer, reply "I don't
know."
```

Personalization — ImpRel application audit

```
You are auditing an Implicit Relevance question for a personalization benchmark.

Target fact(s):
⟨target_facts⟩

Question: ⟨question⟩
Gold answer: ⟨answer⟩

The question must ask what should be suggested, chosen, planned, avoided, or kept in
mind for the user; it must not ask what is true about the user. The gold answer must
be a downstream action or fit judgment, not a restatement of the target fact. Exactly
one target fact should be sufficient to answer.
```

Reply with only valid JSON containing passes, label, question_intent, answer_is_fact, and reason.

Personalization — ImpRel universality audit

You are auditing an Implicit Relevance question for a personalization benchmark. The question is supposed to require knowing a specific user's preference to answer correctly.

Question: ⟨*question*⟩
Gold answer: ⟨*answer*⟩

Suppose you knew nothing about the user. Would the gold answer still be the recommendation a typical assistant would give to most people? In other words, does it coincide with universal best practice or common-sense advice that does not require knowing this user's preferences?

Reply with only valid JSON: {"universal": "YES" or "NO", "reason": "..."}.

Personalization — CompRel generation

You are designing a Compositional Reasoning multiple-choice question.

Required fact frames; all must be satisfied by the gold option:
⟨*required_frames*⟩

Downstream task affordance: ⟨*affordance*⟩
Required question stem: ⟨*question_stem*⟩
Banned words/phrases: ⟨*banned_forms*⟩

Invent one realistic assistant decision where the facts compose naturally. Generate four plausible options of similar length and appeal. Exactly one option must satisfy all required facts; every distractor must violate at least one required fact. The stem must remain generic, under 15 words, and must not name, categorize, or hint at any user fact.

Reply with only valid JSON containing question, options, gold_option, satisfaction, violations, and explanation.

Personalization — CompRel fact-list oracle

You are answering a multiple-choice question about a user. The user has the following facts:

⟨*facts_text*⟩

Question: ⟨*question*⟩
⟨*options_text*⟩

Reply with only one letter (A, B, C, or D). If the facts above do not uniquely determine the answer, reply with "uncertain".

Personalization — CompRel no-context option-bias audit

You are answering a multiple-choice question about an unknown user. You have no facts about this user. You must still choose the option that seems most generally appealing or likely for an average unknown user based only on the option text.

Question: ⟨*question*⟩
⟨*options_text*⟩

Reply with only one letter (A, B, C, or D).

Personalization — CompRel matrix and family audit

```
Required user facts:
⟨facts_text⟩

Question: ⟨question⟩
Options:
⟨options_text⟩

For every option A-D and every fact, mark true only if the option clearly satisfies
that fact; otherwise mark false. Also classify the primary difficulty as one of:
applying one fact, applying multiple facts, tracking temporal updates, paraphrased
recall, direct recall, or abstention. The item is valid only when applying multiple
facts is the dominant difficulty.

Reply with valid JSON containing the satisfaction matrix and the family label.
```

Personalization — DirAcq judge (`gpt-5.4-nano`)

```
You are an evaluation judge. Given a question, a reference answer, and a model's
response, classify the model's response into exactly one category.

Question: ⟨question⟩
Reference answer: ⟨expected_answer⟩
Model response: ⟨trained_answer⟩

Categories:
- MATCH: The model's response contains the key factual claim from the reference answer.
Different wording is fine, but the core fact must be present and correct. Partial
correctness does not count.
- NONMATCH: The model's response is factually wrong, vague, off-topic, or missing the
key claim from the reference answer.
- REFUSAL: The model explicitly declines to answer, says it doesn't know, lacks
information, or cannot answer the question.

Think and then reply with exactly one word: MATCH, NONMATCH, or REFUSAL.
```

F. Common Failure Modes

For each method we surface a question type it most reliably fails on, with one verbatim (question, method response, oracle response) triple. Each cell in Table 1 aggregates many calls; here we want a reader to see *what* the method got wrong, not just *that* it got it wrong. All quotes are pulled directly from the eval streams (responses truncated with `[...]` where indicated).

F.1. Base Model

The base model confidently fabricates precise specifics for events that exist nowhere in the corpus. The example below is from World News BoundAbs, where the gold expects principled abstention.

Type	BoundAbs
Question	*By what percentage did the stock price of Saudi Aramco fall on the Monday following the announcement of the IEA's strategic reserve release?*
Method response	The stock price of Saudi Aramco fell by approximately **2.5%** on the Monday following the announcement of the International Energy Agency's (IEA) strategic oil reserve release in November 2022. This drop occurred amid concerns that the coordinated release of oil from strategic reserves by several countries, including the U.S. and others, would reduce global oil prices and impact oil producers' revenues.
Oracle response	No, the corpus contains no record of the percentage change in the stock price of Saudi Aramco on the Monday following the announcement of the IEA's strategic reserve release.

F.2. Memory Scaffold

At the 32× compression budget, Memory Scaffold's responses on Code Changelogs RefRes frequently begin a draft that includes the right identifier, then enter a degenerate repetition loop that exhausts the token budget so no clean answer is ever produced.

Type	RefRes
Question	*What method in pandas does return a boolean Series indicating whether all characters in each string element belong to the ASCII character set, returning False for empty strings?*
Method response	The method in pandas `pandas` that returns a `Series` of boolean values indicating whether all characters in each string are ASCII, returning `False` for empty strings, is: `Series.str.split(expand=True)` — no, that's not right. The correct answer is: `Series.str.isascii()` returns a boolean Ser[...] [long tail of repeated '0' tokens until `max_new_tokens=4096`]
Oracle response	`isascii`

F.3. KVzip

KV-cache eviction discards updated tokens, and the model falls back to a stale figure from earlier in the prefix. In the example below, the eval's pre-update entity is "Sub-$5,000 per ounce" and post-update is "$5,100 per ounce"; KVzip reports a value far below either operating point.

Type	TempUpd
Question	*As of 2026-03-09, what is the spot price of gold per ounce?*
Method response	Based on the provided text, the spot price of gold is not explicitly stated. However, the text indicates that gold prices have been on a "long upward trend" and recently surpassed $1,000 per ounce in February 2026, and were "above $1,000" in early March 2026.
Oracle response	$5,100 per ounce

F.4. Cartridges

Cartridges retrieves the right topic-level entity but invents specifics — here, the right ousted CEO (Alex Chriss) but a fabricated successor (David Marcus, "former Facebook COO") instead of the corpus-grounded Enrique Lores.

Type	ImpRel
Question	*What major leadership changes happened at PayPal in early 2026?*
Method response	In early 2026, PayPal's CEO, Alex Chriss, stepped down, and the company appointed David Marcus, former Facebook COO, as his successor. This leadership change occurred amid broader changes at the company, including its pursuit of a potential merger.
Oracle response	PayPal's board ousting CEO Alex Chriss and appointing Chair Enrique Lores as the new president and CEO.

F.5. Chunked Summary (32K→512)

At the most aggressive compression ratio, the per-feature identifier names are lost; the model produces plausibly-shaped but non-existent identifiers. All 8 trials below produced the same fabricated method, `DataFrame.write_partitioned_parquet`, which does not exist in Polars.

Type	RefRes
Question	*How to write a DataFrame to a directory structure partitioned by a specific column while providing a custom function to determine the output file path for each partition?*
Method response	`DataFrame.write_partitioned_parquet`
Oracle response	`PartitionBy`

F.6. Vanilla RAG (top-5)

The retriever's BM25 lexical overlap pulls in unrelated M&A pieces, and the reader correctly reports that nothing relevant was retrieved. The top-5 hashes (`eb62dac78605`, `29467df55a52`, `4167d29966e4`, `232b3b0b2520`, `ff1609c4e7e1`) contain none of the Victory Capital / Janus Henderson coverage.

Type	ImpRel
Question	*What major corporate acquisition battles or bidding wars occurred in the asset management industry in late February 2026?*
Method response	There is no mention of major corporate acquisition battles or bidding wars in the asset management industry in late February 2026 in the provided reference material. The reports focus on broader M&A trends, geopolitical impacts on oil prices, and general corporate strategy, but do not detail specific transactions or conflicts within the asset management sector during that period.
Oracle response	Victory Capital's public $8.6 billion rival bid for Janus Henderson, challenging the previously agreed-upon acquisition by Trian Fund Management.

F.7. NTP (full FT, chronological)

On the *earliest*-trained material in the chronological stream, full-FT NTP collapses with token-level corruption. The example below is from SciPy 1.14.0 (released 2024-06-24, the first feature in the training corpus). The base model gets it 7/8 and the SciPy-only LoRA gets it 7/8, but the chronologically-trained full-FT run gets it 0/8 — textbook catastrophic forgetting.

Type	RefRes
Question	*What class in SciPy does represent the probability distribution of the sum of n independent and identically distributed uniform random variables on the interval* $[0, 1]$*?*
Method response	The probability distribution of the sum of n independent and identically distributed (i.i.e.d.) uniform random variables on the interval [0, víde](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats Irwin–Hall distribution is implemented in SciPy. The class that represents this [...] [other trials emit random tokens like 战火, `VMLINUX`, `i.iiversally`.]
Oracle response	`irwinhall`

F.8. SDFT vs. matched SFT (C+Q)

At the matched (C+Q) cell, SDFT's reverse-KL-to-context-conditioned-teacher signal can fail to internalize a fact that plain SFT (C+Q) at the same hyperparameters internalizes cleanly. On the question below, SDFT scores 0/8 while the matched SFT (C+Q) scores 7/8 with the response *"Yes, coordinated airstrikes hit Tehran and other cities across Iran early on Saturday morning local time."*

Type	DirAcq
Question	*Did the United States and Israel launch coordinated airstrikes against Iran on March 1, 2026?*
Method response	No, the United States and Israel did not launch coordinated airstrikes against Iran on March 1, 2026. According to the article, the U.S. and Israel have not launched coordinated strikes, and there is no mention of a joint military operation on that date. The article states that Iran has not yet been [...]
Oracle response	Yes.