TEST-TIME ADAPTATION OF REASONING STRATEGIES WITH BAYESIAN NONPARAMETRIC MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

While modern large language models (LLMs) have been trained to reason through verbalized chains-of-thought, the generation cost grows substantially due to suboptimal paths to reach the final answer. Furthermore, as new insights are discovered while observing various input queries (e.g. through self-reflection), limited mechanisms exist for carrying forward these findings to be applied to subsequent problems. One can view the list of such strategies or behaviors as a growing cheatsheet, with elements retrieved from this memory module at inference-time. In this work, we consider *structured* cheatsheets, with learned clusters of behaviors. We introduce a Hierarchical Dirichlet Process Gaussian Mixture Model (HDP-GMM) over behavior embeddings, which shares components across domains while allowing domain-specific mixing weights, and uses the posterior predictive to retrieve relevant behaviors for a query; we call this a *Bayesian Cheatsheet*. This mechanism allows for cheap adaptation in an online test-time training (TTT) setting, softly updating the mixture's sufficient statistics following each sample and enabling the creation of new components when the synthesized behaviors are sufficiently novel. We demonstrate that Bayesian Cheatsheet achieves clear performance gains relative to existing memory modules across reasoning benchmarks such as AIME'25, Omni-MATH, and PhysReason, even in the cold-start setting. We show that the Bayesian Cheatsheet is an adaptively reorganizing memory module, as behaviors can be re-assigned to components through a single step of collapsed Gibbs sampling. Our findings highlight the value of Bayesian-inspired memory modules for effective test-time adaptation and the role of structure in metacognitive reasoning.

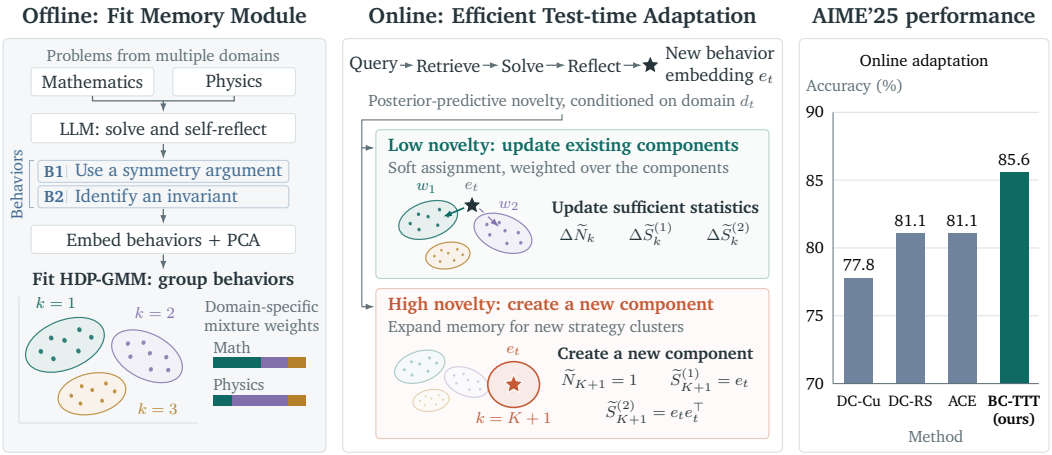


Figure 1: Bayesian Cheatsheet learns a structured memory module offline and adapts it at test time. Left: behaviors are mined offline by an LLM solving problems and self-reflecting on its solutions; the extracted behaviors are embedded and clustered into shared components with domain-specific weights by fitting an HDP-GMM. Middle: at test time, a novelty score based on the HDP-GMM's posterior predictive determines whether a new behavior updates existing components' sufficient statistics (top) or spawns a new component and initializes its sufficient statistics (bottom). Right: evaluation on AIME'25 with Granite-4.2-30B shows that Bayesian Cheatsheet (BC-TTT) outperforms the compared adaptive memory mechanisms in the online setting.

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.

1 INTRODUCTION

Large language models increasingly solve complex problems by generating verbalized intermediate reasoning steps (Wei et al., 2022; Nye et al., 2021). Given that related problems can often be solved with similar strategies, storing such strategies, which we call *behaviors*, can improve reasoning efficiency in solving future problems — that is, they do not need to be rediscovered. This is a key premise of metacognitive reasoning (Shinn et al., 2023; Didolkar et al., 2024; 2025), where reflecting on reasoning to extract generalizable lessons and using those lessons to guide subsequent attempts have been shown to be effective. At the same time, this begs the question: how can we design systems to optimally retain and retrieve useful abstractions for LLM reasoning and decision-making?

Recent methods accumulate and revise reasoning behaviors, reflections, and procedural guidance over time (Didolkar et al., 2025; Suzgun et al., 2026; Zhang et al., 2026b; Ouyang et al., 2026). As memory grows, context limits and costs make it desirable to maintain an external, structured memory with selective reuse to avoid context explosion. Existing structured memory systems (Xu et al., 2025; Chhikara et al., 2025) often rely on LLM judgments and similarity-based heuristics to define and evolve the memory structure; this organization can become difficult to maintain as the memory grows.

In this work, we use the term *cheatsheet* to refer to a general abstraction: it is a collection of concise, natural-language insights that can be consulted while solving a new problem. A cheatsheet induces two key design questions: 1) what type of information should be stored? and 2) how should it be organized for effective reuse and continual adaptation? In this work, we focus on the latter, while for the former, we simply use short reasoning behaviors generated through self-reflection to form the memory content. This allows for a clear comparison of memory organization mechanisms without the confounding effect of differences in the memory content. As a cheatsheet grows, its organization must support more than the retrieval of similar entries — related behaviors may form groups, which may also be reusable across domains. New experiences encountered at test-time may reinforce the contents of an existing group, or reveal a strategy that is not captured under the current organization. However, static cheatsheets, as well as flat similarity-based retrieval, do not address this, and while clustering provides additional organization for reuse, it does not facilitate principled growth and requires the number of clusters to be known in advance.

We introduce the Bayesian Cheatsheet, a structured memory module based on a Hierarchical Dirichlet Process Gaussian Mixture Model (HDP-GMM) (Rasmussen, 2000; Teh et al., 2006) over behavior embeddings. The HDP-GMM formulation, as a non-parametric unbounded clustering mechanism, offers a few benefits that are particularly desirable for test-time memory management: 1) the number of clusters is purely driven by the data seen thus far, 2) the hierarchical structure offers a flexible mechanism to distinguish generally usable memories from domain-specific ones, and 3) the posterior probabilities of cluster assignment serve a dual purpose, in assigning memories to clusters as well as input-conditioned retrieval. The HDP prior additionally induces a rich-get-richer bias – memory components repeatedly observed in experience become more likely to be used in the future, while new components may still be created.

Our Bayesian Cheatsheet memory learns globally shared components (clusters) of behaviors, allowing for probabilistically latent assignment. At inference time, it predicts the query's domain and retrieves behaviors from components to add to the language model's context. In an online test-time training setting with streaming samples, Bayesian Cheatsheet cheaply incorporates newly mined behaviors into the memory through additive sufficient statistics, while also permitting the creation of new components for samples where the behaviors are sufficiently novel. Bayesian Cheatsheet also allows the memory to periodically reorganize itself as its contents evolve without requiring a refitting step every sample. As a result, this mechanism supports lightweight continual adaptation in a statistically-informed, training-free manner through principled changes to an external memory module.

2 RELATED WORK

Test-time adaptation and continual learning. Early test-time training methods adapted models through self-supervised losses on test inputs (Sun et al., 2020), with later work on language models drawing adaptation data from retrieved text (Hardt & Sun, 2024) or input–output demonstrations supplied with a reasoning task (Akyürek et al., 2025). Continual learning additionally aims to retain previously learned capabilities across updates; Learning without Forgetting (Li & Hoiem, 2016) and

elastic weight consolidation (Kirkpatrick et al., 2017) constrain changes in predictions and important parameters, respectively. On-policy distillation trains a student on its own generated sequences using a teacher's predictions (Agarwal et al., 2024); in self-distillation, the model itself supplies the teacher predictions when conditioned on richer information. Self-distillation fine-tuning uses demonstrations to acquire skills with less forgetting (Shenfeld et al., 2026), and related methods distill guidance from textual feedback (Hübotter et al., 2026), verified solutions (Zhao et al., 2026), or subsequent user messages (Kleine Buening et al., 2026). While these methods parametrically update language model weights, we focus on continual adaptation through an external memory.

Memory and retrieval-augmented reasoning with LLMs. Retrieval-augmented generation (Lewis et al., 2020) provides external evidence at inference time by retrieving relevant information from a corpus or memory store, with structures ranging from vector indexes to interconnected knowledge graphs (Jiménez Gutiérrez et al., 2025). As these memories grow, their contents must also be managed and revised: MemGPT manages information beyond the context window (Packer et al., 2023), Mem0 consolidates conversational records (Chhikara et al., 2025), and A-MEM links and updates structured notes (Xu et al., 2025). For reasoning, ExpeL and Metacognitive Reuse extract reusable guidance from prior problem solving (Zhao et al., 2024; Didolkar et al., 2025), while Dynamic Cheatsheet, ACE, and ReasoningBank maintain and refine strategies as experience accumulates (Suzgun et al., 2026; Zhang et al., 2026b; Ouyang et al., 2026). Recent approaches also learn how memory is constructed or used: ALMA searches over executable memory designs (Xiong et al., 2026), and MILES learns selection over an expanding library of step-level instructions (Tong & Gong, 2026). The focus in this work is on the probabilistic organization of such memory content.

3 Structured Retrieval for Metacognitive Reasoning

Problem Setup. We consider J domains $\mathcal{D} = \{1, \ldots, J\}$ (in our experiments, math and physics), each with a corpus of prompts. From these prompts we mine *behaviors* – short natural-language reasoning strategies, extracted by having the model solve a problem, reflect on which general strategies its solution used, and record them (details in Appendix D) – and embed their descriptions with a fixed sentence encoder $\mathcal{E} : \mathcal{S} \to \mathbb{R}^{D_0}$ (where $\mathcal{S}$ is the space of text strings, including queries and behaviors), then project into a lower-dimensional space via $f : \mathbb{R}^{D_0} \to \mathbb{R}^D$ (PCA, fit once over the mined corpus). We use $e_{d,i} \in \mathbb{R}^D$ to denote the projected embedding of the i-th behavior mined from domain d, and n_d for the number of behaviors mined from domain d.

At inference, we condition generation on a small subset of behaviors selected for relevance to the current query x: given $e(x)$, a *selection rule* Sel returns a set of behaviors whose natural-language descriptions are inserted into the prompt. In the *online* setting we study in this work, new pairs of (query, mined-behavior) arrive sequentially as new samples are seen, and the organizing structure itself must be able to adapt and incorporate each one, and optionally, perform periodic reorganization. We formalize any such structure with these properties as three operations on a state Ω, which represents the current memory organization:

$$\mathrm{retrieve} : \mathbb{R}^D \times \Omega \to \Delta^K, \qquad \mathrm{update} : \mathbb{R}^D \times \Omega \to \Omega, \qquad \mathrm{refit} : \Omega \to \Omega$$

Here, Δ^K is the simplex of weights over the set of components. If the mechanism has a flat structure, K is simply the total number of behaviors (each behavior is treated as its own component); otherwise, if there is clustering, we take K to be the number of currently active components. This allows us to consider different practical instantiations (organizing structures) under a consistent online setup, with these being the central three operations of the online TTT loop.

Starting Point: *k*-means clustering. The simplest instantiation partitions $\mathbb{R}^D$ into K hard clusters with centroids $\mu_1, \ldots, \mu_K$, i.e. $\Omega = (\mu_1, \ldots, \mu_K, n_1, \ldots, n_K)$ where n_k counts points currently assigned to cluster k.

$$\mathrm{retrieve}(e(x), \Omega) = \mathrm{softmax}_k\left(-\frac{\|e(x) - \mu_k\|^2}{2\sigma^2}\right), \qquad \sigma^2 = \frac{1}{K}\sum_k \sigma_k^2,$$

where σ_k^2 is the scalar cluster variance. Given a newly mined behavior with embedding e_t, the $\mathrm{update}(e_t, \Omega)$ operator allows for the mechanism to grow the number of clusters by checking

whether $1 - \max_j \cos(e_t, \mu_j)$ exceeds a threshold τ_{KM}. If so, it creates a new cluster with centroid $\mu_{K+1} = e_t$ and count $n_{K+1} = 1$. Otherwise, it assigns the behavior to the nearest centroid $k^\star$ and updates, while every other centroid is unchanged:

$$\mu_{k^\star} \leftarrow \frac{n_{k^\star}\mu_{k^\star} + e_t}{n_{k^\star} + 1}, \qquad n_{k^\star} \leftarrow n_{k^\star} + 1$$

The $\text{refit}(\Omega)$ operator entails one full pass of Lloyd's algorithm: all accumulated points are reassigned to their nearest centroids, each centroid is recomputed as the mean of its assigned points, and any remaining empty clusters are removed.

This approach has a few key limitations. First, hard assignment assumes clear specialization of the components in natural language, while behaviors perhaps should actually lie in multiple clusters, and k-means requires prescribing the number of clusters a priori for its initialization; we want this to be determined in a data-driven manner. Second, "growth" (in terms of the creation of new components) is not tied to the probability-based retrieval mechanism, which can lead to many singleton components or a few overloaded components. Also, k-means remains flat and unable to express the notion of domain-specificity; we want a structure that can encode domain-awareness while allowing for behaviors to possibly be shared across domains, motivating a hierarchy.

4 Bayesian Cheatsheet: Behavior Mixtures via HDP-GMM

We develop the **Bayesian Cheatsheet**, a nonparametric instantiation of a structured cheatsheet using a Dirichlet Process Gaussian Mixture Model (DP-GMM) (Rasmussen, 2000) to model behavior embeddings and counts for each component. This serves as an analogue to the k-means approach, with probabilistic latent assignment in lieu of hard assignment, countably infinite components with probability-driven growth rather than a fixed number, and an extension to a Hierarchical DP-GMM for domain-aware modeling. The detailed algorithm is included in Appendix B.

4.1 Generative Mixtures with Dirichlet Processes

First consider a single domain. We draw a random probability measure:

$$G \sim \text{DP}(\gamma, H), \qquad H = \text{NIW}(\mu_0, \kappa_0, \Psi_0, \nu_0)$$

with the NIW distribution being the conjugate base distribution over Gaussian parameters. G is a discrete random probability measure, and thus can be written as

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}, \qquad \phi_k = (\mu_k, \Sigma_k), \qquad \phi_k \overset{\text{i.i.d.}}{\sim} H$$

as many observations can share the same atomic value ϕ_k. We model each behavior embedding e_i as:

$$z_i \mid \pi \sim \text{Cat}(\pi), \qquad e_i \mid z_i = k \sim \mathcal{N}(\mu_k, \Sigma_k).$$

Marginalizing over G yields the Chinese Restaurant Process (CRP). Conditioned on the previous assignments, a new behavior is assigned to an existing component k with probability proportional to its occupancy n_k, or instantiates a new component with probability proportional to γ. As a result, the number of components (K) is determined in a data-driven manner rather than fixed a priori.

Given a mined corpus of behavior embeddings X, the $\text{retrieve}(x)$ operation uses the posterior assignment probability:

$$p(z_x = k \mid e(x), X) \propto n_k \, p(e(x) \mid X_k) \tag{1}$$

where $X_k = \{e_i : z_i = k\}$ and $p(e(x) \mid X_k)$ is the posterior predictive density for component k. Correspondingly, the unnormalized score for spawning a new component is given by $\gamma \, p(e(x) \mid H)$. Under the NIW prior, the predictive densities are multivariate Student-t distributions.

Crucially, the posterior of each component depends only on three additive sufficient statistics:

$$n_k = |\{i : z_i = k\}|, \qquad S_k^{(1)} = \textstyle\sum_{i:z_i=k} e_i, \qquad S_k^{(2)} = \textstyle\sum_{i:z_i=k} e_i e_i^\top \tag{2}$$

Thus, incorporating a newly discovered behavior at test-time requires only incrementing these three values, allowing for online adaptation without needing to refit the full mixture after each observation.

4.2 Hierarchical Construction for Cross-Domain Structure

A single DP-GMM fit on a pooled corpus of cross-domain problems and behaviors cannot express the prevalence of components at the domain-specific level. At the same time, fitting independent DP-GMMs per domain can result in redundancy and unnecessary overhead, preventing components from being shared. For instance, a behavior like "use modular arithmetic" could be useful for a number theory problem as well as for periodic boundary conditions in a physics problem.

A Hierarchical Dirichlet Process (HDP) (Teh et al., 2006) addresses this by drawing the atoms $\{\phi_k\}$ once at the global level, and letting each domain draw its own DP over them, with $\theta_{d,i} = (\mu_{d,i}, \Sigma_{d,i})$:

$$\begin{aligned} G_0 &\sim \mathrm{DP}(\gamma, H), \\ G_d \mid G_0 &\sim \mathrm{DP}(\alpha_d, G_0), \quad \forall\, d \in \mathcal{D}, \\ \theta_{d,i} \mid G_d &\sim G_d, \\ e_{d,i} \mid \theta_{d,i} &\sim \mathcal{N}(\mu_{d,i}, \Sigma_{d,i}) \end{aligned}$$

Thus, the Gaussian parameters $\phi_k = (\mu_k, \Sigma_k)$ are shared across domains, while the mixing weights π_d are domain specific. Marginalizing the global and domain-specific random measures yields the Chinese Restaurant Franchise (CRF) representation.

As in the single-domain construction, let $z_{d,i} = k$ denote that behavior (d, i) uses the shared atom ϕ_k. Conditional on π_d, this assignment has probability $p(z_{d,i} = k \mid \pi_d) = \pi_{d,k}$. We distinguish the local occupancy of component k in domain d, $n_{d,k} = |\{i : z_{d,i} = k\}|$, from its global occupancy $N_k = \sum_{d \in \mathcal{D}} n_{d,k}$. The local counts determine the domain-specific mixing weights, whereas the posterior of the shared Gaussian component uses observations pooled across domains. The sufficient statistics (Equation 2) generalize by using N_k and summing $S_k^{(1)}$ and $S_k^{(2)}$ over every domain.

4.3 Posterior Inference via Collapsed Gibbs Sampling

To fit the HDP-GMM, we use a collapsed Gibbs sampler (Neal, 2000) in the CRF representation, leveraging NIW-Gaussian conjugacy in marginalizing out the Gaussian parameters ϕ_k. We also integrate out the random measures, retaining local groups and their mappings to shared global components (Teh et al., 2006). Marginalizing over the local-group choice gives the assignment probability for an existing component:

$$p(z_{d,i} = k \mid \cdot) \propto \left(n_{d,k}^{-i} + \alpha_d \widehat{\beta}_k^{-i}\right) p_k^{-(d,i)}(e_{d,i}), \qquad k = 1, \ldots, K \tag{3}$$

Here, $n_{d,k}^{-i}$ counts behaviors assigned to component k within domain d, $\widehat{\beta}_k^{-i}$ is the global predictive mass derived from the group-to-component counts, and $p_k^{-(d,i)}$ is the Student-t posterior predictive computed from behaviors pooled across domains – all excluding observation (d, i).

Relative to the single-domain posterior rule (1), the hierarchical structure lets the memory reflect both domain-specific patterns as well as cross-domain reuse. The local count $n_{d,k}^{-i}$ concentrates observations onto prevalent components that are already common within domain d, following the "rich-get-richer" effect associated with DPs. The $\alpha_d \widehat{\beta}_k^{-i}$ term allows a domain to draw on components learned across the corpus through the mixing weights. The predictive density weights this by how well the behavior's embedding fits the component, so assignment depends on the both the behavior itself and the frequency of its component. Overall, this allows the memory to share existing reasoning patterns across domains while also accomodating additional components as behaviors accumulate.

A complete step of partially collapsed Gibbs resamples every $z_{d,i}$ in random order and recomputes both the local counts and the globally pooled sufficient statistics. Offline fitting of the HDP-GMM repeats this procedure for a fixed iteration count.Further mathematical details on the two-level sampling procedure for our collapsed Gibbs sampling process are included in Appendix C.

4.4 Test-Time Adaptation

Retrieval via the posterior predictive. In the offline phase, the embedding used is the behavior's own embedding (applying $\mathcal{E}$), and we apply PCA for efficiency in fitting the mixture model (described

in Appendix D.2). At inference time, $e(x)$ is approximated with a small learned projector f_θ to map raw query embeddings into the same space (see Appendix D). With domain-conditioning, the query's domain is inferred using a nearest-centroid classifier, where c_d is the centroid of domain d's data:

$$d_x = \widehat{d}(x) = \arg\min_{d \in \mathcal{D}} \mathrm{dist}(e(x), c_d)$$

Then $\mathrm{retrieve}(e(x), \Omega)$ combines these domain-specific mixture weights with the globally shared component predictives, so domains can share the same components but retrieve them with different probabilities. The normalized weight for each component under the Student-t posterior predictive is:

$$w_{d_x,k}(e(x)) = \frac{\widehat{\pi}_{d_x,k}\, p_k(e(x))}{\sum_{j=1}^{K} \widehat{\pi}_{d_x,j}\, p_j(e(x))} \tag{4}$$

We consider component selection either by taking the top-k' components by weight or choosing every component whose weight exceeds a threshold τ. Once components are selected, the medoid element is returned for each component, treated as its representative element.

Online updates and new component creation. After generating a response to query x_t, the model mines a new behavior by self-reflection on its reasoning, with embedding e_t and domain $d_t = \widehat{d}(x_t)$. A discrete CRF partition produces the mixing weights, while responsibility-weighted Gaussian statistics determine the online densities; the predictive probability of a new component is:

$$\rho_{\mathrm{new}}(e_t) = \frac{\widehat{\pi}_{d_t,\mathrm{new}} p_0(e_t)}{\widehat{\pi}_{d_t,\mathrm{new}} p_0(e_t) + \sum_{k=1}^{K} \widehat{\pi}_{d_t,k}\, p_k(e_t)} \tag{5}$$

If $\rho_{\mathrm{new}}(e_t)$ exceeds a threshold τ_{new} (or the memory is empty), we spawn a new globally shared component, initialize its Gaussian sufficient statistics using e_t, and create a local group containing this behavior. Otherwise, it is softly added to existing components with $w_{d_t,k}(e_t) > 10^{-6}$, weighted accordingly. The effective Gaussian sufficient statistics (denoted with tildes) for $\mathrm{update}(e_t, \Omega)$ are:

$$\widetilde{N}_k \leftarrow \widetilde{N}_k + w_{d_t,k}(e_t), \qquad \widetilde{S}_k^{(1)} \leftarrow \widetilde{S}_k^{(1)} + w_{d_t,k}(e_t) e_t, \qquad \widetilde{S}_k^{(2)} \leftarrow \widetilde{S}_k^{(2)} + w_{d_t,k}(e_t) e_t e_t^\top \tag{6}$$

We also separately store the hard assignment $z_t = \arg\max_k w_{d_t,k}(e_t)$ and sample its local group assignment conditional on z_t, incrementing the local group element count and n_{d_t,z_t}. If a new local group is created, we also increment M_{z_t}, the global group count.

Periodic refitting. Every g samples (the refit interval), we perform $\mathrm{refit}(\Omega)$ by rebuilding Gaussian statistics from the stored hard assignments and performing a single collapsed Gibbs step over all behaviors, keeping domain labels fixed. We then greedily merge components whose symmetric Gaussian KL-divergence is $< \epsilon$, updating group mappings, counts, and statistics. This allows for reorganization to reconcile drift that may be introduced by performing many soft updates over time.

5 Results

5.1 Experimental Setup

Models. All experiments are performed with the Qwen3-32B (Yang et al., 2025) and Granite-4.2-30B (IBM Granite Team, 2026) language models, with thinking/reasoning modes enabled. Allowing language models to leverage their reasoning abilities to the fullest ensures that retrieved reasoning behaviors from cheatsheets can be reflected in their natural language reasoning traces.

Datasets. We evaluate on three reasoning-intensive benchmarks: AIME'25 (Art of Problem Solving, 2025), Omni-MATH[1] (Gao et al., 2025), and PhysReason[2] (Zhang et al., 2025), averaged over 3 seeds. Our behavior corpus (existing memory) was mined offline from NuminaMath-CoT (Li et al., 2024) and the Camel Physics (Li et al., 2023) datasets; the respective language model solves each problem and produces a self-reflection of the behaviors that were useful in solving the problem, regardless of the answers' correctness. The corpus was mined separately for each language model, providing an on-policy reasoning bank. Further details on the behavior corpus are included in Appendix D.1.

[1] Specifically, we consider a hard subset consisting only of problems with $\mathrm{difficulty} \geq 7.0$.

[2] We consider solely the questions which do not require an image (i.e. text-only), given that the open-weight language models we consider do not have multimodal capabilities, yielding 226 questions (mini has 31).

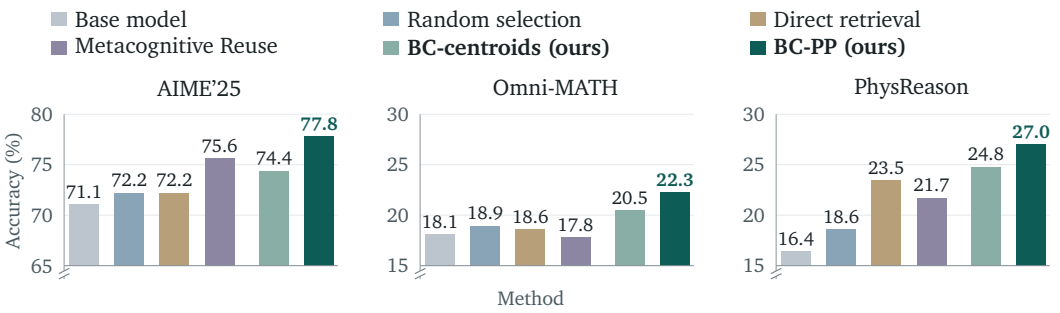


Figure 2: Static retrieval with Qwen3-32B using the mined behavior corpus. Bayesian Cheatsheet – specifically, our posterior predictive strategy (BC-PP) – achieves the strongest performance across the board, while our top-k' nearest-centroid variant also performs well relative to the baselines.

Baselines. We consider a number of baselines, evaluated both with and without the pre-mined memory, with the key differences being the retrieval and adaptation mechanisms. This includes: 1. random behavior selection of k' elements; 2. Metacognitive Reuse (Didolkar et al., 2025), which involves flat cosine-similarity retrieval over the behavior library (top-k') in embedding space; and 3. direct behavior retrieval, performing flat retrieval in the f_θ-projected space rather than embedding space. For the dynamic (test-time adaptation) setting, we consider the Dynamic Cheatsheet (Suzgun et al., 2026) method (both DC-Cu and DC-RS), which grows the library as it solves, with up to max_new=3 elements added per sample, and has the ability to rewrite and refine the cheatsheet purely in natural language (as it has a flat structure). We also consider the ACE method (Zhang et al., 2026b) which, on top of Dynamic Cheatsheet, introduces iterative refinement through a reflector as well as editing and deduplication. In practice, we use $k' = 5$ for all experimental results.

Methods. We evaluate the Bayesian Cheatsheet (BC) as described in Section 4, as well as k-means clustering as introduced in Section 3. We also include BC-centroids, which selects the k' closest centroids over the HDP-GMM-fitted components, rather than posterior predictive selection. This allows us to isolate the role of 1. structure (these mechanisms against the baselines), 2. adaptation (HDP-GMM vs k-means), 3. flexible assignment (hard k-means vs soft k-means and HDP-GMM), and 4. component selection (posterior predictive vs. nearest-centroid top-k'). To bring the k-means algorithms to a level playing field with the HDP-GMM mechanism, we introduce adaptive versions with online updates for each, along with analogues for the Gibbs refit. Hard k-means assigns each new behavior to its nearest centroid and updates that centroid, while refitting is just Lloyd reassignment. For soft k-means (an isotropic generative variant), the online updates fold each new behavior into every cluster with weight above 10^{-6}, with the refitting being a responsibility-weighted soft EM procedure. To standardize across methods, we take K (number of initial clusters for k-means methods) to be the same number of components yielded in fitting the HDP-GMM, yielding 64 clusters (Qwen), and 62 clusters (Granite) – we include a list of labels for each component in Appendix G.1.

5.2 Bayesian Cheatsheets Improve Metacognitive Reasoning

Figure 2 and Appendix Table 4 show that BC-PP improves over Metacognitive Reuse by +2.2–5.3 points for Qwen and +1.3–3.2 for Granite across all three benchmarks. While BC-PP generally outperforms BC-centroids (with the exception of AIME'25 and a tie for PhysReason on Granite), the latter outperforms the baselines in nearly all settings. Since these methods have access to the same offline-mined corpus for each respective language model, our gains indicate that memory structure and component selection are vital. Furthermore, the results with BC-PP indicate that there is additional value yielded through the posterior-predictive selection mechanism beyond organizing behaviors into components. Thus, effective reasoning strategy reuse depends not only on the availability of useful behaviors, but also on how the memory makes them accessible for a given query.

5.3 Bayesian Cheatsheet Enables Online Adaptation

We next evaluate Bayesian Cheatsheet in the online adaptation setting, where we denote it as BC-TTT. As described in Section 4.4, we allow for new behaviors to be generated through self-reflection and subsequently either added to existing components or used to seed new components that are spawned.

Table 1: Static prompting vs. online adaptation, with and without periodic refitting ($g = 50$ for OM and PR, $g = 10$ for AIME'25), for Qwen3-32B and Granite-4.2-30B. The static version of DC-RS is Metacognitive Reuse (in Table 4). DC-Cu and DC-RS involve "curation" where the cheatsheet can be refined (does not have a separate refit), while ACE similarly uses a curator for adaptation and does not have a static equivalent. We denote these entries with a "–". Bayesian Cheatsheet (BC-TTT) improves over existing methods across models and benchmarks, both with and without refit.

		Qwen3-32B			Granite-4.2-30B		
Method	Condition	AIME'25	OM	PR	AIME'25	OM	PR
Hard k-means	Static	73.3	20.3	18.7	75.6	27.0	18.3
	Online, no refit	75.6	24.5	23.5	77.8	28.6	20.4
	Online, with refit	74.4	24.8	24.8	76.7	31.6	24.6
Soft k-means	Static	73.3	21.6	23.6	75.6	28.4	23.2
	Online, no refit	75.6	23.4	28.5	78.9	26.9	22.0
	Online, with refit	76.7	23.1	29.1	77.8	29.8	<u>27.4</u>
DC-Cu	Static	73.3	17.9	18.1	74.4	26.8	18.1
	Online, curation	75.6	21.8	24.6	77.8	28.3	22.4
	Online, with refit	–	–	–	–	–	–
DC-RS	Static	75.6	17.8	21.7	77.8	28.4	22.6
	Online, curation	78.9	21.0	25.7	81.1	30.2	23.0
	Online, with refit	–	–	–	–	–	–
ACE	Static	–	–	–	–	–	–
	Online, curator	78.9	23.2	26.0	81.1	29.3	25.1
	Online, with refit	–	–	–	–	–	–
BC-TTT (ours)	Static	77.8	22.3	27.0	80.0	31.6	23.9
	Online, no refit	<u>81.1</u>	<u>26.4</u>	<u>30.1</u>	**85.6**	**34.6**	26.8
	Online, with refit	**82.2**	**26.6**	**31.3**	<u>84.4</u>	<u>34.3</u>	**30.7**

Our results in Table 1 show that online updates improve Bayesian Cheatsheet over static retrieval across both models and all benchmarks, even without periodic refitting. Our comparison against the k-means methods (described in Section 3) demonstrates that 1) soft assignment can be valuable, as soft k-means outperforms the hard assignment variant on AIME'25 and PhysReason for both models, and 2) the HDP-GMM structure helps with domain-conditioning in retrieval as well as principled growth in the online setting. Our comparisons against Dynamic Cheatsheet and ACE – methods which also incorporate experience through online memory updates, albeit in a flat structure – are particularly relevant. The gains of BC-TTT over these methods suggest that component-based online adaptation is effective; while allowing the organization of the cheatsheet to adapt based on the novelty of the behaviors encountered can help, it could also have tradeoffs, as seen with Granite slightly degrading on AIME'25 and Omni-MATH, but still continuing to outperform all other methods.

Periodic Refitting of Components. We quantify how much a refit changes the behavior-component assignment with a maximum-overlap bipartite matching. Let $\ell_{\text{before}}, \ell_{\text{after}}$ be the cluster labels assigned to the n accumulated points immediately before and after a refit (which may consist of different label sets and active label counts). Consider the contingency table:

$$M[b, a] = \left|\{i \in [n] : \ell_{\text{before}}(i) = b,\ \ell_{\text{after}}(i) = a\}\right| \tag{7}$$

We want to find the label correspondence maximizing total overlap over one-to-one matchings σ:

$$\sigma^\star = \arg\max_{\sigma} \sum_{b} M[b, \sigma(b)] \tag{8}$$

This is an assignment problem, which is solvable with the Hungarian algorithm. A point is counted as a reassignment only if its label in ℓ_{after} differs from the matched image of its label in ℓ_{before}. A pure relabeling should receive zero churn, and we are only counting the points that change clusters:

$$\textit{churn} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[\ell_{\text{after}}(i) \neq \sigma^\star(\ell_{\text{before}}(i))]\,, \tag{9}$$

Figure 3 shows that Bayesian Cheatsheet's assignment churn remains near 4% across refits on PhysReason-mini. At the third refit, churn is 3.98%, compared to 12.31% for hard k-means and 11.19% for soft k-means. The gap becomes wider over the refit rounds, as both k-means methods have increasingly larger changes in component membership. As a result, this indicates that Bayesian Cheatsheet's online updates preserve more of the existing organization through refitting while component parameters and domain-specific weights still evolve as new behaviors are incorporated. Thus, churn is a measure of the stability of memory's organization, and while BC is still subject to changing through online adaptation, it is quite stable.

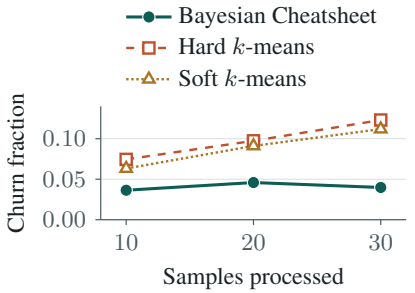


Figure 3: Assignment churn (Qwen3-32B) on PhysReason-mini; refits every 10 queries.

Cold-Start Memory Accumulation. Thus far, we have considered the setting where an offline-mined behavior corpus is available to the language model as memory in metacognitive reasoning. What if the language model were instead given a blank slate and tasked with learning the cheatsheet on-the-fly, purely through self-reflection? We evaluate our method (an empty memory initialized with an NIW prior) as well as ACE and DC-Cu (in their cold-start constructions), to determine whether Bayesian Cheatsheet is dependent on a warm-started memory module.

Table 2: Accuracy on Omni-MATH ($n = 916$) and PhysReason ($n = 226$) with no pre-mined behavior corpus. Memory-based methods start from an empty memory and accumulate behaviors through self-reflection at test time. Bayesian Cheatsheets use a periodic refit interval of $g = 50$.

	Qwen3-32B		Granite-4.2-30B	
Method	OM	PR	OM	PR
Base model (no memory)	18.1	16.4	24.8	<u>16.4</u>
DC-Cu, cold-start	<u>22.5</u>	15.9	25.5	15.3
ACE, cold-start	21.0	<u>17.7</u>	<u>27.9</u>	<u>16.4</u>
BC-TTT, cold-start (ours)	**24.1**	**20.5**	**31.9**	**18.1**

Our findings in Table 2 with Omni-MATH and PhysReason show that even without an offline-mined corpus, Bayesian Cheatsheet outperforms cold-start baselines for both language models and clearly improves over the memory-free setting. PhysReason seems particularly difficult to improve on in the cold-start setting (DC-Cu regresses), suggesting that physics benefits from more extensive guidance; nonetheless, BC-TTT still achieves gains. This demonstrates that Bayesian Cheatsheet can construct useful behavior memory purely from test-time experience, aided by its dynamic reorganization capabilities with periodic refitting, achieving effective adaptation to new data.

6 Discussion

We introduce the *Bayesian Cheatsheet*, a structured memory module for language models which uses Dirichlet Process Gaussian Mixture Models for component structure. We leverage this mechanism to organize reasoning strategies for inference-time retrieval, and demonstrate that this mechanism exhibits desirable properties in continual test-time adaptation while achieving clear gains in performance across reasoning benchmarks. Bayesian Cheatsheets offer a potential mechanism for offline domain adaptation through their hierarchical design; new behaviors and mixing weights can be learned for unseen domains, while also inferring shared cross-domain structure. In this work, we focus on single-turn problem solving with small-to-medium-size non-agentic language models ($\approx 30B$ parameters); while agentic and multi-turn benchmarks are currently beyond our scope, they offer practical potential for iterative retrieval from the Bayesian Cheatsheet. Furthermore, the refitting and retrieval costs eventually grow over indefinitely long evaluation streams, suggesting the need for principled *discarding* mechanisms to selectively retain behaviors. Overall, our findings highlight the potential for Bayesian-inspired methods to yield value for test-time adaptation of language models, given the setting's analogues to online learning, offering an effective, statistically-grounded mechanism.

AI Use Statement. We leveraged the use of AI tools in polishing the writing of this manuscript, as well as finding related works along with writing and debugging implementation code. It was also used as a coding assistant for qualitative analysis (see meta-labels in Appendix G.1). All AI-generated text and code was reviewed and edited by the authors, who take full responsibility for the content of this paper and its artifacts.

Reproducibility Statement. We include relevant details for reproducibility of our results in our experimental setup (Section 5.1), which includes the models and datasets, as well as details of our baselines and the settings we evaluate. Further details on our behavior mining procedure, fitting the HDP-GMM (including hyperparameters), and our inference mechanism are given in Appendix D. Our code will be released upon publication.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations (ICLR)*, 2024. https://arxiv.org/abs/2306.13649.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The Surprising Effectiveness of Test-Time Training for Few-Shot Learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 942–963. PMLR, 2025. https://arxiv.org/abs/2411.07279.

Art of Problem Solving. 2025 AIME I Problems. https://artofproblemsolving.com/wiki/index.php?title=2025_AIME_I_Problems, 2025.

Parul Awasthy, Aashka Trivedi, Yulong Li, Meet Doshi, Riyaz Bhat, Vignesh P, Vishwajeet Kumar, Yushu Yang, Bhavani Iyer, Abraham Daniels, Rudra Murthy, Ken Barker, Martin Franz, Madison Lee, Todd Ward, Salim Roukos, David Cox, Luis Lastras, Jaydeep Sen, and Radu Florian. Granite embedding R2 models, 2025. arXiv preprint arXiv:2508.21085. https://arxiv.org/abs/2508.21085.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to Optimally Memorize the Context at Test Time. *arXiv preprint arXiv:2505.23735*, 2025a. https://arxiv.org/abs/2505.23735.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to Memorize at Test Time. *arXiv preprint arXiv:2501.00663*, 2025b. https://arxiv.org/abs/2501.00663.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. https://arxiv.org/abs/2005.14165.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 2993–3000, 2025. doi: 10.3233/FAIA251160. https://arxiv.org/abs/2504.19413.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian LDA for Topic Models with Word Embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 795–804, 2015. https://aclanthology.org/P15-1077/.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving, 2024. arXiv preprint arXiv:2405.12205.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring LLM reasoning into concise behaviors, 2025. arXiv preprint arXiv:2509.13237.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. arXiv:2508.06433.

Jenny Rose Finkel and Christopher D. Manning. Hierarchical Bayesian Domain Adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 602–610. Association for Computational Linguistics, 2009. https://aclanthology.org/N09-1068/.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A Universal Olympiad Level Mathematic Benchmark for Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2025.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Contextual Dependencies in Unsupervised Word Segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 673–680, 2006. https://aclanthology.org/P06-1085/.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.18466.

Edward J. Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Esmeralda S. Whitammer. Amortizing intractable inference in large language models. In *International Conference on Learning Representations (ICLR)*, 2024. https://arxiv.org/abs/2310.04363.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large Language Models Cannot Self-Correct Reasoning Yet. In *International Conference on Learning Representations (ICLR)*, 2024. https://arxiv.org/abs/2310.01798.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation. *arXiv preprint arXiv:2601.20802*, 2026. https://arxiv.org/abs/2601.20802.

IBM Granite Team. Granite-4.2-30B. https://huggingface.co/ibm-granite/granite-4.2-30b, 2026.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. https://arxiv.org/abs/2502.14802.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pp. 641–648, 2007. https://papers.nips.cc/paper_files/paper/2006/hash/62f91ce9b820a491ee78c108636db089-Abstract.html.

Borui Kang, Jinrui Gu, Junhan Lv, Wenbin Li, Lei Wang, and Yang Gao. Harness Continual Learning: Continual Adaptation Beyond Model Parameters. *arXiv preprint arXiv:2608.19013*, 2026. https://arxiv.org/abs/2608.19013.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*, 2020. https://arxiv.org/abs/1911.00172.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning Language Models from User Interactions. *arXiv preprint arXiv:2603.12273*, 2026. https://arxiv.org/abs/2603.12273.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training Language Models to Self-Correct via Reinforcement Learning. *arXiv preprint arXiv:2409.12917*, 2024. https://arxiv.org/abs/2409.12917.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. *arXiv preprint arXiv:2303.17760*, 2023. Physics dataset available at https://huggingface.co/datasets/camel-ai/physics.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Numina, Hugging Face repository, 2024. https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Computer Vision – ECCV 2016*, pp. 614–629. Springer, 2016.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026. https://arxiv.org/abs/2603.03329.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000. doi: 10.2307/1390653.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2021. arXiv preprint arXiv:2112.00114.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. https://arxiv.org/abs/2310.08560.

Keshav Ramji, Tahira Naseem, and Ramón Fernandez Astudillo. Latent Principle Discovery for Language Model Self-Improvement. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025. https://arxiv.org/abs/2505.16927.

Carl Edward Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, pp. 554–560, 2000.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From Isolated Conversations to Hierarchical Schemas: Dynamic Tree Memory Representation for LLMs. *arXiv preprint arXiv:2410.14052*, 2024. https://arxiv.org/abs/2410.14052.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *International Conference on Learning Representations (ICLR)*, 2024. https://arxiv.org/abs/2401.18059.

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

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential Reinforcement Learning. *arXiv preprint arXiv:2602.13949*, 2026. https://arxiv.org/abs/2602.13949.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 2020.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 7080–7106, 2026. arXiv:2504.07952.

Yee Whye Teh. A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 985–992, 2006. https://aclanthology.org/P06-1124/.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. doi: 10.1198/016214506000000302.

Ruilin Tong and Dong Gong. MILES: Modular Instruction Memory with Learnable Selection for Self-Improving LLM Reasoning. *arXiv preprint arXiv:2607.06974*, 2026. https://arxiv.org/abs/2607.06974.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An Explanation of In-context Learning as Implicit Bayesian Inference. In *International Conference on Learning Representations (ICLR)*, 2022. https://arxiv.org/abs/2111.02080.

Yiming Xiong, Shengran Hu, and Jeff Clune. Learning to Continually Learn via Meta-learning Agentic Memory Designs. *arXiv preprint arXiv:2602.07755*, 2026. https://arxiv.org/abs/2602.07755.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic Memory for LLM Agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. https://arxiv.org/abs/2502.12110.

Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian Low-rank Adaptation for Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2024a. https://arxiv.org/abs/2308.13111.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026a. https://arxiv.org/abs/2603.19461.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, 2026b. arXiv:2510.04618.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory. *arXiv preprint arXiv:2601.03192*, 2026c. https://arxiv.org/abs/2601.03192.

Xinyu Zhang, Yuxuan Dong, Yanrui Wu, Jiaxing Huang, Chengyou Jia, Basura Fernando, Mike Zheng Shou, Lingling Zhang, and Jun Liu. PhysReason: A Comprehensive Benchmark towards Physics-Based Reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16593–16615, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

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. https://arxiv.org/abs/2601.18734.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models, 2025. arXiv preprint arXiv:2506.10943.

A Additional Related Work

LLM reasoning methods. In-context learning provides language models with the ability to infer a task based on demonstrations, without updating their parameters (Brown et al., 2020), a behavior that has an implicit Bayesian interpretation (Xie et al., 2022). Reasoning-oriented prompting makes intermediate computations available in natural language in the context through scratchpads (Nye et al., 2021) and chain-of-thought (Wei et al., 2022), which enables approaches that decompose problems into simpler subproblems (Zhou et al., 2023), aggregate multiple reasoning paths through self-consistency (Wang et al., 2023), or search over alternative intermediate steps (Yao et al., 2023).

Self-reflection extends this notion by allowing models to critique and revise their own solutions. Self-Refine alternates between generation, feedback, and revision (Madaan et al., 2023), while Reflexion retains verbal feedback in episodic memory for subsequent attempts (Shinn et al., 2023). Related work on metacognition studies whether models can extract generalizable lessons from problem solving (Didolkar et al., 2024), although some evaluations of intrinsic self-correction show that revisions can fail to improve reasoning without external feedback (Huang et al., 2024). These behaviors can also be learned through training, as in STaR's fine-tuning on self-generated rationales that lead to correct answers (Zelikman et al., 2022), training to self-correct (Kumar et al., 2024; Ramji et al., 2025), and Experiential Reinforcement Learning's use of feedback-guided revisions (Shi et al., 2026).

Bayesian modeling for natural language. Bayesian nonparametric models have been used in language tasks to learn latent structures where size is not fixed a priori, ranging from word segmentation (Goldwater et al., 2006) to the reusable grammar subtrees of adaptor grammars (Johnson et al., 2007). The hierarchical Dirichlet process (Teh et al., 2006) is particularly useful in topic modeling, allowing for topics to be shared across documents while having document-specific proportions, while hierarchical Pitman–Yor language models (Teh, 2006) are useful for different-length contexts. Related hierarchical sharing appears in domain adaptation, where Finkel & Manning (2009) link domain-specific model parameters through a shared Gaussian prior, which balances cross-domain guidance with domain-specific observations. For continuous representations, Gaussian LDA (Das et al., 2015) models word embeddings with Gaussian topic distributions with a fixed topic inventory. By contrast, infinite Gaussian mixtures, as introduced in (Rasmussen, 2000), should allow the number of occupied components to grow with the data. Bayesian inference has also been used with modern language models in uncertainty quantification and guided generation. Laplace-LoRA approximates a posterior over adaptation parameters for improved calibration (Yang et al., 2024a), while Hu et al. (2024) use amortized posterior inference for constrained generation and implicit reasoning.

Test-time adaptation and continual learning (continued). SEAL (Zweiger et al., 2025) learns to generate synthetic training data and the updates to adapt the language model (i.e. learning how to adapt itself). In architectures such as Titans (Behrouz et al., 2025b) and ATLAS (Behrouz et al., 2025a), adaptation instead takes place through a neural memory that learns from the incoming context. For agents, AutoHarness (Lou et al., 2026) uses environment feedback to synthesize and refine code that constrains an LLM's actions, while Hyperagents (Zhang et al., 2026a) make both the task-solving program and the procedure for modifying it editable. Harness Continual Learning (Kang et al., 2026) treats prompts, memory, skills, and routing as a persistent, jointly updated state, evaluating candidate changes to improve on the current task while retaining earlier capabilities before deployment.

Memory and retrieval-augmented reasoning with LLMs (continued). Nonparametric retrieval allows for new information to be incorporated with retraining the language model; nearest-neighbor language models can do this by changing the retrieval datastore (Khandelwal et al., 2020). Hierarchical retrieval methods organize information at multiple levels of abstraction: for example, RAPTOR recursively clusters and summarizes document passages into a retrieval tree (Sarthi et al., 2024), while MemTree incrementally builds a hierarchy of semantic summaries as observations arrive (Rezazadeh et al., 2024). Buffer of Thoughts maintains reusable thought templates as procedural memory (Yang et al., 2024b), and Memp distills trajectories into both detailed instructions and higher-level abstractions that can be revised over time with experience (Fang et al., 2026). In MemRL, the utility of stored experiences is learned from feedback, so that retrieval accounts for their usefulness (Zhang et al., 2026c). In ALMA, memory organization is a learned goal, where executable designs specifying what to store, how to retrieve, and how to update (the three key operations we defined in Section 3) are searched over (Xiong et al., 2026).

B BAYESIAN CHEATSHEET ALGORITHM

The memory (modeled by Ω) stores records of behaviors and their embeddings, as well as domain, component, and local group assignments, local and global CRF counts, Gaussian sufficient statistics, and the medoid elements for retrieval.

Algorithm 1 Bayesian Cheatsheet for Test-Time Adaptation (with periodic refitting)

Require: Initial memory Ω_0, NIW prior H, concentrations $\gamma, \{\alpha_d\}$, queries $x_{1:T}$
Require: Retrieval budget k', novelty threshold τ_{new}, refit interval g (or disabled), merge threshold ϵ
Set $\Omega \leftarrow \Omega_0$; compute component predictives, CRF weights, and medoids
for $t = 1, 2, \ldots, T$ **do**
  Embed query as $e(x_t)$ and infer $d_t = \widehat{d}(x_t)$; use the sole domain in cold start
  Set the retrieved behavior set $\mathcal{R}_t \leftarrow \varnothing$
  if memory is nonempty **then**
    Select up to k' highest-weight components via $w_{d_t,k}(e(x_t))$ (Eq. 4)
    Set $\mathcal{R}_t$ to one medoid behavior per selected component
  end if
  Generate the response using $\mathcal{R}_t$, then mine a behavior b_t and compute its embedding e_t
  Set $\rho_{\text{new}}(e_t) = 1$ if memory is empty; otherwise evaluate Eq. 5 on the current memory
  if memory is empty or $\rho_{\text{new}}(e_t) > \tau_{\text{new}}$ **then**
    Create a global component z_t with zero CRF counts; set $\ell_t \leftarrow \text{new}$
    $(\widetilde{N}_{z_t}, \widetilde{S}^{(1)}_{z_t}, \widetilde{S}^{(2)}_{z_t}) \leftarrow (1, e_t, e_t e_t^\top)$
  else
    Cache $w_{d_t,k}(e_t)$ for all existing k using Eq. 4, before changing Ω
    for each k with $w_{d_t,k}(e_t) > 10^{-6}$ **do**
      $(\widetilde{N}_k, \widetilde{S}^{(1)}_k, \widetilde{S}^{(2)}_k) \leftarrow (\widetilde{N}_k, \widetilde{S}^{(1)}_k, \widetilde{S}^{(2)}_k) + w_{d_t,k}(e_t)(1, e_t, e_t e_t^\top)$
    end for
    Set $z_t \leftarrow \arg\max_k w_{d_t,k}(e_t)$ using the cached weights
    Assign weight $n_{d_t,\ell}$ to each local group ℓ in d_t mapped to z_t
    Include a new group with weight $\alpha_{d_t}\widehat{\beta}_{z_t}$; normalize and sample ℓ_t
  end if
  if $\ell_t = \text{new}$ **then**
    Create an empty local group ℓ_t mapped to z_t; increment m_{d_t,z_t} and M_{z_t}
  end if
  Increment integer occupancies n_{d_t,ℓ_t} and n_{d_t,z_t}; store $(b_t, d_t, e_t, z_t, \ell_t)$ once
  if refitting enabled and $t \bmod g = 0$ **then**
    Rebuild Gaussian statistics from all stored hard assignments and refresh NIW predictives
    Run one random-order collapsed Gibbs step with fixed domains (Appendix C)
    Greedily merge pairs with symmetric Gaussian KL $< \epsilon$; update Ω and predictives at each merge
  end if
  Refresh NIW predictives, all CRF mixing weights (Appendix C), and medoids
end for

C COLLAPSED GIBBS SAMPLING

This section expands on the collapsed Gibbs sampler in Section 4.3, which organizes behavior embeddings into shared components while preserving domain-specific use patterns. We first obtain each component's posterior predictive from its pooled sufficient statistics, then describe the CRF assignment updates and the resulting domain-specific retrieval weights.

Given the global sufficient statistics for component k, the posterior over ϕ_k is $\text{NIW}(\mu'_k, \kappa'_k, \Psi'_k, \nu'_k)$:

$$\kappa'_k = \kappa_0 + N_k, \qquad \nu'_k = \nu_0 + N_k,$$
$$\mu'_k = \frac{\kappa_0 \mu_0 + S^{(1)}_k}{\kappa'_k}, \qquad \Psi'_k = \Psi_0 + \left(S^{(2)}_k - N_k \bar{e}_k \bar{e}_k^\top\right) + \frac{\kappa_0 N_k}{\kappa'_k}(\bar{e}_k - \mu_0)(\bar{e}_k - \mu_0)^\top$$

Here, $\bar{e}_k = \frac{S_k^{(1)}}{N_k}$ for $N_k > 0$. Integrating ϕ_k out of the Gaussian likelihood gives a multivariate Student-t posterior predictive:

$$p_k(e) \equiv p(e \mid X_k) = t_{\nu'_k - D+1}\left(e \;\middle|\; \mu'_k, \frac{\kappa'_k + 1}{\kappa'_k(\nu'_k - D + 1)}\Psi'_k\right) \tag{10}$$

For an empty component, the sufficient statistics are zero, so the posterior parameters reduce to $(\mu_0, \kappa_0, \Psi_0, \nu_0)$. Then, integrating over this NIW prior yields the prior predictive, which we denote $p_0(e) = p(e \mid H)$.

When resampling observation (d, i), we use a two-level assignment/partition structure based on Teh et al. (2006). We maintain an auxiliary partition of the behavior observations within each domain d into local groups, each of which is mapped to a global component, of which there are K total. Multiple groups may map to the same component, inducing component-sharing across domains. Let $\ell_{d,i}$ denote the local-group assignment for sample (d, i), so $n_{d,\ell}^{-i}$ is the count of behaviors already in group ℓ excluding (d, i), k_ℓ is the component that group ℓ maps to, and $M_k^{-i} = \sum_{d' \in \mathcal{D}} m_{d',k}^{-i}$ is the total number of groups across all domains mapped to component k, excluding (d, i).

A first categorical draw is done to assign $\ell_{d,i}$ over the set of existing local groups as well as a potential new group.

$$p(\ell_{d,i} = \ell \mid \cdot) \propto \begin{cases} n_{d,\ell}^{-i}\, p_{k_\ell}^{-(d,i)}(e_{d,i}), & \ell \text{ is an existing local group} \\ \frac{\alpha_d}{\gamma + \sum_{j=1}^K M_j^{-i}} \left(\sum_{k=1}^K M_k^{-i} p_k^{-(d,i)}(e_{d,i}) + \gamma p_0(e_{d,i})\right), & \ell = \text{new} \end{cases} \tag{11}$$

Marginalizing over local group choice gives the equivalent component-level assignment score, for $\widehat{\beta}_k^{-i} = \frac{M_k^{-i}}{\gamma + \sum_{j=1}^K M_j^{-i}}$:

$$p(z_{d,i} = k \mid \cdot) \propto \left(n_{d,k}^{-i} + \alpha_d \widehat{\beta}_k^{-i}\right) p_k^{-(d,i)}(e_{d,i}),$$

If a new group is spawned, we next sample the component this new group maps to, setting $z_{d,i}$:

$$p(z_{d,i} = k \mid \ell_{d,i} = \text{new}, \cdot) \propto \begin{cases} M_k^{-i}\, p_k^{-(d,i)}(e_{d,i}), & k = 1, \ldots, K \text{ (existing)} \\ \gamma\, p_0(e_{d,i}), & k = \text{new} \end{cases} \tag{12}$$

The mixing weights used in retrieval are estimates derived from the group-to-component mapping counts, which are posterior means under the partition structure:

$$\widehat{\beta}_k = \frac{M_k}{\gamma + \sum_{j=1}^K M_j}, \qquad \widehat{\beta}_{\text{new}} = \frac{\gamma}{\gamma + \sum_{j=1}^K M_j}$$

$$\widehat{\pi}_{d,k} = \frac{n_{d,k} + \alpha_d \widehat{\beta}_k}{\alpha_d + \sum_{j=1}^K n_{d,j}}, \qquad \widehat{\pi}_{d,\text{new}} = \frac{\alpha_d \widehat{\beta}_{\text{new}}}{\alpha_d + \sum_{j=1}^K n_{d,j}}$$

D Implementation and Reproducibility

D.1 Behavior Mining Procedure and Corpus

We construct a separate behavior corpus for each language model, Qwen3-32B and Granite-4.2-30B, using the same model for behavior mining and evaluation (i.e. the behaviors are on-policy for the respective model). We randomly sampled 20,000 mathematics prompts from NuminaMath-CoT (Li et al., 2024) and 20,000 physics prompts from Camel Physics (Li et al., 2023).

Mining follows a three-stage metacognitive procedure, inspired by Didolkar et al. (2025). First, the model solves each problem by generating a chain-of-thought solution. It then reflects on the solution, assessing its correctness and identifying missing or reusable reasoning behaviors. Finally, it

extracts these behaviors as structured {name: description} records. Extraction is performed regardless of whether the generated answer is correct or not. The extraction prompt instructs the model to replace problem-specific numerical values with symbolic placeholders, so that the descriptions are generalizable and transferable strategies are expressed.

After extraction is performed over the full prompt set, we deduplicate behavior records based on exact match of their labels (the `name` field). The resulting Qwen corpus contains 57,955 mathematics behaviors and 75,324 physics behaviors, while the Granite corpus contains 26,305 mathematics behaviors and 15,839 physics behaviors; the difference reflects this deduplication step. In warm-started comparisons for a given language model, all memory-based methods use the same model-specific corpus, for standardized comparison.

We embed behavior descriptions using `granite-embedding-english-r2` (Awasthy et al., 2025) and fit a separate PCA transform on each model's mined corpus, reducing the embeddings to $D = 128$ dimensions. Each of these PCA transforms is held fixed during fitting of the mixture as well as test-time adaptation.

D.2 Mixture Hyperparameters

We fit a separate HDP-GMM for each model's corpus using the configuration detailed in Table 3, using the Gibbs sampling procedure described in Section 4.3 and Appendix C. The resulting fitted mixtures contain 64 components for Qwen and 62 for Granite, without setting any fixed component numbers a priori. The prior scale Ψ_0 is set relative to the empirical covariance of the projected behavior embeddings. The main results use explicit domain conditioning (described in Section 4.4), where the projected query is assigned to its nearest domain centroid, and retrieval combines the mixture weights for that domain with the shared component predictive densities. In both the static and the online settings (Tables 4 and 1), we use the top-$k' = 5$ components based on normalized weights.

Table 3: Hyperparameters for HDP-GMM fitting and retrieval, as well as the refit window. The refit settings only apply to the settings where periodic refitting is used.

Parameter	Value
Global concentration γ	5.0
Domain concentration α_d	1.0
NIW scale Ψ_0	$1.2\times$ empirical covariance in PCA space
NIW degrees of freedom ν_0	$D + 2 = 130$
NIW mean strength κ_0	0.01
Projection dimension D	128
Offline Gibbs steps / burn-in	100 / 50
Novelty threshold τ_{new}	0.5
Symmetric KL merge threshold ϵ	0.1
Refit interval g	10 (AIME'25, PhysReason-mini); 50 (Omni-MATH, PhysReason-full)
Retrieval (static and online)	Top-$k' = 5$
Domain conditioning	Nearest-centroid classifier (explicit)

D.3 Learning the Query Projector (f_θ)

The learned query projector f_θ maps the problem's embedding into the (PCA-transformed) space in which the behavior mixture was fitted. We use a multilayer perceptron with two hidden layers (`width`=[512, 256]) and a 128-dimensional output. The training data of (query, behavior) pairs comes from the mining procedure in Appendix D.1; for a pair (x, b), let $q(x) = \mathcal{E}(x)$ be the raw query embedding and let $u_b = \mathrm{PCA}(\mathcal{E}(b))$ be the projected behavior embedding.

Because downstream retrieval evaluates a Student-t posterior predictive density under the fitted mixture, which depends on the scale of $f_\theta(q(x))$ relative to each component, we train f_θ with a mean squared error objective over the training pairs $\mathcal{T}$ to train its output to match the coordinates and scale

of the behavior embeddings:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{(x,b)\in\mathcal{T}} \|f_\theta(q(x)) - u_b\|_2^2 .$$

We learn such a projector for the respective language models, so one each for Qwen and Granite, ensuring that projected queries and the stored behavior embeddings are expressed in a shared space. Our training is performed with AdamW for 10 epochs with a batch size of 256, learning rate of 1×10^{-4}, and weight decay of 10^{-4}. The resulting projector f_θ, which is just $0.56M$ parameters, is held fixed during evaluation, with minimal overhead.

D.4 Cold-Start Setting and Evaluation

In the cold-start setting, we no longer provide the language model with access to offline-mined behaviors, and we do not have f_θ. This means that we directly operate in the raw embedding dimension (768), but since Gibbs steps occur over a smaller corpus (fewer behaviors are mined over the test set compared to the offline mining setting), this remains computationally feasible. Since there is no empirical covariance to work from or PCA performed, we set $\Psi_0 = I_{768}$, and $\nu_0 = D+2 = 770$. Domain-conditioning does not arise since we treat the samples in a test set as belonging to a single domain, so this is analogous to the non-hierarchical construction.

For evaluation (in both the cold-start setting as well as with the offline-mined memory), all models are given a thinking budget of 64k tokens, and generations are sampled with temperature 0.7. For both settings, memory is reset following each evaluation run – for the cold-start setting, this ensures every benchmark is run without memory, while for the offline-mined memory setting, we revert back to the "pre-trained" mixture fit on the offline behavior corpus.

E Complete Baseline Results

We include the full static-retrieval results against the baselines in Table 4, complementing the results with Qwen3-32B in Figure 2. We find that with Granite-4.2-30B, Bayesian Cheatsheet outperforms the baselines again across the benchmarks, with the BC-centroids variant performing comparably to BC-PP, our main method using the posterior predictive; BC-centroids outperforms BC-PP on AIME'25, while BC-PP is better on Omni-MATH, and they are tied on PhysReason.

Table 4: Static-retrieval results using the mined behavior corpus for Qwen3-32B and Granite-4.2-30B. OM and PR denote Omni-MATH and PhysReason, respectively. Bold and underlined values indicate the best and second-best results within each model and benchmark.

	Qwen3-32B			Granite-4.2-30B		
Method	AIME'25	OM	PR	AIME'25	OM	PR
Base model	71.1	18.1	16.4	73.3	24.8	16.4
Random behavior selection	72.2	18.9	18.6	73.3	27.5	16.4
Direct behavior retrieval	72.2	18.6	23.5	74.4	28.2	19.5
Metacognitive Reuse, $k' = 5$	<u>75.6</u>	17.8	21.7	77.8	28.4	<u>22.6</u>
BC-centroids (ours)	74.4	<u>20.5</u>	<u>24.8</u>	**81.1**	<u>29.1</u>	**23.9**
BC-PP (ours)	**77.8**	**22.3**	**27.0**	<u>80.0</u>	**31.6**	**23.9**

F Ablations

The ablations below use the Qwen3-32B language model, and consider different parts of our inference and online update procedure.

F.1 Top-k' Retrieval

In Table 5, we perform a sweep over the number of retrieved components (k') used in static retrieval with BC-PP (corresponding to Table 4). Since we draw the medoid element from each component,

k' is also the number of behaviors added to the prompt for metacognitive reasoning. We hold all other hyperparameters fixed. We adopt $k' = 5$, which generally performs the best across the board. Notably, relative to the baselines in Table 4, BC-PP with $k' = 3$ would also outperform every baseline across the three benchmarks for Qwen3-32B.

Table 5: Effect of the retrieval size k' (number of components) on static BC-PP accuracy.

k'	AIME'25	Omni-MATH	PhysReason
1	72.2	19.5	24.8
2	76.7	20.2	24.8
3	76.7	22.5	25.2
5	77.8	22.3	27.0

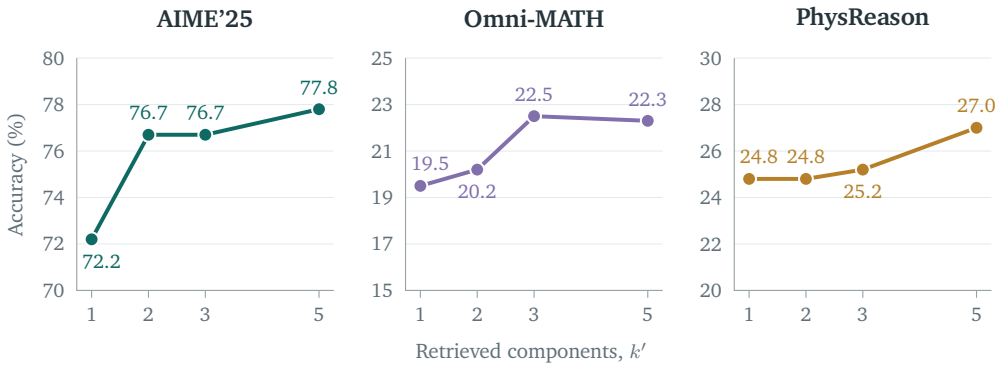


Figure 4: Static BC-PP accuracy versus retrieval size k' for Qwen3-32B.

F.2 Refit Interval Sweep

For the online setting with refitting, we sweep the refit interval on Omni-MATH ($n = 916$), while holding $k' = 5$ and the novelty threshold as well as the inference recipe (including domain conditioning) constant, and report the performance. We explore a wide range of values including $g = 0$ (no refits, but new components may still be spawned) for comparison.

There are two ways that the component count can increase: 1) a new component is spawned through the novelty-driven trigger specified in Section 4.4, and 2) a new component is formed during collapsed Gibbs resampling (refit step), so a new component effectively "splits" out of an existing one. In Table 6, we report both the total number of new components compared to the offline-fit state of the HDP-GMM ($K = 64$ for Qwen3-32B) as well as the subset of *novelty-induced* new components.

Table 6: Effect of the refit interval g on online Qwen3-32B accuracy on Omni-MATH. K is the final component count (initial is 64), "net increase" is the net change in the number of components over the run, and "novelty" denotes the subset of those new components that were novelty-triggered.

g	Accuracy	K (final)	Net increase	Novelty-triggered
0	26.4	76	12	12
5	24.8	94	30	5
10	25.2	86	22	4
25	25.2	85	21	4
50	**26.6**	85	21	11
75	26.4	85	21	14
100	25.3	79	15	11
250	25.9	73	9	7
500	25.9	75	11	10

We observe that when refits are more aggressively performed (at small values of g), the proportion of novelty-triggered new components to the total new components is much smaller. This suggests that

most new components emerge due to the stochastic nature of the collapsed Gibbs sampling procedure, which is performed more often. As the refit interval increases, the majority of the new components are due to novelty identified over the test set pass. Based on having the highest performance and a near-even balance of novelty-triggered components and Gibbs-induced splits, we take $g = 50$ for the results with refitting in Tables 1 and 2. We note that other values of g also perform comparably: in fact, every value of g tested here would at least match or outperform the strongest performing baseline for Omni-MATH in Table 1. The $g = 10$ value is used for AIME'25 and PhysReason-mini since a single pass over the test set only contains 30 and 31 samples, respectively.

G Qualitative Examples

G.1 Component Meta-Labels

To analyze the components generated by fitting the HDP-GMM and the structure it induces (in terms of common thematic areas), we study whether generalizable meta-labels can be produced for each component. Specifically, we utilize Claude Opus 5 to produce a label given the list of behaviors assigned to a given component: the results for Qwen (Table 7) and Granite (Table 8) are below.

Table 7: Short thematic labels for each of the $K = 64$ HDP-GMM components fit on the Qwen3-32B mined behavior corpus.

#	Domain	Label
0	Math	Unit consistency & dimensional-analysis checks
1	Physics	Multi-method cross-checks & literature verification
2	Math	Solution verification & domain-restriction checks
3	Physics	Cosmology & dark-matter physics
4	Math	Symmetry-based algebraic simplification
5	Physics	Applied materials & optics engineering formulas
6	Physics	Fluid dynamics & flow-stability analysis
7	Math	Algebraic equation-solving techniques
8	Physics	Particle-physics conservation laws & decay formulas
9	Physics	Empirical calibration & uncertainty propagation
10	Math	Algebraic simplification & symbolic manipulation
11	Math	Multi-step problem decomposition & dependency tracking
12	Math	Polynomial/rational-expression factoring & proportionality
13	Physics	Dynamical-systems stability & boundary-case verification
14	Math	Euclidean geometry theorems & area formulas
15	Physics	Nuclear astrophysics & stellar nucleosynthesis
16	Physics	Condensed-matter magnetism & superconductivity
17	Math	Coordinate systems, symmetry & 3D rotations
18	Physics	Conservation laws & flow/capacity analysis
19	Math	Graph-theoretic counting & distribution symmetry
20	Physics	Estimation, decomposition & bounding heuristics
21	Math	Calculus optimization: extrema & discriminants
22	Physics	Magnetostatics & magnetic-field formulas
23	Physics	Peer-review citation & experimental-design practices
24	Math	Financial & percentage word problems
25	Math	Unit verification & function-symmetry identities
26	Physics	Cross-disciplinary tradeoff & Bayesian reasoning

(continued from previous page)

#	Domain	Label
27	Physics	General problem-solving verification checks
28	Physics	Astrophysics: supernovae, dark matter & black holes
29	Physics	Fluid/thermodynamic physics & dimensional consistency
30	Physics	Topological & condensed-matter quantum systems
31	Physics	General relativity, black holes & cosmology
32	Physics	Nuclear/particle physics & authoritative-source verification
33	Math	Combinatorics: subsets, Burnside's lemma & pigeonhole
34	Math	Coordinate geometry: lines, conics & perpendicularity
35	Math	Percentage/ratio word problems
36	Physics	Quantum information & computation protocols
37	Physics	Physics heuristics & fluid/thermodynamic formulas
38	Physics	Materials science: nanoparticles & phonon scattering
39	Math	Combinatorial game theory & optimal strategy
40	Math	Geometry: circles, loci & optical formulas
41	Physics	Problem-solving meta-strategies & memorized constants
42	Math	Arithmetic: place value, fractions & percentages
43	Physics	Cross-domain dimensional analysis (physics/chemistry)
44	Physics	Special relativity: time dilation & Lorentz transforms
45	Physics	Biophysics & fluid/cell mechanics
46	Physics	Condensed matter: band structure & Hall effect
47	Physics	Thermodynamics: phase changes & adiabatic processes
48	Math	Logic & proof techniques
49	Math	Geometry/trigonometry fundamentals (circles, hyperbolas)
50	Physics	Optics: lenses, mirrors & convection
51	Physics	Atomic physics: quantum numbers & spectra
52	Physics	Biophysics/biochemistry: membranes & virial coefficients
53	Math	Probability: independence & conditional rules
54	Math	Number theory: modular arithmetic & divisibility
55	Math	Calculus: derivatives, optimization & monotonicity
56	Physics	Symbolic problem-solving with placeholder variables
57	Physics	Solid-state/thermal physics & perturbation theory
58	Math	Number theory: primality, divisors & CRT
59	Physics	Astrophysics: gravitational waves & black holes
60	Math	Algebra: Vieta's formulas & symmetric polynomials
61	Math	Statistics: central tendency & distributions
62	Physics	Theoretical physics: QFT & string theory
63	Math	Integration techniques & trigonometric identities

Table 8: Short thematic labels for each of the $K = 62$ components in the HDP-GMM memory module fit from the Granite-4.2-30B-mined behavior corpus.

#	Domain	Label
0	Physics	Chaos theory & dynamical-systems criteria
1	Math	Trigonometric identities in optics & triangle geometry
2	Math	Combinatorics: permutations & circular arrangements
3	Physics	Condensed-matter defect & superconductor classification
4	Physics	Energy-efficiency metric definition protocols
5	Math	Combinatorial game theory & parity arguments
6	Physics	Cryptographic protocol layering & classification
7	Physics	Radiation dosimetry & hazard classification
8	Physics	Quantum field theory formalism
9	Math	Solid geometry: coordinates, area & volume
10	Math	Rectangle/area word problems
11	Physics	Biomechanics & sensory-perception force models
12	Math	Circle/conic geometry: tangents & distances
13	Physics	Atomic transitions & qubit-state definitions
14	Physics	Nanostructure electronic transport & quantum Hall effect
15	Math	Solution-verification protocols & boundary conditions
16	Physics	Astrophysics: gravitational-wave & stellar-remnant scaling
17	Math	Age/time word problems & exponential decay
18	Math	Probability: equally-likely outcomes & GF(2) spaces
19	Physics	Magnetostatics: Biot–Savart law & synchrotron radiation
20	Math	Relative motion & relativistic velocity addition
21	Physics	Ion-channel biophysics & membrane potentials
22	Math	Arithmetic sequences & periodic scheduling
23	Physics	Dimensional consistency in thermo/fluid dynamics
24	Math	Graph theory: degree sums & polyhedra
25	Math	Combinatorial geometry: line arrangements & gears
26	Math	Function symmetry: odd/even parity arguments
27	Math	Number-theory & arithmetic word problems
28	Math	Descriptive statistics: mode, median & dice probability
29	Math	Volume/geometry word problems
30	Math	Financial word problems: sequential transactions & tax
31	Math	Algebra: matrices, determinants & polynomial roots
32	Math	Algorithmic complexity & search-strategy heuristics
33	Math	Balance-scale, BMI & series-convergence identities
34	Math	Statistical estimation & election vote-counting
35	Physics	Vector sign conventions in optics & waves
36	Math	Chemistry stoichiometry & mixture conservation
37	Math	Animal/leg counting & ratio-substitution algebra
38	Math	Calculus optimization & blackbody/standard-candle checks
39	Physics	Traffic-flow control & capacity-constrained systems
40	Physics	Condensed-matter & particle-physics regime classification
41	Physics	Topological defects & anyon/monopole physics

(continued from previous page)

#	Domain	Label
42	Physics	Diffusion & optical-fiber transport formulas
43	Physics	Space-weather & astrobiology assessment protocols
44	Physics	Nuclear physics & technology-readiness evidence
45	Physics	Dark matter, gravity & black-hole bounds
46	Physics	Materials characterization & attenuation formulas
47	Physics	Wave/vibration frequency analysis
48	Math	Olympiad geometry/algebra lemmas & proof techniques
49	Math	Grid/number-theory formulas & metric geometry
50	Math	Geometric length & area word problems
51	Math	State-tracking & cyclic-structure combinatorics
52	Physics	Dark-matter & gravitational-wave detector-signature protocols
53	Physics	Particle-physics decay & resonance formulas
54	Physics	System boundary & scope-definition protocols
55	Math	Logic puzzles & multi-component reasoning
56	Physics	Thermodynamics & propulsion/fission-yield formulas
57	Physics	Domain-agnostic evaluation & classification protocols
58	Physics	Circuit & electrostatics formulas
59	Physics	Biophysics & semiconductor measurement protocols
60	Math	Kinematics & geometric-probability word problems
61	Math	Logic puzzles & linguistic disambiguation

G.2 Example Behaviors (Component Medoids)

To illustrate the notion of a behavior obtained through the language model's self-reflection, we provide a list of a few behaviors – specifically, the medoid elements of some of the Qwen3-32B components described in Table 7. We observe that the behaviors, while providing guidance for solution strategies, are still fairly general, which facilitates effective reuse at inference time.

Table 9: Selected representative behaviors from a subset of the $K = 64$ HDP-GMM components fit from the Qwen3-32B-mined behavior corpus (component numbers match those in Table 7)

#	Domain	Representative behavior
0	Math	Always check and convert units to ensure consistency before performing calculations, using known conversion factors if necessary.
4	Math	When solving equations, check for symmetry in variables (e.g., if swapping x and y yields the same equation) to simplify or identify solutions.
8	Physics	Use symmetry to simplify field contributions: opposite sides of a loop may cancel if their contributions are equal and opposite, but verify directions to avoid assuming cancellation.
14	Math	When solving for area, cross-check the result with perimeter calculations (if relevant) to ensure logical consistency.
18	Physics	Use multiple independent sources or methods to confirm the value of critical parameters (e.g., particle masses) before finalizing a solution.
30	Physics	When analyzing perturbations, always check if they break a symmetry that protects the system's properties.

(continued from previous page)

#	Domain	Representative behavior
37	Physics	When solving for critical points (e.g., minima/maxima) in equations, use known physical constraints (e.g., SOFAR channel depth) to simplify calculations and avoid unphysical solutions.
39	Math	In strategic games, determine if a player has a forced win by identifying whether the opponent can always block or avoid the winning condition regardless of moves.
41	Physics	When solving a problem, outline the key concepts and their relevance before providing detailed analysis to ensure clarity and completeness.
43	Physics	Use a domain-specific checklist of experimental techniques (e.g., NMR, neutron scattering) to verify that no relevant methods are omitted when solving problems involving material property analysis.
49	Math	When solving geometric problems, always consider whether the figure is a special case (e.g., square, rhombus, rectangle), as this can simplify calculations.
52	Physics	Consider material properties (e.g., a, b) when solving applied problems to ensure practical feasibility.