WHEN DOES CONTINUAL LEARNING REQUIRE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models (LLMs) become increasingly capable, the next question is how to enable them to continually learn. Today, the field largely frames this as a problem of context management and mitigating forgetting. We argue this framing is incomplete: continual learning is fundamentally about increasing model competence as the world changes. We disentangle this change along two axes: space, where the model encounters new domains, and time, where the underlying data drifts under a fixed task. We introduce a common framework that treats continual learning methods as update operators acting on model state, allowing weight updates, prompt optimization, retrieval, and context compression to be studied together. We operationalize this framework through sequential evaluations built from existing benchmarks and public datasets, comparing supervised learning, reinforcement learning, prompt-based methods, context compression, and retrieval. Our experiments with Qwen3-8B reveal distinct requirements for adaptation across settings. Under domain shift, self-distillation achieves higher final average performance than prompt-based methods. For changing factual knowledge, retrieval that replaces outdated records outperforms the evaluated learning methods without updating model weights, but this advantage degrades when conflicting versions accumulate. Under financial temporal drift, SDPO improves economic performance over the frozen model, while accuracy alone obscures differences in useful adaptation. These findings make the distinction between storing new information and learning from it central to continual learning. More broadly, our framework connects the choice of update mechanism to the structure of how data changes. We hope that understanding where each method succeeds and fails will guide the design of stronger continual learning systems.

1 INTRODUCTION

Traditionally, continual learning has been defined as mitigating catastrophic forgetting (McCloskey & Cohen, 1989): *how do we learn task B without forgetting task A?* One approach to this problem is to augment or compose model parameters as proposed in superposition (Cheung et al., 2019), adversarial networks (Ebrahimi et al., 2020) and side networks (Zhang et al., 2020). However, this setup is not realistic in the current paradigm of large language models (LLMs), which often rely on large scale pre-, mid-, and post-training to acquire general capability and after deployment need to be adapted to a *broad* range of potential use cases. At the same time, standard LLM evaluations focus on measuring capabilities on static tasks such as general knowledge (Wang et al., 2024), domain specific abilities (Hendrycks et al., 2021; Jimenez et al., 2023; Ouyang et al., 2025; Grady et al., 2026), and challenging frontier problems (Glazer et al., 2024; Mang et al., 2025); these however fail to measure the ability to become more competent under conditions that evolve through time – which is critical for practical deployment.

Without a unified framework, the community approached continual learning as a memory problem solved with retrieval-augmented generation (Lewis et al., 2020), in-context learning (Brown et al., 2020), smart context management (Li et al., 2023; Zhang et al., 2025a) and harnesses (Zhang et al., 2025c; Ye et al., 2026; Lee et al., 2026); as a steering problem solved with prompt optimization (Novikov et al., 2025; Sharma, 2025; Agrawal et al., 2025); as a reasoning problem solved with reinforcement learning and distillation (Lu & Lab, 2025; Zhao et al., 2026; Hübotter et al., 2026; Shenfeld et al., 2026); and as a compression problem solved with architectural changes (Eyuboglu

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

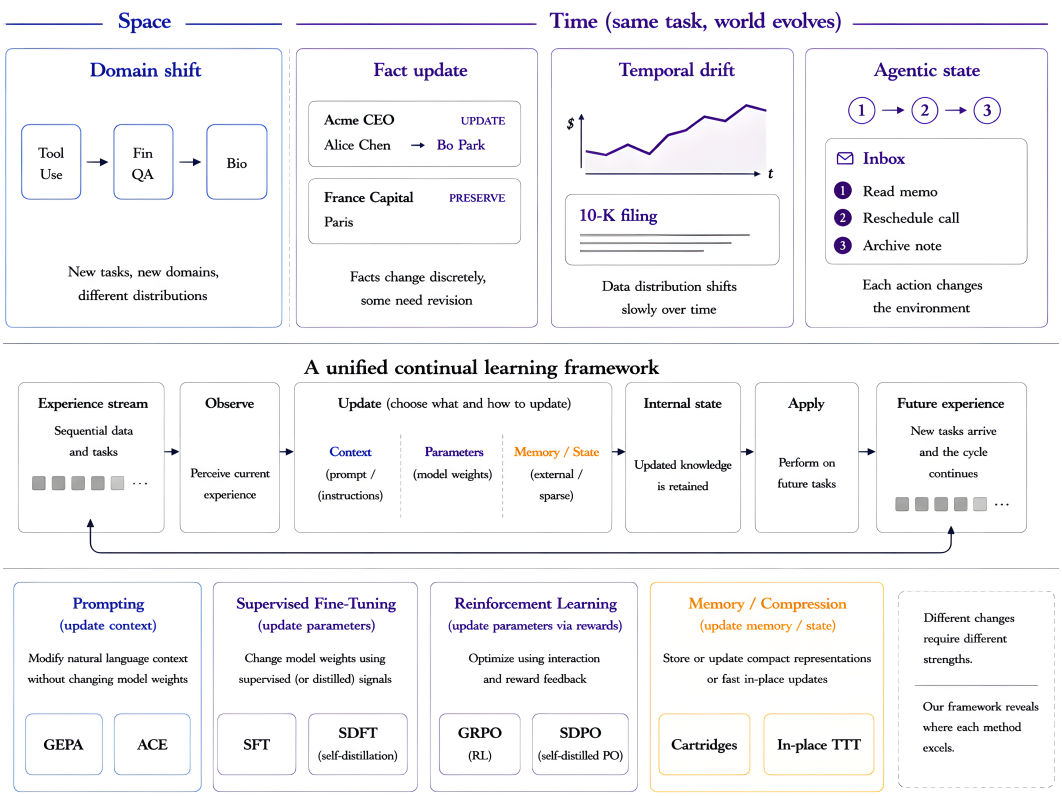


Figure 1: **A unified continual learning framework spanning axes of change and update mechanisms.** We identify four axes along which tasks evolve: domain shift, fact update, temporal drift, and agentic state. All methods are cast within a common framework where an experience stream triggers updates to one of three internal components: context, parameters, or memory/state. Existing methods are unified accordingly: prompting methods (GEPA, ACE) modify context; fine-tuning methods (SFT, SDFT) and reinforcement learning methods (GRPO, SDPO) update parameters; and memory/compression methods (Cartridges, In-place TTT) maintain external state.

et al., 2025; Lin et al., 2025; Tandon et al., 2025; Feng et al., 2026). Each of these takes a different bet on what continual learning actually requires, but the field still lacks unified insights across these methods to direct progress.

In our work, we provide a unified framework to define and measure continual learning in modern LLMs. We define continual learning as the problem of increasing competence as the world changes. We consider change along two axes: *space* and *time*, as depicted in Figure 1. Space corresponds to the classical notion of domain or task shift, where the model is asked to acquire a new skill on a different distribution without losing the previous one. Time corresponds to a different and, more common form of change in real deployed systems: the task stays the same, but the underlying world drifts. Time itself can be further split into two sub-regimes that the literature usually conflates. Slow trends (e.g. yearly financial filings) ask the model to stay aligned; the best strategy here is to keep extracting whatever structure survives across years and ignore year-specific noise. Discrete fact changes (e.g. Wikipedia revisions) ask the opposite – the right move is to actually rewrite the affected belief while leaving the rest of the model's knowledge untouched. A third sub-regime is agentic accumulation: the task family itself stays fixed (e.g., actions inside a web app), but the environment's state drifts as a direct consequence of the model's own prior actions, so that what one step leaves behind becomes part of the task the model faces at the next step. This second sub-regime is the LLM analogue of the frame problem (McCarthy & Hayes, 1981): when the world updates, a learner must decide which of its beliefs are now outdated and which can remain fixed.

To measure continual learning under our definition, we cast widely used LLM evaluations as problems that unfold sequentially under a single mechanism-agnostic protocol, so prompt, weight, and architectural updates can be compared on equal ground. On a common backbone (Qwen3-8B) (Yang et al., 2025), we evaluate eight methods across four families: prompt optimization (GEPA (Agrawal et al., 2025), ACE (Zhang et al., 2025c)), offline supervised updates (SFT, SDFT (Shenfeld et al.,

2026)), online reinforcement learning (GRPO (Shao et al., 2024), SDPO (Hübotter et al., 2026)), and context-compression (Cartridges (Eyuboglu et al., 2025), In-place TTT (Feng et al., 2026)).

Interestingly, we observe a consistent set of tradeoffs across methods: (i) Prompting-based approaches such as GEPA and ACE fit quickly and heavily to past trajectories, achieving strong backward accuracy, but encounter sharp degradation on future tasks. (ii) Distillation-based methods such as SDFT and SDPO accumulate knowledge more stably over time, yet struggle to rapidly incorporate new information or update outdated knowledge. (iii) Context compression approaches, including Cartridges and In-place TTT, improve efficiency and memory management but do not substantially improve the ability to learn new tasks. (iv) In contrast, online reinforcement learning methods such as GRPO adapt more effectively to knowledge updating, though they remain highly sensitive to noisy or unstable reward signals. (v) On the agentic axis, where the stage ordering is generated by the agent's own actions rather than imposed by us, both a prompt-based playbook (ACE) and weight-based fine-tuning (SFT) compound experience and beat their zero-shot baselines at every chain length, though absolute success still decays as the chain grows.

All together, we make the following contributions:

- We formalize continual learning in LLMs as the problem of increasing competence over time. Through our protocol, we provide a unified perspective that allows us to consider both parametric and non-parametric update methods for continual learning. We adapt these methods to the continually learning setting and align their evaluation metrics.
- We introduce a broad suite of sequential LLM tasks to evaluate continual learning. These including domain adaptation, agentic tasks, financial analysis, and temporally-dependent knowledge updates.
- We find that prompt-based methods alone are insufficient across most regimes, calling for actual learning. Different learning methods each have their own strengths depending on how the task and data shift over time.

2 Related Works

Continual Learning. Traditionally, the field has considered continual learning as preventing the problem of catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999). In this context, the problem is that sequential parameter updates can overwrite knowledge needed for previous tasks; this led to a focus on designing methods that reduce task interference (Javed & White, 2019; Golkar et al., 2019; Ebrahimi et al., 2020; Zhang et al., 2020; Cheung et al., 2019). From another angle, continual learning has been positioned as maintaining plasticity (Dohare et al., 2024), where models must not only remember old tasks, but also have the ability to learn new ones (Lyle et al., 2022; Nikishin et al., 2022).

Non-parametric updates. At deployment, we can adapt LLMs without weight changes by changing the inference-time environment around a fixed model. One family of methods changes the information available to the model, for example through retrieval-augmented generation (RAG) (Lewis et al., 2020), external memory, reusable long-context representations (Eyuboglu et al., 2025), context compression (Li et al., 2023), or recursive access to large external contexts (Zhang et al., 2025a). A closely related family changes the behavioral scaffold through which that information is used, including demonstrations (Brown et al., 2020), prompt instructions and agent contexts (Agrawal et al., 2025; Zhang et al., 2025c; Yuksekgonul et al., 2024) and multi-stage LM programs (Opsahl-Ong et al., 2024; Khattab et al., 2023; Li et al., 2026). More broadly, agentic systems can adapt by modifying their surrounding harnesses (Ye et al., 2026; Lee et al., 2026), learn component-level reward specifications (Wu et al., 2025) or improve through external programs and agent code modifications (Cemri et al., 2026; Novikov et al., 2025; Zhang et al., 2025b; Sharma, 2025).

Gradient-based (parametric) updates. Another adaption approach is to update weights or activations. At one extreme, full fine-tuning modifies all parameters of the model; while effective, this regime is often computationally expensive and unrealistic for frequent or online adaptation. Activation editing (Ilharco et al., 2022) uses task vectors applied to all parameters. More commonly, work studies *parameter-efficient* methods, such as LoRA (Hu et al., 2022) and adapter-based tuning (Lin et al., 2025; Lopez-Paz & Ranzato, 2017), which restrict updates to small subsets of parameters. An important but underexplored regime involves severely constrained updates—very few gradient steps and tightly limited compute—closer to the conditions under which continual adaptation would

Table 1: The eight methods, organized by what crosses the stage boundary. "Sequential adaptation" is the modification that lifts a single-stage method to the staged protocol of §3. Prompting refers to methods that update in natural language. Offline corresponds to supervised learning whereas online refers to reinforcement learning. Lastly compression notes methods that compress context in activation or weight space.

Family	Method	Carried	Sequential adaptation	Distill.	Weights
Prompting	GEPA (Agrawal et al., 2025)	evolved prompt	init from $k-1$ prompt	✗	—
	ACE (Zhang et al., 2025c)	markdown playbook	init from $k-1$ playbook	✗	—
Offline	SFT	full params	resume from θ_{k-1}	✗	Full
	SDFT (Shenfeld et al., 2026)	params + teacher	θ_{k-1} as teacher	✓	Full
Online	GRPO (Shao et al., 2024)	full params	resume from θ_{k-1}	✗	Full
	SDPO (Hübotter et al., 2026)	params + pref. pool	θ_{k-1} generates pairs	✓	Full
Context	Cartridges (Eyuboglu et al., 2025)	backbone + adapter	new adapter per stage	✓	Sparse
Compression	In-place TTT (Feng et al., 2026)	per-input update	reset between inputs	✗	Sparse

need to occur in deployed systems. Continual learning methods differ not only in where they store updates, but also in what learning signal drives the update. OPSD (Zhao et al., 2026; Lu & Lab, 2025), SDFT (Shenfeld et al., 2026) and SDPO (Hübotter et al., 2026) learn from self-distillation, GRPO (Shao et al., 2024) and h1 (Motwani et al., 2025) optimize end-of-task rewards, LoRD (Liu et al., 2025b) uses teacher–student distillation, sparse memory finetuning (Lin et al., 2025) localizes the next-token objective to sparsely accessed memory slots, test-time training (Feng et al., 2026) performs input-conditioned weight updates during inference.

3 METHOD

Typically, continual-learning evaluations are tightly coupled to a particular method: weight-update methods use task accuracy with a fixed fine-tuning budget; prompt-optimization often uses few-shot generalization with a fixed token budget; agentic frameworks rely on end-to-end pass rate. As a result, prompt, weight, and architectural updates are rarely compared fairly against one another. To gather broader insights about what methods are necessary for continual learning, we define a simple protocol that is compatible with any update operator. To do so, we consider all forms of updates (prompting, full weight, sparse weight/activation) as potential strategies, and group methods into families based on which aspect of the model they change during optimization (Tab. 1). Formally:

A learner sees stages $\mathcal{T}_1, \ldots, \mathcal{T}_K$ in fixed order. Each $\mathcal{T}_k$ specifies a training distribution $\mathcal{D}_k^{\text{tr}}$ and an evaluation set $\mathcal{D}_k^{\text{eval}}$. Between stages the learner applies an update operator$\theta_k = \mathcal{U}_k(\theta_{k-1}, \mathcal{D}_k^{\text{tr}}), \quad \theta_0 = \theta_{\text{base}}$. $\mathcal{U}_k$ is unrestricted: it may modify model weights, edit a system prompt, write to or read from external memory, attach an adapter, or perform no update. A method is fully specified by its family of $\mathcal{U}_k$ and a per-stage compute budget C (held constant across methods within a benchmark).

We evaluate eight methods across four families: prompt optimization, supervised weight updates, reinforcement learning, and per-stage architectural updates. All use Qwen3-8B (non-thinking) (Yang et al., 2025) as θ_0 and run under the protocol of §3. Table 1 states what each method carries across the stage boundary; the per-family paragraphs that follow specify each $\mathcal{U}_k$ and its sequential adaptation.

Prompt optimization (GEPA, ACE) Both methods leave model weights unchanged. GEPA optimizes the system prompt through a reflect-and-mutate loop scored on a validation minibatch; the best-scoring mutations propagate forward. ACE maintains a markdown playbook edited incrementally through explicit add/keep/drop operations, which the authors show resists brevity-bias and context collapse. In our protocol, $\mathcal{U}_k$ initializes optimization for stage k from the optimized state at the end of stage $k-1$ (the evolved prompt for GEPA, the playbook for ACE), prepended to the default instruction for $\mathcal{T}_k$. No replay of prior data is performed. Apparent forward transfer is therefore attributable to the carried prompt or playbook, not to revisiting training examples.

Supervised / Offline Learning (SFT, SDFT) SFT fine-tunes all parameters on $\mathcal{D}_k^{\text{tr}}$, resuming from θ_{k-1}. **SDFT** (Shenfeld et al., 2026) is self-distillation under a forward-KL loss against soft targets

from a teacher; in the original work the teacher is the model itself on demonstration-conditioned inputs. Our sequential adaptation: at stage k, the teacher is θ_{k-1}, so the student learns the new stage under a regularizer pulling it toward its own previous-stage distribution. This is the smallest modification that lifts SDFT to a continual-learning operator and is the only modification we make. SFT and SDFT share a per-stage update budget within C and differ only in loss applied at each step.

Reinforcement / Online Learning (GRPO, SDPO) GRPO (Shao et al., 2024) samples rollouts in groups, computes within-group advantages from a verifiable reward, and updates the policy under a KL constraint. For benchmarks without a verifiable reward (10-K, TempWiki) we use a binary correctness signal from the ground-truth label. **SDPO** (Hübotter et al., 2026) is sequential DPO. Our adaptation: at stage k, θ_{k-1} generates both the chosen and rejected continuation pools, so the preference signal is implicitly relative to the model's accumulated prior rather than to a fresh reference. Both methods consume C as policy-update steps.

Architectural / Context Compression (Cartridges, In-place TTT) These methods change the model's update interface rather than the loss applied to its parameters. **Cartridges** (Eyuboglu et al., 2025) freezes the backbone and learns a small per-stage component, distilled from a teacher trace and stitched in at inference. The backbone is θ_0 for every stage; each stage's update is structurally local. **In-place TTT** (Feng et al., 2026) performs an input-conditioned weight update at inference using a self-supervised auxiliary loss, reset between inputs. We include In-place TTT as a locality reference: its update is per-input rather than per-stage, which lets us isolate the locality axis from the accumulation axis in the experiment.

4 Experiments

For each task evaluation, we apply our general continual learning protocol and vary only how the world changes between optimization stages. We work with the two axes of change: **space**, where the model moves across substantively different domains, and **time**, where the model stays in one domain but the underlying data drifts across stages. Within these axes we instantiate four settings: *independent domains* (space), where each stage is a different task; *selective fact change* (time), where some labels change between stages while others remain stable; *continuous temporal drift* (time), where the task is fixed but the data slides smoothly across stages; and and *agentic accumulation* (time), where the task family is fixed but the environment's state drifts as a direct consequence of the agent's own prior actions rather than an external clock or editor. More details on task design choices can be found in Appendix A.2.

4.1 Domain Transfer

When a model is exposed to a new domain, it will need to acquire new skills or capabilities without catastrophically forgetting. This is our evaluation setting and the classic test of continual learning and covers the axis of space.

Setup. Three unrelated supervised tasks, ToolUse (Tang et al., 2023), FinQA (Chen et al., 2021), and SciKE-Bio (Feng et al., 2024) (ToolUse and SciKE-Bio subsets from Shenfeld et al. (2026)), are presented in a fixed order. Each stage trains Qwen3-8B on 500 examples of one task and evaluates on all three at every step; the best checkpoint on the current task is carried into the next stage. Because the tasks share no surface structure, a method can fail either by not acquiring the new task or by overwriting earlier output formats. Details and hyperparameters are in Appendix A.2.1.

Prompt and harness optimization surfaces a speed-vs-stability trade-off. We observe that GEPA fits very quickly to some training stages such as FinQA, but has the most severe catastrophic forgetting when moving on to the next stage. In Fig. 2 we observe that despite an almost 40% increase in performance during the FinQA training stage, after the training on biology accuracy drops almost entire back down the baseline accuracy before any training. While ACE appears to be slight more stable than GEPA, it similarly improves and then immediately degrades at this same phase transition.

Distillation-based methods manage to accumulate knowledge. We observe that SDFT and SDPO both finish the sequential training with FinQA and Bio above their zero-shot baselines while keeping ToolUse near 55–60%; SDFT is the only method in our table whose three final-stage scores are all at or above their respective zero-shot levels.

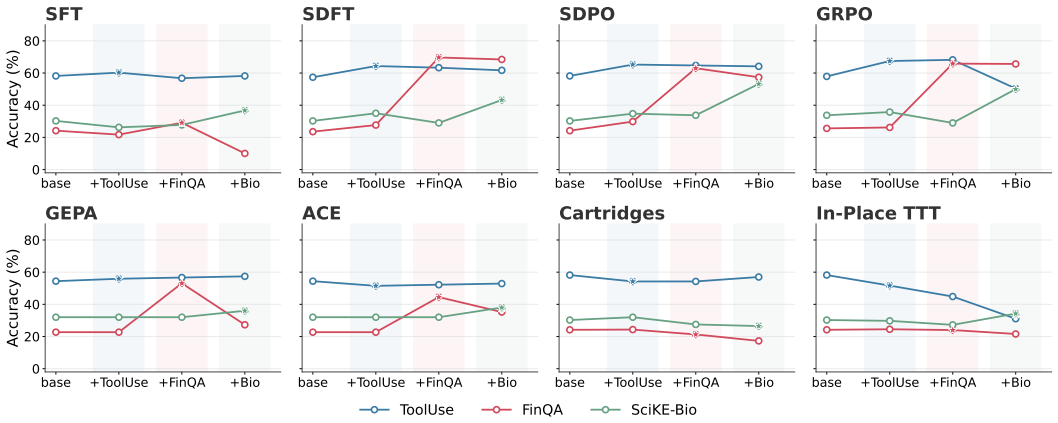


Figure 2: Per-method accuracy across the three-stage domain chain (ToolUse → FinQA → SciKE-Bio) for Qwen3-8B (non-thinking). Each panel shows one method; the three colored curves are held-out accuracy on each of the three eval sets across all training steps, with the corresponding panels marking stage boundaries. Diagonal acquisition, the curve corresponding to the currently-trained task, and the other two curves showing off-diagonal forgetting, are visible side by side.

Context compression does not help learn new tasks. Cartridges and In-place TTT are less sensitive to than prompt methods to domain shift, but at the same time they do not show steady improvement across all tasks over training phases like SDFT. Performance is largely flat or even simply degrades per task at each progressive training phase. These methods never perform a full weight update, and thus appear to struggle to accumulate new capabilities under the domain shift task.

4.2 Catastrophic Memorizing: Update Truly New Knowledge

In the previous experiment, we tested methods on their ability to stay aligned and accumulate knowledge across time. Here, we ask the opposite: when presented with discrete fact changes, can a model rewrite the affected belief while leaving the rest of its knowledge untouched? Avoiding catastrophic forgetting is necessary, but so is knowing when and how to update. The failure to do so is what we call 'catastrophic memorizing'.

Data and setup. We build the chain from four monthly Wikipedia snapshots after the pretraining cutoff (2025-11-20 to 2026-02-01). Each adjacent pair yields a *drift set* of (subject, relation, object) triples whose object changed, which the model must learn; a held-out *stable set* of facts that never change across the window serves as a forgetting probe. Weight-update and RL methods (SFT, SDFT, SDPO, GRPO) train on 500 drift triples per slice as Q/A pairs, while Cartridges and In-place TTT consume the corresponding article bodies (≈528 per slice). Construction details and hyperparameters are in Appendix A.2.2; a filtered, continuously-scored variant (TempWiki-Easy) is analyzed in Appendix A.2.3.

Stability-anchored methods fail to learn how to change. Interestingly, SDFT and SDPO, which perform well on accumulating knowledge in noisy signal, struggle to learn changes in this setting. SDFT's stable-fact F1 falls from ≈0.30 at s_1 to ≈0.15 at s_3: by trying to interpolate the new slice against its own previous beliefs, the model ends up corrupting facts that were never supposed to move. SDPO shows the same effect at smaller magnitude. In contrast, GRPO shows strong performance in learning how to change while preserving performance averaging above baseline across the chain $+1.6$ points. GRPO witnesses the sharpest jump at s_2 of $+6.2$ points which is the largest measured gain of any weight update method.

Prompt evolution shows strong optimization ability on adapting to new knowledge at the cost of forgetting stable knowledge. GEPA reaches the highest drift F1 in the comparison (0.18 on s_1, 0.24 on s_3) by writing the new fact directly into the playbook in a single optimization step. The optimizer is rewarded on the validation set, which contains drift facts; it is not rewarded on stable facts, which it never sees. So GEPA edits the playbook freely, and the same edits overwrite stable entries that

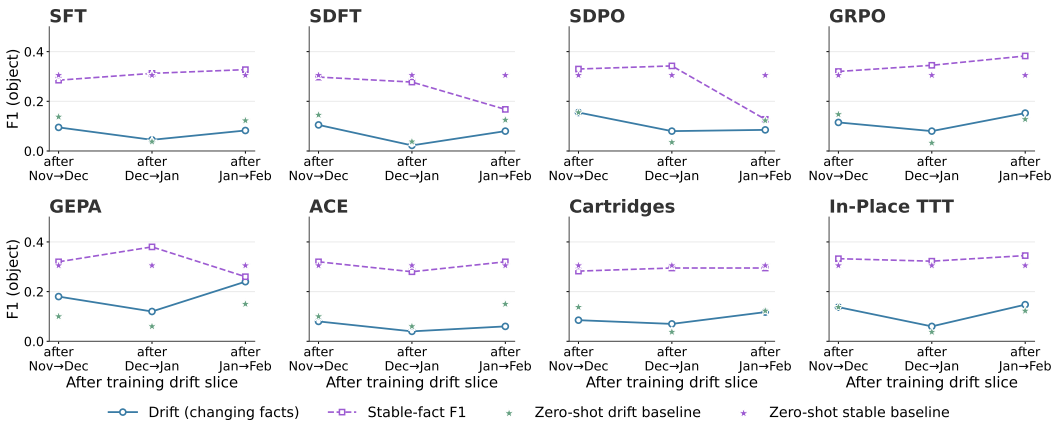


Figure 3: F1 score across the three TempWiki slices for each method (Qwen3-8B, no thinking). A method that genuinely learns the new facts should improve drift F1 while preserving stable F1; methods that conflate the two appear as diverging curves. Cartridges and In-place TTT preserve stable F1 most cleanly; SDFT and SDPO degrade stable F1 sharply; GEPA and ACE achieve the largest drift gains but pay heavily on stable. GRPO is the only weight-update method whose stable F1 trajectory does not move in the opposite direction of its drift trajectory

depend on the same playbook tokens. Stable F1 falls from 0.32 at s_1 to 0.26 at s_3. ACE shows the same profile although at a smaller magnitude.

Context compression barely influences the base model. For cartridges and In-place TTT drift F1 stays within a few points of baseline across all three slices, with no separation in either direction. Cartridges averages -0.8 on drift, with one $+3.2$ inflection on s_2 and despite performing better than SDFT and SDPO, but shows no improvement compared to baseline, even harming the performance, belonging to the same category at times.

4.3 Noisy Temporal Drift

From the previous experiments, we observe over longer time horizons, performance steadily degrades. In our following temporal drift experiments, we dig deeper into the problem by setting up a task where the model can only improve if it manages to accumulate weak and noisy signals through time.

Setup. We predict whether a stock moves up or down in the 30 days after its 10-K filing. This forces a two-sided drift: some patterns in how firms describe risk, growth, and capital plans persist, while others are regime-specific (e.g., before and after Covid) and stop working. *Doing well now does not guarantee doing well in the future.*

We use full-length SEC EDGAR filings (10–13K tokens) labeled by 30-day forward return (Loukas et al., 2021; Magner et al., 2025), train sequentially on filing years 2015–2020, and after each year evaluate on every year to measure backward and forward transfer. The per-filing signal is weak, but a small, tradeable edge accumulates in aggregate (Magner et al., 2025) (Figure 8); zero-shot Qwen3-8B sits near 50%. Details and hyperparameters are in Appendix A.2.4.

RL struggles when the reward signal is weak and noisy. GRPO shows a significant accuracy drop both on forward- and backward-looking sentiment; its success depends heavily on the reward ranking inside each group. If the reward is noisy, the algorithm may reinforce the wrong answer.

Stability-anchored methods show the strongest forward transfer with minimal past degradation. SDFT's past-year accuracy stays flat around 0.50 while its future-year accuracy climbs to ≈ 0.62 by 2020: distilling against its own previous state avoids both drift and over-fitting to the current year. Cartridges is the most stable method we tested, with past and future hovering around 0.55–0.58 throughout the chain. In-place TTT is the locality reference: its accuracy stays in a tight 34.5–51.5% band across the chain (mean 46.9%) and never separates from baseline. The reset-per-input update geometry is incompatible with absorbing a slow-moving trend, but it pays no price for trying – the method neither acquires nor degrades.

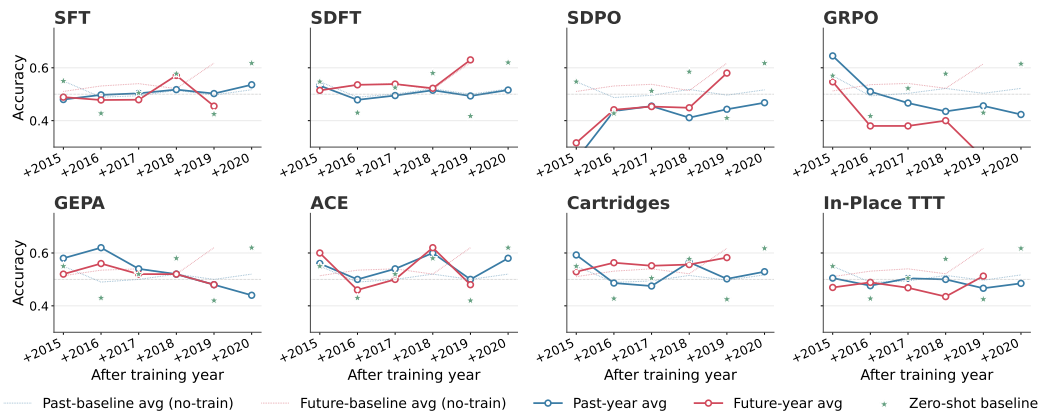


Figure 4: Backward and forward accuracy on the 10-K sentiment task, evaluated after sequential training on each year from 2015 to 2020 (Qwen3-8B, no thinking). Blue: average accuracy on past years ($\leq$ current). Red: average accuracy on future years ($>$ current). Stars: zero-shot baseline per year. Distillation-based methods (SDFT, SDPO, Cartridges) sustain forward accuracy, while prompt-based methods (GEPA, ACE) and GRPO degrade on future years. SFT stays near chance throughout.

Prompting methods fail to extract useful knowledge for forward transfer. On this task, similar to RL, GEPA also struggles with learning meaningful information from noisy signals. ACE fits better on past-year data, but also completely fails to predict future years' trend, which suggests that it does not accumulate meaningful knowledge.

Qualitatively, GEPA discovers *generic* financial-analysis heuristics (Figure 5). These edits fail to capture characteristics of an ideal adaptive analyst e.g. learn to discard irrelevant sections, normalize sentiment relative to sector or company baselines, or distinguish between newly informative elements and those that are already priced in by the market. While it is possible to construct a strategy with a positive Profit and Loss (PnL) with a rule as simple as focusing on relevant sections in the 10-K (see Figure 8), consistent with the observed drop in forward accuracy after optimization, the optimized prompt reduces even this forward PnL. We conclude that in this noisy, time-changing regime, GEPA overfits to transient textual heuristics while failing to preserve exposure to economically meaningful signals that remain predictive out-of-sample.

Ideal analyst diff

```
You are a financial analyst analysing 10k's. For each 10k, infer
whether short-term post-filing sentiment should be UP or DOWN.

+ Discard sections 1B, 4, 6, 10-15 as often sparse / inconsistent
+ and not tied to forward-looking operating performance

+ Section 1A containing Risk Factors shows value.

+ Companies in the tech sector often have more
+ optimistic language.

+ Be sure to compare current sentiment to a historical baseline
+ for the company and industry.

- Section 1A seems to be priced in fast, becoming less useful
- for short-term predictions.
```

Optimizer diff

```
You are a financial analyst analysing 10k's. For each 10k, infer
whether short-term post-filing sentiment should be UP or DOWN.

+ Look for positive indicators:
+ dividends, covenant compliance, growth, expansion.

+ Look for negative indicators:
+ cash-flow restrictions, losses, compliance issues, uncertainty, risk.

+ Pay particular attention to:
+ financial metrics, legal proceedings, MD&A, governance.

+ Be cautious of optimistic language unsupported by performance.
+ Consider historical returns after similar filings.

- update your internal heuristics over time
+ use feedback to refine heuristics after each prediction
```

Figure 5: Prompt evolution on the 10-K sentiment task: (L) how an idealised analyst may evolve its instructions over time, both noting persistent trends and updating outdated information; (R) the evolution of GEPA shows the addition of generic, static-in-time financial heuristics.

4.4 Agentic Task

The previous two experiments test the time axis under drift that comes from outside the model. Here we test a third sub-regime of time introduced in §3: the task family stays fixed (actions inside a web

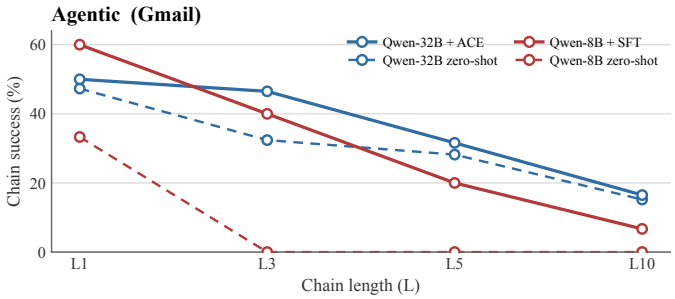


Figure 6: End-to-end chain success rate across chain lengths L1–L10 on the Gmail agentic suite. Color marks the model (blue Qwen-32B, red Qwen-8B); solid lines are the update mechanisms (an ACE playbook on Qwen-32B, and SFT on Qwen-8B distilled from teacher traces) and dashed lines are the corresponding zero-shot baselines. Both mechanisms improve over their baseline at every length, most on the three-step (L3) chains.

app), but the environment's state drifts only because the agent itself acted on it, so state left behind by one step becomes part of the task the model faces at the next.

Data. We build on WebArena-Infinity (Zhou, 2026), generating chains from each app's feature description and state files so that step $i{+}1$ depends on state left behind by step i (e.g., in Gmail: create a label, rename it, then apply it to an email). Each step has a programmatic verifier that checks the app's state directly.

Setup. We ask whether experience *compounds* across a chain, adding learning to a `browser-use` agent in two ways: an ACE playbook (Zhang et al., 2025c) on Qwen-32B that accumulates procedural guidance from prior rollouts, and supervised fine-tuning of Qwen-8B on successful traces from a stronger teacher agent. A single agent session persists across all steps of a chain, so the model sees its own prior trajectory. The harness, teachers, and hyperparameters are in Appendix A.2.5.

Results. Both update mechanisms raise end-to-end chain success over their no-learning baseline at every horizon length (Figure 6). The ACE playbook lifts Qwen-32B over its zero-shot baseline at every length, most on the three-step (L3) chains (32.4% to 46.5%). Without training, Qwen-8B collapses to zero past L1; supervised fine-tuning on the teacher's traces recovers substantial success (40.0% at L3, 20.0% at L5, and 6.7% even at L10), and overtakes both Qwen-32B lines at L1 (60.0%). Absolute success still decays with chain length for every configuration, as accumulated trajectory state grows and later steps must be grounded in increasingly distant environmental cues, converging near the floor by L10. This is the closest setting to deployed use, since the chain comes from the agent's own actions rather than an ordering we picked, and future work could go deeper on this realistic agentic setting.

5 Discussion

Conclusion. We introduced a unified framework that evaluates continual learning in large language models as increasing competence as the world changes across space (domain shift) and time (temporal drift). Recasting widely used benchmarks as sequential problems and evaluating eight methods under one mechanism-agnostic protocol, we find that no single method handles all regimes: prompt-based methods fit quickly but overwrite prior capabilities; distillation-based methods accumulate knowledge stably but resist new updates; context compression improves efficiency without improving task acquisition; and online reinforcement learning adapts best to factual change but degrades under noisy rewards. Continual learning is thus not a single capability: different patterns of change require different update behaviors, and we hope mapping where each method succeeds and fails will guide the design of more capable continual learning systems.

Limitations and Challenges. Our experiments center on a single model family (Qwen3), and the relative behavior of methods may change for larger models or in reasoning mode. Our suite also covers only four kinds of change (domain shift, factual updates, noisy temporal drift, and agentic sequences), while deployed LLMs face far more open-ended environments. The results should therefore be read as evidence about which update behaviors different regimes require, not as a definitive ranking of continual-learning algorithms.

ETHICS STATEMENT

This work develops evaluation infrastructure for continual learning in large language models, with the goal of helping the research community build systems that remain competent as the world changes. Better continual learning methods could have positive societal impact by reducing the need to retrain large models from scratch as new information becomes available, lowering the associated computational cost and energy consumption. At the same time, systems that can efficiently update their knowledge after deployment also carry risks: a model that rapidly incorporates new information could more easily be steered toward harmful or false beliefs through targeted data injection. Our work does not propose a new training method, but to the extent that it accelerates progress in continual learning, these dual-use considerations apply. We do not anticipate direct harms from the framework or evaluation protocols introduced here, as they are designed for measuring and understanding model behavior rather than enabling any specific application.

REPRODUCIBILITY STATEMENT

Compute resources and hyperparameters are listed in Appendix A, and the experimental setup, data, and evaluation details for each benchmark are described in Appendix A.2.1 (domain shift), Appendix A.2.2 and Appendix A.2.3 (factual updates), Appendix A.2.4 (temporal drift on SEC filings), and Appendix A.2.5 (agentic sequences).

AI USE STATEMENT

Large language models are the object of study in this work: the models we train and evaluate (Qwen3-8B, DeepSeek-v3.2, GPT-OSS-120B, Gemini-2.5-Flash-Lite, and the teacher agents described in Appendix A.2.5) are part of the experimental setup rather than aids to authorship. Beyond this, we used generative AI tools in three ways. First, in benchmark construction: an LLM judge filters candidate fact edits for TempWiki-Easy (Appendix A.2.3), and LLMs assisted in generating the agentic task chains and the TempWiki-Easy question–answer pairs. Every agentic step is paired with a programmatic verifier that checks the resulting application state directly, and the authors manually inspected samples of judged edits and generated items. Second, we used AI coding assistants to help write and debug experiment, evaluation, and plotting code; all such code was reviewed and tested by the authors. Third, we used AI assistants to edit the manuscript for clarity and grammar. The research questions, experimental design, analyses, and conclusions are the authors' own. We have reviewed all AI-assisted work and take full responsibility for the final content of this paper, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, et al. Adaevolve: Adaptive llm driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021.

Brian Cheung, Alexander Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno Olshausen. Superposition of many models into one. *Advances in neural information processing systems*, 32, 2019.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *European conference on computer vision*, pp. 386–402. Springer, 2020.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Di He, Wenhao Huang, and Tianle Cai. In-place test-time training. *arXiv preprint arXiv:2604.06169*, 2026.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*, 2024.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

Thomas Grady, Kip Parker, Iliyan Zarov, Henry Course, Chengxi Taylor, and Ross Taylor. Kellybench: A benchmark for long-horizon sequential decision making. *arXiv preprint arXiv:2604.27865*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6237–6250, 2022.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The twelfth international conference on learning representations*, 2023.

Aman Khan. Financial reports sec. URL https://huggingface.co/datasets/JanosAudran/financial-reports-sec.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, et al. Dspy: compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2023.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Hanchen Li, Runyuan He, Qizheng Zhang, Changxiu Ji, Qiuyang Mang, Xiaokun Chen, Lakshya A Agrawal, Wei-Liang Liao, Eric Yang, Alvin Cheung, et al. Combee: Scaling prompt learning for self-improving language model agents. *arXiv preprint arXiv:2604.04247*, 2026.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 6342–6353, 2023.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*, 2025.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Ruiqi Liu, Boyu Diao, Libo Huang, Zijia An, Hangda Liu, Zhulin An, and Yongjun Xu. Low-redundancy distillation for continual learning. *Pattern Recognition*, 167:111712, 2025b.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Lefteris Loukas, Manos Fergadiotis, Ion Androutsopoulos, and Prodromos Malakasiotis. Edgar-corpus, September 2021. URL https://doi.org/10.5281/zenodo.5528490.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

Nicolás Magner, Pablo A Henríquez, and Aliro Sanhueza. Decoding risk sentiment in 10-k filings: Predictability for us stock indices. *Finance Research Letters*, 81:107472, 2025.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, et al. Frontiercs: Evolving challenges for evolving intelligence. *arXiv preprint arXiv:2512.15699*, 2025.

John McCarthy and Patrick J Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, pp. 431–450. Elsevier, 1981.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Sumeet Ramesh Motwani, Alesia Ivanova, Ziyang Cai, Philip Torr, Riashat Islam, Shital Shah, Christian Schroeder de Witt, and Charles London. h1: Bootstrapping llms to reason over longer horizons via reinforcement learning. *arXiv preprint arXiv:2510.07312*, 2025.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. Kernelbench: Can llms write efficient gpu kernels? *arXiv preprint arXiv:2502.10517*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL `https://github.com/algorithmicsuperintelligence/openevolve`.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenic Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, et al. Optimas: Optimizing compound ai systems with globally aligned local rewards. *arXiv preprint arXiv:2507.03041*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. *arXiv preprint arXiv:2601.21557*, 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: a baseline for network adaptation via additive side networks. In *European conference on computer vision*, pp. 698–714. Springer, 2020.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025c.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Shuyan Zhou. Webarena-infinity: Generating browser environments with verifiable tasks at scale. *shuyanzhou.com*, March 2026. URL `https://webarena.dev/webarena-infinity/`.

A Additional Experimental Details

A.1 Compute and Hyperparameters

Within each benchmark, every weight-update method (SFT, SDFT, SDPO, GRPO) uses the same global batch (32), one epoch over 500 examples per phase (≈16 optimizer steps), and full-weight updates. The outer learning rate is set per-method-per-benchmark to keep cumulative parameter displacement under the empirical collapse threshold for that label regime (Dohare et al., 2024). Each training phase took no longer than 1 hour on a node of 8 H100s.

Method	Domain (B)	Finance (F)	TempWiki (T)
SFT	1×10^{-6}	1×10^{-6}	1×10^{-6}
SDFT	1×10^{-5}	1×10^{-5}	3×10^{-6}
SDPO	1×10^{-5}, warmup 10	1×10^{-5}, warmup 10	3×10^{-6}, warmup 10
GRPO	1×10^{-5}, warmup 10	3×10^{-6}, warmup 10	3×10^{-6}, warmup 10

Table 2: Outer learning rates for verl-trained methods across benchmarks. SFT trains on bare-answer gold (∼5–30 tokens), so per-token gradient signal is roughly $10\times$ the SDFT/SDPO signal, which trains on ∼200+ token rollout golds; lr=1×10^{-5} over-displaces the SFT chain in roughly one phase and the model collapses. The single-token-gold regimes (TempWiki everywhere; finance for GRPO specifically) collapse to majority-class at 1×10^{-5}, so we drop those to 3×10^{-6}.

Cartridges and In-place TTT use their own optimization recipes; hyperparameters are reported per benchmark in the sections below.

For prompting-based methods, we used cloud API models via OpenRouter. The total cost of all experiments (including development and hyperparamter tuning) was $1000 USD.

A.2 Task Settings

A.2.1 Domain shift

The learner specialises, in sequence, across three substantively different domains: tool use (ToolUse (Tang et al., 2023)), finance (FinQA (Chen et al., 2021)), and biology (SciKE-Bio (Feng et al., 2024)). Each stage provides 500 training instances and a 50-row held-out evaluation set per task; stages are presented in the canonical order `[ToolUse, FinQA, Bio]` so that per-method results are directly comparable, with permutation-averaged variants reported in the appendix to control for order effects.

Tasks and data. Each phase trains on a 500-example subsample of one task's training set and validates against all three tasks at every step. The FinQA test set is unusually large (1,125 prompts); we deterministically subsample it to 128 prompts using the SDPO baseline's seed (`np.random.RandomState(1042).permutation(1125)[:128]`, sorted), so every method evaluates on identical FinQA prompts.

Table 3: Domain shift benchmark per-phase data. The three tasks deliberately share no surface-level structure, so a method that nominally improves task acquisition can still fail by drifting the model's output format or by overwriting earlier formats once it learns a later one.

Phase	Task	Train	Val	Output format	Gold example
1	Tool Use	500	50	`Action: <tool>...`	`[{"Action": "...", "Action_Input": "{}"}]`
2	FinQA	500	128	`\boxed{<n>}`	`\boxed{94.0}`
3	SciKE-Bio	500	68	`<answer>X</answer>`	`<answer>B</answer>`

Per-method hyperparameters. SFT, SDFT, SDPO, and GRPO share the optimization budget in Table 2. SDFT, SDPO, and GRPO additionally use 8 on-policy rollouts per training prompt (temperature 0.6, top-p 0.95). SDPO optimizes a DPO-style pairwise objective over scored rollouts

(chosen = high-score, rejected = low-score) against a frozen reference model. GRPO uses PPO-style clipped objective with per-prompt z-scored advantages over the 8-rollout group.

Cartridges configuration. Each phase synthesizes a self-study corpus from the cumulative train pool of phases 1..i: Qwen3-8B (running locally as a synthesis server) is prompted with the four standard seed prompts (`use_case`, `question`, `summarization`, `structuring`) to emit 8,192 conversation-style Q/A pairs grounded in the corpus. Those 8,192 conversations are then context-distilled into a fresh trainable KV cache. Base weights are frozen. At evaluation the cartridge KV is loaded as a fixed prefix and the validation prompt is generated against it.

Table 4: Cartridges configuration on the domain shift benchmark.

Param	Value
Synthesizer	Qwen3-8B (local server)
Synthesis seed prompts	`use_case`, `question`, `summarization`, `structuring`
Synthesized Q/A pairs	8,192 / phase
Trainable KV length	2,048 tokens
KV initialization	`KVFromRandomText` (no carry-over between phases)
Distillation epochs	1
Global batch	32
Learning rate	2×10^{-2}
Trainable params	KV cache only

In-place TTT configuration. Continual pretraining via VeOmni plaintext-iterable JSONL with one record per training prompt (cumulative pool of phases 1..i in order B). Fast-weight updates are activated on a sparse layer set (paper-recommended for continual training). The chain hands each phase's HF-format checkpoint forward as the next phase's `model_path`.

Table 5: In-place TTT configuration on the domain shift benchmark. The `kl` target is for from-scratch context lengthening and is not used here.

Param	Value
Outer-loop steps / phase	50
Global batch	64
Micro-batch	1
Sequence length	16,384
Outer optimizer	AdamW
Outer-loop LR	5×10^{-6}, cosine decay (warmup 0.02, decay ratio 0.90)
Weight decay	0.1
Max grad norm	1.0
Attention backend	`flash_attention_2`
`ttt_layers`	`[0, 6, 12, 18, 24, 30, 36]`
`ttt_mode` / `ttt_proj`	`True` / `True`
`ttt_lr` (inner-loop fast-weight LR)	3
`ttt_chunk`	4096
`ttt_target`	`hidden_states` (continual-training mode)
Sharding	FSDP2, mixed precision, gradient checkpointing
Source corpus	Cumulative training prompts of phases 1..i in order B

Scoring. Each task has its own scorer. A prediction in the wrong format scores zero — the benchmark measures task knowledge *and* format-fidelity together.

- **Tool-use:** regex-extract every `Action: <name>` and `Action Input: {...}` line; parse each `Action_Input` body as JSON; accept iff the multiset of predicted action names equals the gold and the merged `Action_Input` dict equals the gold dict key-by-key.

- **FinQA:** extract the contents of the last `\boxed{...}`; accept iff $|\text{pred} - \text{gold}| \leq \max(0.005 \cdot |\text{gold}|, 0.01)$ (FinQA paper convention; exact-match would penalize harmless rounding).
- **SciKE-Bio:** look for a strict `<answer>X</answer>` tag; accept iff the single-letter `X` equals the gold letter.

A.2.2 TEMPWIKI BENCHMARK DETAILS

Task and data. Wikipedia is mirrored at four monthly snapshots — 2025-11-20, 2025-12-01, 2026-01-01, 2026-02-01 — and for each adjacent pair we extract the drift set of (subject, relation, object) triples whose object value changed across snapshots. We also extract a stable set of (subject, relation) pairs whose answer did not change across the chain, and we hold this set out across all phases as a forgetting probe.

Using these snapshots, we construct a chain with three phases corresponding to three diff-pair slices, with 500 training drift triples, 50 validation drift triples, and 50 stable validation triples. Existing temporal-knowledge benchmarks such as the original TemporalWiki (Jang et al., 2022) stop at facts that have changed and do not audit unchanged knowledge, which makes the catastrophic-memorization failure mode invisible. Pairing the drift set with a held-out stable probe is what lets the matrix expose the trade-off methods make between acquiring the new fact and preserving the surrounding knowledge.

Table 6: TempWiki per-phase data. The validation prompt is intentionally terse: a system message instructing the model to emit only the object string, a user message of the form `<subject> <relation>`, and a gold object string. Two illustrative examples: (drift) user "*São Paulo twinned administrative body*" → gold "*Buenos Aires*"; (stable) user "*Jibra'il Dallal place of death*" → gold "*Aleppo*".

Phase	Slice (snapshots)	Train drift	Val drift	Stable val (held constant)
1 (ts1)	Nov → Dec 2025	500	50	50
2 (ts2)	Dec 2025 → Jan 2026	500	50	50
3 (ts3)	Jan → Feb 2026	500	50	50

Per-method hyperparameters. SFT/SDFT/SDPO/GRPO use the recipe from the 500 drift triples Q/A-formatted directly; stable triples are held out and never appear in training. Cartridges uses the same configuration as on the domain benchmark, with two adjustments: the source corpus is the cumulative Wikipedia article bodies rather than the cumulative training pool, and the slice-1 article pool deliberately includes the stable-val subjects so the synthesizer has reason to write Q/A about them; only drift articles are added on later slices, which keeps the stable probe legitimately held out from the gradient-equivalent updates. In-place TTT uses the same fast-weight recipe as on the 10-K benchmark; per-article records are formatted as `=== Wikipedia article title=<sitelink> as_of=<snapshot> ===\n<body>\n` followed by an optional `Known facts:` block holding the canonical `subject relation -> gold` lines.

Scoring (word-level F1, paraphrase-permissive). The scorer computes word-level F1. The prediction is stripped of `<think>` traces and chat suffixes; the model's answer is taken to be the contents of an explicit `Answer: ...` line if present, otherwise the first non-empty line. Normalization (applied identically to prediction and gold): lower-case, strip the articles `a`, `an`, `the`, strip all ASCII punctuation, collapse whitespace, tokenize by whitespace into a token bag.

We score predictions with a token-level F1 against the gold answer. Both strings are normalized (lowercased, articles and punctuation stripped, whitespace collapsed) and split into tokens. Let S be the number of tokens shared between the prediction and the gold (counted with repetition). Then

$$P = \frac{S}{N_{\text{pred}}}, \qquad R = \frac{S}{N_{\text{gold}}}, \qquad F_1 = \frac{2PR}{P+R}.$$

A prediction counts as a hit if $F_1 \geq 0.5$, and zero otherwise. We report SCORE_MEAN@8: the average hit rate across 8 rollouts per prompt and across all prompts in a slice.

Table 7: Cartridges configuration on the TempWiki benchmark.

Param	Value
Synthesizer	Qwen3-8B (local server)
Synthesis seed prompts	`use_case`, `question`, `summarization`, `structuring`
`min_tokens_per_chunk` / `max_tokens_per_chunk`	1,024 / 2,048
Synthesized Q/A pairs	8,192 / phase
Trainable KV length	2,048 tokens
KV initialization	`KVFromRandomText` (no carry-over between phases)
`top_k_logits` (distillation)	20
`packed_seq_length` / `packing_mode`	2,048 / `truncate`
Distillation epochs	1
Global batch	32
Learning rate	2×10^{-2}
Source corpus	Cumulative Wikipedia article bodies (slices $1..i$)
Articles / slice	$\sim$528 (cumulative $\approx$1,484 after phase 3)
Trainable params	KV cache only

Table 8: In-place TTT configuration on the TempWiki benchmark.

Param	Value
Outer-loop steps / phase	50
Global batch	64
Micro-batch	1
Sequence length	16,384
Outer optimizer	AdamW
Outer-loop LR	5×10^{-6}, cosine decay (warmup 0.02, decay ratio 0.90)
Weight decay	0.1
Max grad norm	1.0
Attention backend	`flash_attention_2`
`ttt_layers`	`[0, 6, 12, 18, 24, 30, 36]`
`ttt_mode` / `ttt_proj`	`True` / `True`
`ttt_lr` (inner-loop fast-weight LR)	3
`ttt_chunk`	4096
`ttt_target`	`hidden_states` (continual-training mode)
Sharding	FSDP2, mixed precision, gradient checkpointing
Source corpus	Cumulative Wikipedia article bodies (slices $1..i$)
Per-article header	`=== Wikipedia article title=<sitelink> as_of=<snapshot> ===`

A.2.3 TempWiki-Easy: a filtered, continuously-scored variant

The TempWiki drift set in the main text (Fig. 3) pools *every* relation type whose object value changed between two snapshots. Two properties of that construction make the catastrophic-memorizing signal noisier than it should be. First, many Wikipedia diffs are not genuine world-knowledge updates: reformatting, disambiguation edits, list re-orderings, and annotation-style relations change the object string without there being a well-posed "new fact" to learn. Second, the object-F1 scorer used there is effectively binary at the string level, so a near-correct object (e.g. a partial or alternately formatted name) is scored identically to a wholly wrong one. Both effects inflate the apparent divergence between drift and stable curves. TempWiki-Easy is a cleaned re-cut of the same benchmark designed to isolate the update behavior from these artifacts.

Construction. We keep the four monthly snapshots and the identical three-slice chain as TempWiki (Nov→Dec 2025, Dec 2025→Jan 2026, Jan→Feb 2026), and apply three changes to the diff extraction:

- **Category allow-list.** Only (subject, relation) pairs drawn from an allow-list of relation types for which "change" is well-defined (single-valued, factual, verifiable object) are eligible. Relations prone to formatting or annotation churn are excluded outright, which removes the pathological categories responsible for spurious drift/stable divergence in the original cut.
- **Aggressive judged filtering.** Each surviving candidate diff is passed through an LLM judge that keeps the triple only if the edit reflects a semantically meaningful factual update between the two snapshots, discarding cosmetic or ambiguous edits.
- **QA regeneration + continuous scoring.** We regenerate question/answer pairs over the filtered facts and, rather than thresholding the word-level F1 into a hit at $F_1 \geq 0.5$ (the SCORE_MEAN@8 hit rate used in Appendix A.2.2), report the continuous word-level F_1 directly. Partially correct objects receive partial credit, so small real movements in knowledge are visible rather than being quantized to $0/1$.

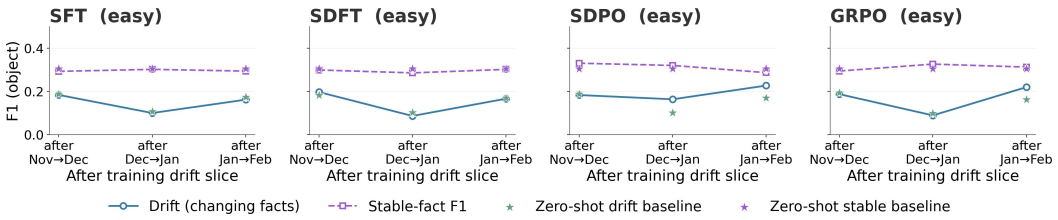


Figure 7: TempWiki-Easy: object F1 across the three drift slices (Qwen3-8B, no thinking, seed 42).

The most visible difference from TempWiki Fig. 3 is that the stable-fact trajectory (purple dashed) no longer trades off against drift. In the original cut, SDFT's stable F1 fell from to and SDPO's collapsed to by the final slice; in TempWiki-Easy both remain flat and essentially coincident with the zero-shot stable baseline across all three slices. With the confound removed, most methods sit on top of the zero-shot drift baseline: SFT and SDFT neither learn the changed facts nor harm the stable ones. The two exceptions are the reinforcement/distillation updaters. SDPO climbs to 0.22 on the final slice, above its 0.17 zero-shot baseline, while its stable F1 drifts only mildly GRPO shows the largest net effect: after the same mid-chain dip it reaches ∼0.22 drift on Jan→Feb while holding stable F1 flat.

A.2.4 TEMPORAL DRIFT

10-K Sentiment Benchmark Details

The core task is fixed (predict the directional sentiment of a 10-K filing) (Loukas et al., 2021; Magner et al., 2025) but slides across calendar time: each stage corresponds to one fiscal year of 10-K filings, presented in chronological order from 2015 to 2020. After training on year y we evaluate on every other year, splitting the score into *past-year* accuracy ($\leq y$) and *future-year* accuracy ($> y$). The target is intentionally weak-signal: real financial text is informative only in aggregate, and the question is precisely whether a learner can stay extract rules to align with a slow-moving, noisy trend rather than rely on spurious local-in-time patterns.

A second temporal benchmark constructed from Wikipedia revisions captured between November 2025 and February 2026 (Jang et al., 2022). Stages correspond to three chronological slices (s_1, s_2, s_3) of facts whose object value changes over the window. We evaluate object-level F1 on each slice after every stage, together with F1 on a control set of *stable* (unchanged) facts. This exposes a failure mode invisible in standard CL benchmarks: methods that successfully update drifted facts may simultaneously degrade unchanged ones —the mirror image of forgetting.

Task and data.

Specifically, we construct a six-year chain of 10-K filings drawn from SEC EDGAR (Loukas et al., 2021; Khan), covering fiscal years 2015 through 2020 in chronological order. Each phase contains ∼500 training filings and 50 held-out validation filings drawn from that year, with each filing represented in full (typically 10–13K tokens) wrapped between custom `[START OF FILING]` / `[END OF FILING]` markers. Cumulative training tokens grow from ∼6M after phase 1 to ∼37M after phase 6. The label is a single token, `up` or `down`, derived from the issuer's 30-day forward simple stock return joined to the filing by Central Index Key (CIK) and filing date. Prior work on financial

NLP either treats each year as an independent task or pools all years; we hold the task fixed and let the underlying data-generating process drift, exposing methods to the actual condition under which deployed financial models age out of distribution.

In this task,the model predicts the directional sign of an issuer's 30-day forward simple stock return from the textual content of its annual 10-K filing. The chain has six sequential phases for filing years 2015 through 2020 in chronological order, producing a 6×6 matrix. Each filing is full-length SEC EDGAR text wrapped between custom `[START OF FILING]` / `[END OF FILING]` markers (typically 10–13K tokens); the gold label is a single token, `up` or `down`, derived from a precomputed returns table joined by issuer CIK and filing date.

Per-method hyperparameters. SFT/SDFT/SDPO/GRPO use the optimization recipe from Table A.1 with two finance-specific data settings: `data.max_prompt_length` = 16,384 to fit the 10-K body, and `data.max_response_length` = 256 since the response is a single token plus chat tail.

Cartridges configuration. Per phase, concatenate the cumulative pool of all filings seen so far (years $1..i$, shuffled at filing granularity); synthesize 8,192 self-study Q/A pairs grounded in that pool; distill into a fresh 2,048-token KV cache. The cartridge is loaded as a prefix at evaluation, with the full filing body still included in the user message.

In-place TTT configuration. Continual pretraining on the same cumulative corpus, written as VeOmni plaintext-iterable JSONL with one record per filing (full body). Fast-weight updates are activated on a sparse layer set (paper-recommended for continual training). The chain hands each phase's converted HF checkpoint forward as the initial weights for the next phase.

Table 9: In-place TTT configuration on the 10-K benchmark. The `kl` target is for from-scratch context lengthening and is not used here.

Param	Value
Outer-loop steps / phase	50
Global batch	64
Micro-batch	1
Sequence length	16,384
Outer-loop LR	5×10^{-6}, cosine decay (warmup 0.02, decay ratio 0.90)
Weight decay	0.1
Max grad norm	1.0
`ttt_layers`	`[0, 6, 12, 18, 24, 30, 36]`
`ttt_lr` (inner-loop fast-weight LR)	3
`ttt_chunk`	4096
`ttt_target`	`hidden_states` (continual-training mode)
Sharding	FSDP2, mixed precision, gradient checkpointing
Source corpus	Cumulative filings, years $1..i$

Scoring. The prediction string is stripped of any `<think>` trace and chat suffixes, then matched against the case-insensitive whole-word patterns `\bup\b` and `\bdown\b`. The first match wins; ties break in favor of `up`. A prediction is correct iff the matched token equals the gold.

Further Details on the 10K sentiment task with GEPA

PnL computation. We construct a daily long-short strategy from model predictions. For each filing observed on day t, the model predicts a directional position $p_i \in -1, +1$, corresponding to DOWN or UP sentiment respectively. Let r_i denote the realized 30-day return following the filing. Daily portfolio returns are computed as the average position-weighted return across all filings observed on that day $R_t = \frac{1}{N_t}\sum_{i=1}^{N_t} p_i r_i$, where N_t is the number of filings on day t. The cumulative PnL is obtained by cumulatively summing the daily returns R_t over time. The optimized PnL curve is causal. After optimizing the prompt on year X, predictions generated using the optimized prompt are evaluated only on year $X+1$, reflecting a sequential deployment setting. Figure 8 shows that it is possible to construct a strategy with a positive PnL pre-Covid through a simple rule: attend only to a subset of sections. Similar to the observed accuracy drop in Figure 4, GEPA's optimisation fails to generalise in a causal forward-looking manner, dropping PnL performance.

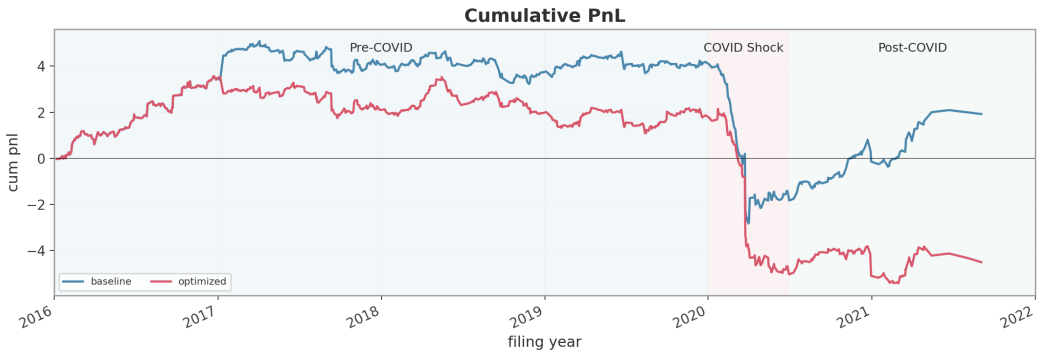


Figure 8: A PnL constructed from retaining only Items 2, 5, 7, 7A, 8, 9, and 9A from the 10-K filings. These sections capture the firm's physical operating footprint, shareholder and capital-allocation information, management's discussion of performance and risks, market-risk exposures, core financial statements, auditor changes or disagreements, and internal-control quality.

Section-wise sentiment with possibility to abstain We additionally explored a setup in which the model was allowed to abstain from a prediction for a given filing sections during GEPA optimization, as not all sections are considered equally informative. Figure 9 reports the relative change in section utilization frequency under this setup. Some sections became increasingly downweighted, while others were used more heavily by the optimized prompts. Notably, some down or upweighted sections indeed led to accuracy increases, but not consistently. Equally, Section 1A ("Risk Factors"), which prior work has shown to contain predictive risk sentiment information (Magner et al., 2025), remained largely unchanged. This may indicate that the optimizer failed to infer its predictive usefulness.

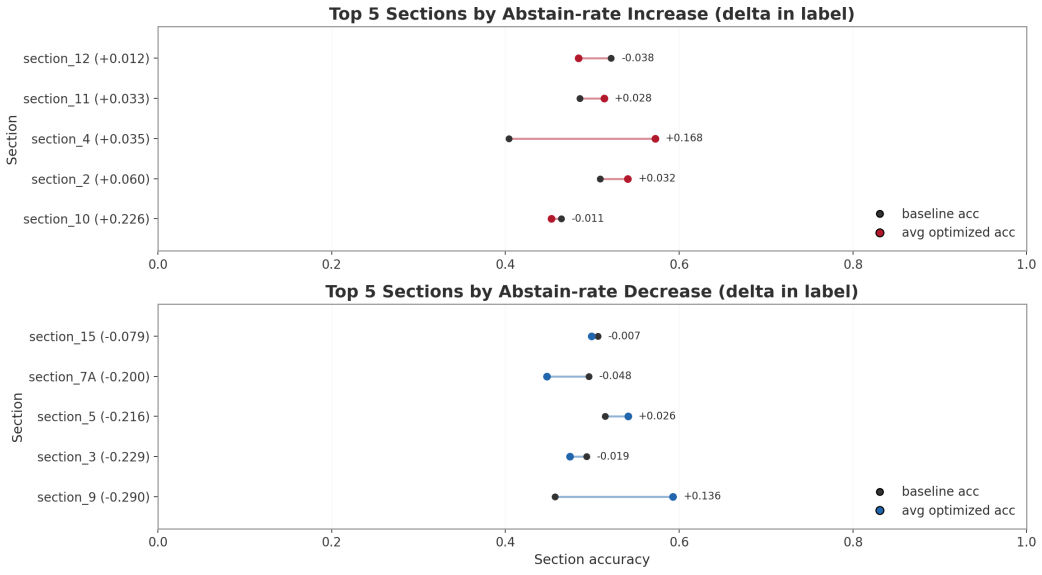


Figure 9: The change in abstain-rate per section. On the aggregate, predicting more often (abstain-rate decrease) or abstaining more often (abstain-rate increase) does not consistently lead to an increase in accuracy, showing that GEPA fails to extract section-level usefulness.

A.2.5 AGENTIC.

Update-mechanism hyperparameters. The two update mechanisms plotted in Figure 6 are configured as follows.

ACE (Qwen3-32B). An online batch pipeline runs each batch of 5 chains live in the browser, then reflects and curates. The reflector runs once per chain, tagging each existing bullet helpful, harmful, or neutral; the curator runs once per batch over the combined batch reflections with ADD-only operations under a 50,000-token budget. The reflector/curator LLM is Bedrock `qwen.qwen3-32b-v1:0` via the Converse API, with `temperature=` 0.3 and `maxTokens=` 4096. The playbook has five fixed sections (`strategies_and_insights`, `common_mistakes_to_avoid`, `problem_solving_heuristics`, `ui_navigation_tips`, `others`) and bullets in the format `[slug-NNNNN] helpful=H harmful=X :: content`. We run 1 epoch over 5 batches (≈25–30 chains, no held-out test) and 5 seeds (`seed0–seed4`), each shuffling the chain order. At evaluation the playbook is prepended to the browser-use system prompt, the agent (qwen3-32b) decodes at factory-default settings, and each cell covers 5 seeds × 30 chains. Per-horizon action budgets and timeouts are L1: 10 steps / 300s, L3: 20 / 900s, L5: 25 / 1200s, L10: 40 / 1800s.

SFT (Qwen3-8B). We fine-tune Qwen3-8B with LoRA (rank 32, $\alpha = 64$) on the `q,k,v,o,gate,up,down` projections, learning rate 5×10^{-5}, gradient accumulation 4, maximum sequence length 16,384, for 1 epoch, since training beyond one epoch overfit and decreased performance on some task-suite tasks. Trajectories are distilled from a teacher agent (Qwen3-32B for L1–L5, Qwen3-235B for L10); per-horizon trajectory and training-example counts are given in Table 10.

Table 10: Distillation data for the SFT mechanism, by horizon length. Traj.: kept (successful) trajectories; Train/Val: trajectory split; Steps: training-step examples.

Level	Tasks	Traj.	Traj./task	Train/Val	Steps	Steps/traj	Teacher
L1	46	612	13.3	551 / 61	5,485	∼10	Qwen3-32B
L3	96	1,360	14.2	1,224 / 136	22,074	∼18	Qwen3-32B
L5	132	1,536	11.6	1,383 / 153	32,199	∼23	Qwen3-32B
L10	242	2,299	9.5	2,070 / 229	69,674	∼34	Qwen3-235B

Suite construction. We build the agentic suite on top of WebArena-Infinity (Zhou, 2026). Chain generation is seeded from two sources per app: the app's description, which enumerates the available features, and its underlying state files, which fix the concrete entities those features can act on. Conditioning on both lets us generate chains whose steps reference entities that actually exist in the environment. From these we generate sequential chains of length 1, 3, 5, and 10, in which step $i{+}1$ depends on a concrete piece of state that step i leaves behind in the environment. In Gmail, for instance, the agent creates a label, renames it, and then applies the renamed label to a specific email, so that each later step is only well-defined given the state produced by the earlier ones. The full suite contains 510 chains across four apps (Gmail, Gmail Contacts, Handshake, PayPal), spanning the four horizon lengths.

Verification. Each step is paired with a programmatic verifier that reads the app's state directly rather than parsing the agent's narration, so success is checked against the environment. A step is marked correct only when its verifier confirms the intended state change, and a chain of length L succeeds only if all L of its steps pass. Grounding verification in state ensures that an agent receives no credit for claiming to have completed a step it did not actually perform.

Agent harness. All runs use a `browser-use` agent driving a headless Chromium against the live app. Within a chain we keep a single agent instance alive across all steps: the browser session is not reset between steps, and each follow-up instruction is appended to the same conversation rather than starting a fresh context, so that when acting on step $i{+}1$ the model sees its own prior trajectory—observations, actions, and errors—from the preceding steps. Each step is given a fixed action budget and is scored independently by its verifier.

Zero-shot models. In addition to the learning methods reported in the main body, we evaluate Deepseek-v3.2 (Liu et al., 2025a), GPT-OSS-120B (Agarwal et al., 2025), and Gemini-2.5-flash-lite (Comanici et al., 2025) zero-shot under the identical harness. All three degrade sharply as chain length grows (Figure 6). The decay tracks the point at which the accumulated trajectory approaches the model's context window: once earlier trajectory state has to be summarized to fit, the model loses access to the precise environmental cues it needs to ground later actions. Gemini's larger context window does not rescue this failure mode—it degrades faster than Deepseek despite more available context—suggesting that the bottleneck is the agent's ability to actually *use* long trajectory information rather than raw context capacity.

The same "does experience stick across tasks" question, but where the chain arises from the agent's own actions rather than from a fixed ordering we picked. A learner operates in sequential chains of web tasks built on WebArena Infinity (Zhou, 2026), where the environment produced by task i is the starting state of task $i{+}1$: an action taken early in the chain (an archived email, an applied label, a navigation state) persists and affects every subsequent task's success. We measure end-to-end pass rate as a function of chain length $L \in \{1, 3, 5, 10\}$, across three apps (Gmail, Gmail Accounts, PayPal Wallet) and three backbones (Deepseek-v3.2 (Liu et al., 2025a), Gemini-2.5-Flash-Lite (Comanici et al., 2025), GPT-OSS-120B (Agarwal et al., 2025)). This is the closest of our benchmarks to deployed agentic use; we treat it as the natural-regime counterpart to Domain shift's controlled regime.

Table 11: Chain pass rate on the `gmail` app, by model and horizon length. Denominator is 30 chains at every length.

Model	Context	L1	L3	L5	L10
deepseek-v3.2	128k	70%	30%	23%	0%
gemini-2.5-flash-lite	1M	47%	7%	7%	0%
gpt-oss-120b	128k	60%	40%	17%	0%

Table 12: Chain pass rate on the `gmail-contacts` app, by model and horizon length. Denominators are 55/32/20/12 chains at L1/L3/L5/L10.

Model	Context	L1	L3	L5	L10
deepseek-v3.2	128k	75%	91%	85%	8%
gemini-2.5-flash-lite	1M	49%	25%	5%	0%
gpt-oss-120b	128k	40%	21%	0%	0%

Table 13: Chain pass rate on the `paypal` app, by model and horizon length. Denominators are 50/24/14/12 chains at L1/L3/L5/L10.

Model	Context	L1	L3	L5	L10
deepseek-v3.2	128k	80%	67%	7%	17%
gemini-2.5-flash-lite	1M	72%	29%	0%	0%
gpt-oss-120b	128k	68%	29%	14%	0%

Table 14: Chain pass rate on the `handshake` app, by model and horizon length. Denominators are 70/41/30/30 chains at L1/L3/L5/L10.

Model	Context	L1	L3	L5	L10
deepseek-v3.2	128k	80%	37%	27%	7%
gemini-2.5-flash-lite	1M	44%	7%	0%	0%
gpt-oss-120b	128k	57%	17%	7%	0%