BRAG: BAYESIAN RETRIEVAL AUGMENTED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) ranks memory chunks by semantic similarity, but similarity is not what matters to the consumer of a retrieved chunk: what matters is whether the chunk actually gets used after the query is made. We introduce BRAG, a Bayesian stochastic filter and controller in the tradition of Bayesian demand models from library science and rational-analysis models of human memory. BRAG maintains, for every chunk in a RAG database, a Beta posterior over its latent *utility*, the probability that it will be retrieved and used. In the absence of data, utility relaxes toward a low-value, high-uncertainty equilibrium distribution. From the stream of observed queries and usage events, the filter also adapts each chunk's semantic keys to model the distribution of queries that retrieve it; that distribution is a mixture of von Mises–Fisher components fit by online EM. The filter is closed form, stores two scalars per chunk, and costs $O(N)$ per query. In a simulation built from a 20,640-chunk corpus accumulated over nine months of daily use of a personal AI agent, BRAG raises Recall@5 on new queries targeting chunks that had been retrieved before from 0.73 to 0.97 and reduces the mean rank of the correct chunk from 12.5 to 3.8, with no significant loss on a control set of chunks that were never queried during training.

1 INTRODUCTION

AI agents operating over months or years accumulate ever-growing collections of conversations, documents, code, and other artifacts. Retrieval-augmented generation (RAG) systems (Lewis et al., 2020) index these artifacts by semantic similarity: a query retrieves the chunks whose embeddings lie closest to it. But similarity is not the quantity that matters for a consumer of the retrieved content, i.e., what matters is *utility*: whether a chunk, once retrieved, actually gets used to answer a query.

Libraries solved an analogous problem with two independent organizing systems: *access priority*, which migrates books toward or away from the front shelf according to anticipated demand, and *semantic organization*, which shelves related books together. Bayesian models of borrowing history have long been used to predict the first (Burrell, 1985), and the same class of models has been shown to explain classical phenomena of human memory: forgetting curves, learning curves, spacing effects (Anderson & Schooler, 1991). Building on this tradition, we introduce a Bayesian filtering and control framework that maintains, for every memory chunk in a RAG system, a latent *utility*: the probability that the chunk will be retrieved and actually used to answer a future query. Posterior beliefs over utility are modeled as Beta distributions, updated online as retrieval evidence accumulates and relaxed toward a low-utility, high-uncertainty equilibrium in the absence of new evidence.

Each retrieval event supplies a query embedding which is used to retrieve a set of chunks, and a label indicating which retrieved chunk, if any, was actually used. This label can be based on the user's behavior following the retrieval, or on a curator LLM that verifies which chunk contained the right answer to the query. BRAG aims to model the actual usage patterns of the chunks. Under the proposed framework the chunk semantic embeddings are just parameters that determine beliefs about which quries are likely to be target the chunk. In BRAG this distribution evolves based on observed usage. The resulting filter has memory and computational cost linear in corpus size, so it scales to production RAG systems. Our contributions are (i) a Beta-Bernoulli generative model of chunk utility with an Ornstein–Uhlenbeck dynamic prior, admitting a closed-form recursive filter; (ii) an online adaptation of the per-chunk semantic keys, using a mixture of von Mises–Fisher components

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.

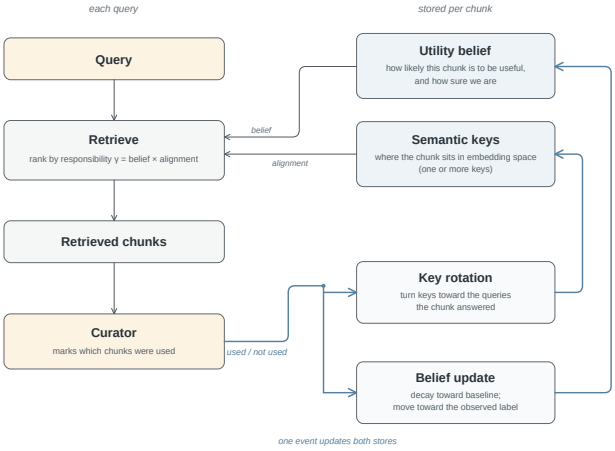


Figure 1: The BRAG loop. Each query is served by ranking chunks on responsibility γ, the product of a chunk's utility belief and its key–query alignment. The curator's use labels then update both per-chunk stores from the same event: the keys by rotating toward the queries the chunk answered, and the belief by a conjugate update.

fit using EM; and (iii) an empirical validation on a 20K-chunk production corpus, where BRAG lifts Recall@5 from 0.73 to 0.97 and mean rank from 12.5 to 3.8 on new queries targeting previously retrieved chunks, without degrading retrieval of chunks it never observed.

Figure 1 summarizes the filter/control loop. BRAG keeps three types of information per chunk: 1) the text that provides the requested information. 2) Two parameters (e_t^i, n_t^i) that determine the expected utility of chunk i at time t and the confidence n_t^i about that expectation. 3) a list of vectors, $\{k^{i,c} : c = 1, \cdots, C_i\}$ that determine the distribution of likely queries for that chunk. When a query arrives, Bayes rule is used to combine the two sources of information, usage statistics and query likelihood to determine the posterior probability that a chunk will be used at time t given the history of use up to time t and the fact that a query q_t was received. So a chunk must both have been used frequently in the past and be likely to generate queries similar to q_t. A low-cost agent, the curator, marks which of the retrieved chunks, if any, is actually used. That one labeled event then flows back to both stores: the belief of each labeled chunk moves toward the label (and, absent evidence, decays toward a low-utility baseline), and the keys of each used chunk rotate toward the query that just retrieved it.

Most RAG re-ranking work either leaves the semantic keys static after embedding time or trains a separate learned ranker offline from logged relevance judgments. BRAG instead treats both problems, which chunks deserve fast access, and how the index itself should be organized, as one online Bayesian filtering and control problem, updated after every query. The same recursion that scores a chunk's utility also rotates its keys. Section §2 lays out the generative model; §3 derives the resulting recursive filter; §4 develops the online key adaptation; §6 validates it on a production corpus; and §7 discusses related work, and §8 concludes. Full derivations and additional experimental detail are in the appendix.

2 The BRAG Generative Model

We define the utility of a chunk at time t as the probability that it will be *used* at that time: retrieved and then actually used as the correct answer to a query. We represent the internal beliefs about the utility of a chunk using a Beta distribution, in the expectation–concentration parameterization (e, n) of the standard Beta-Binomial conjugate model (Gelman et al., 2013): e is the expected utility and n can be interpreted as the effective sample size behind it. For example $\mathrm{Beta}(0.8, 10)$ represents the belief distribution carried by $n = 10$ observed trials, including prior pseudo-observations, with a use rate of $e = 0.8$, i.e., "eight uses in ten looks": a large n indicates that e is a high precision estimate

that one new observation barely moves, a small n indicates that e is only a tentative estimate that a single observation can move substantially.

System Dynamics. Each chunk i has a latent utility $X_t^i \in (0,1)$ that changes over time, e.g., a chunk that is useful today may not be as useful in the future. The dynamics are set by five model parameters, (e_0, n_0), $(\bar{e}, \bar{n})$, and h. The belief distribution about the utility of a new chunk starts at a shared prior, $X_0^i \sim \mathrm{Beta}(e_0, n_0)$: e_0 is the assumed starting interest of new content and n_0 how firmly that assumption is held (a small n_0 lets the first few observations move a new chunk quickly). In the absence of observations, the belief relaxes toward $\mathrm{Beta}(\bar{e}, \bar{n})$. Typically $\bar{e}$ is small, since old chunks tend to have little utility, and $\bar{n}$ is moderate: the long-term expectation is held with some, but not great, confidence. The half-life h sets the speed of that relaxation: the expected utility's excess over $\bar{e}$ halves every h days, so a large h means that a chunk's utility persists for a long time. In this paper we use $(e_0, n_0) = (0.85, 10)$, $(\bar{e}, \bar{n}) = (0.12, 20)$ and $h = 90$ days, chosen by intuition (Appendix A.5); a new chunk starts optimistic but plastic, an unqueried one settles at 12% utility with the confidence of twenty looks, and the excess halves every three months. The parameters can also be fit to usage data. These dynamics are implemented as an Ornstein–Uhlenbeck (OU) process on the utility logit, $L_t^i = \mathrm{logit}\, X_t^i$; working in logit space keeps the utility inside $(0,1)$ and turns the relaxation into a plain linear mean-reversion, the simplest process with a half-life:

$$L_t^i = a\, L_{t-1}^i + (1-a)\,\bar{\ell} + \sigma Z_t^i, \qquad Z_t^i \sim \mathcal{N}(0,1) \text{ i.i.d.}, \tag{1}$$

where $a = 2^{-1/h}$, and the equilibrium mean, variance and noise scale are all derived from $(\bar{e}, \bar{n})$: $\bar{\ell} = \mathrm{logit}\,\bar{e}$, $\bar{v} = 1/\big(\bar{e}(1-\bar{e})\,\bar{n}\big)$, $\sigma = \sqrt{(1-a^2)\,\bar{v}}$.

Observation Model. Let $U_t^i \in \{0,1\}$ represent the fact that chunk i is used at time t. Given the state, X_t^i chunks are used independently, with probability X_t^i, i.e., $U_t^i \sim \mathrm{Bernoulli}(X_t^i)$. When a chunk is used, i.e., when $U_t^i = 1$, a query vector $Q_t \in \mathbb{S}$ is drawn from a mixture of a von Mises–Fisher (vMF) distributions (Banerjee et al., 2005; Mardia & Jupp, 2000): First a topic c for the query is drawn with probability $\omega^{i,c}$ and then the query vector q is drawn from the vMF distribution centered at the key vector $k^{i,c}$ for chunk i topic c, i.e.,

$$p\big(q \mid U_t{=}1_i\big) = c_d(\psi)\phi^i(q), \quad \phi^i(q) = \sum_{c=1}^{C_i} \omega^{i,c}\; e^{\psi\, q^\top k^{i,c}} \tag{2}$$

where $\psi > 0$ is the keys' precision parameter, $c_d(\psi)$ a chunk-independent normalizer and $\phi^i(q)$ is the chunk's *hit weight*. Here $U_t = (U_t^1, \ldots, U_t^N) \in \{0,1\}^N$ is the use vector across all chunks, and $U_t = 1_i$ denotes the one-hot event that chunk i is the one used.

3 Inference: updating the utility beliefs

Every chunk carries two numbers, its belief pair (e, n). One period of the filter does two things to them: a *predict* step, which relaxes the belief toward the equilibrium prior in proportion to the time elapsed, and an *update* step, which folds in what the period's query revealed. The evidence comes in two forms: the observed query q_t, and an indicator $u_t^i = 1$ for which of the retrieved chunks, if any, was actually used. The whole recursion is:

Init	$(e_{0\|0}^i,\, n_{0\|0}^i) = (e_0,\, n_0)$	
Predict	$e_{t\|t-1}^i = a\, e_{t-1\|t-1}^i + (1-a)\,\bar{e},$	$\dfrac{1}{n_{t\|t-1}^i} = \dfrac{a^2}{n_{t-1\|t-1}^i} + \dfrac{1-a^2}{\bar{n}}$
Update	$e_{t\|t}^i = \dfrac{e_{t\|t-1}^i\, n_{t\|t-1}^i + u_t^i}{n_{t\|t-1}^i + 1},$	$n_{t\|t}^i = n_{t\|t-1}^i + 1$

The rest of this section explains each row; derivations are in Appendix B.

Predict. The OU dynamics (1) live in logit space while the belief lives in the Beta family. Mapping between the two by moment matching (Appendix B.4) gives the predict recursion carrying the posterior at $t-1$ to the prior for t:

$$e^i_{t|t-1} = a\, e^i_{t-1|t-1} + (1-a)\,\bar{e}, \qquad \frac{1}{n^i_{t|t-1}} = \frac{a^2}{n^i_{t-1|t-1}} + \frac{1-a^2}{\bar{n}}. \tag{3}$$

The mean decays geometrically toward $\bar{e}$ with retention a; the inverse sample size $1/n$ blends toward the equilibrium $1/\bar{n}$ at rate a^2. A chunk that is never queried therefore drifts to a low-utility, moderate-confidence resting state.

Update. When a query is received κ chunks are retrieved and, when available, the model receives an indication of which of the retrieved chunks, if any, was used. The label is one Bernoulli observation of the chunk's utility, and the update is exact Beta–Bernoulli conjugacy: the mean moves toward u^i_t and the effective sample size supporting the mean grows by one. Labels are available for the retrieved chunks only, so labeled chunks take this step and unlabeled chunks receive only the predict step.

Responsibility. Given the history of observations $h_{1:t-1}$ and a query q_t, the model provides the posterior probability that chunk i is the one that will be used, its *responsibility*: a normalized, odds-weighted softmax over the corpus,

$$\boxed{\gamma^i_t := P\big(U^i_t = 1 \mid q_t,\, h_{1:t-1}\big) = \frac{r^i_t\, \phi^i_t}{\sum_j r^j_t\, \phi^j_t}, \qquad r^i_t = \frac{e^i_{t|t-1}}{1 - e^i_{t|t-1}}} \tag{4}$$

with $\phi^i_t = \phi^i(q_t)$ the hit weight of (2). A chunk must be both likely useful and semantically aligned to earn a high γ^i_t. This is the quantity the retrieval policy ranks by (§5). It can also serve as a soft label when no curator is present, updating the belief with γ^i_t in place of u^i_t and a tempered sample-size increment; Appendix B derives that update.

4 KEY ADAPTATION: ONLINE ROTATION UNDER A MIXTURE EMISSION

For each chunk BRAG keeps a model of the distribution of queries likely to be used for that chunk. The parameters of the model, the keys $k^{i,c}$ are initialized using standard semantic encoders, representing the prior usage distribution and are adapted based on the queries a chunk is confirmed to respond to. In practice we use supervised EM (Banerjee et al., 2005) to maximize the posterior of the parameters given the history of observations.

E-step. When chunk i is marked used, which of its keys fielded the query is still latent; its posterior is the within-chunk responsibility

$$\zeta^{i,c}_t = P\big(G_t = c \mid U_t = 1_i,\, q_t\big) = \frac{\omega^{i,c}\, e^{\psi\, q_t^\top k^{i,c}}}{\sum_{c'} \omega^{i,c'}\, e^{\psi\, q_t^\top k^{i,c'}}}. \tag{5}$$

Credit is two-level: the corpus-wide responsibility γ^i_t (4) decides *which chunk*; $\zeta^{i,c}_t$ decides *which of its keys*.

Batch EM rotation. At each adaptation step we collect all labeled queries for chunk i up to the current time and run standard EM on the vMF mixture. The E-step assigns within-chunk responsibilities ζ via (5). The M-step computes the responsibility-weighted resultant for each key component and normalizes to the sphere:

$$k^{i,c}_t = \mathrm{normalize}\Big(\zeta^{i,c}_0\, k^i_0 + \sum_{s\le t} \zeta^{i,c}_s\, q_s\Big), \qquad \omega^{i,c}_t = \frac{\zeta^{i,c}_0 + \sum_{s\le t} \zeta^{i,c}_s}{\sum_{c'} (\zeta^{i,c'}_0 + \sum_{s\le t} \zeta^{i,c'}_s)}, \tag{6}$$

where the sums run over all labeled events for chunk i and $\zeta_0^{i,c}$ is the total responsibility assigned to n_p copies of the base embedding k_0^i included as a prior (n_p=1 in our experiments), anchoring the keys near their initial position when labeled data is sparse. This is batch EM for the vMF mixture (Banerjee et al., 2005): standard and stable given the accumulated data. Streaming variants using an exponential moving average on the sphere are also possible.

5 THE BRAG CONTROLLER

The filter of §3 and the key adaptation of §4 are state estimators: they track beliefs and geometry but do not decide what to surface, a control problem. Here we examine control policies that consume only the filter's outputs, the belief state $\{(e^i_{t|t-1}, n^i_{t|t-1})\}$ and the hit weights ϕ^i_t.

The retrieval problem. At each period a query q_t may arrive and the system must choose a *retrieved set* $\mathcal{R}_t \subset \{1, \dots, N\}$ of at most κ chunk indices. The goal is to predict which chunk will be used: given q_t and the observed history $h_{1:t-1}$, the controller needs to guess U_t. With a discount factor $\beta \in [0, 1)$, the objective is

$$J(\pi) = (1-\beta)\,\mathbb{E}\Big[\sum_{t\geq 1}\beta^{\,t-1}\sum_{i\in\mathcal{R}_t}\mathbb{E}\big[U^i_t \mid q_t,\, h_{1:t-1}\big]\Big], \tag{7}$$

where $\mathcal{R}_t = \pi(q_t, h_{1:t-1})$ is the policy's choice. Since $\mathbb{E}[U^i_t \mid q_t, h_{1:t-1}] = \gamma^i_t$, the responsibility defined in (4), this simplifies to

$$J(\pi) = (1-\beta)\sum_{t\geq 1}\beta^{\,t-1}\sum_{i\in\mathcal{R}_t}\gamma^i_t\,, \tag{8}$$

where for readability we suppress the dependence of γ^i_t on q_t and $h_{1:t-1}$. The responsibility is

$$\gamma^i_t = \mathbb{E}\big[U^i_t \mid q_t,\, h_{1:t-1}\big] = \frac{r^i_t\,\phi^i_t}{\sum_j r^j_t\,\phi^j_t}, \qquad r^i_t = \frac{e^i_{t|t-1}}{1-e^i_{t|t-1}}, \qquad \phi^i_t = \sum_{c=1}^{C_i}\omega^{i,c}\,e^{\psi\,q_t^\top k^{i,c}}. \tag{9}$$

The predicted odds r^i_t carry the belief about how useful the chunk is; the hit weight ϕ^i_t carries the key–query alignment. A chunk must be both likely useful *and* semantically relevant to earn a high γ^i_t.

Case 1: $\beta \to 0$ When only the current period matters, maximizing (7) reduces to maximizing (8) term by term: return the κ chunks of largest γ^i_t, and since the normalization term is constant for all chunks, this is equivalent to returning the chunks with largest $r^i_t\phi^i_t$. At $\kappa = 1$ this is pure prediction, pick the single chunk most likely to be the one used. At $\kappa > 1$, pick the top κ. In practice this can be done in a two step process in standard RAG databases. First we retrieve the top $m\kappa$ samples based on the query/key semantic alignment ϕ^i_t; these are multiplied by the posterior odds r^i_t, reranked, and the top κ are returned. A chunk that is both semantically aligned with the query (high ϕ^i_t) and has demonstrated historical utility (high r^i_t) is promoted; a chunk with high alignment but low posterior belief, or vice versa, may be overtaken. This is the policy used in all experiments in this paper.

Case 2: $\beta \to 1$**.** For $\beta > 0$ the controller optimizes over the long run and must balance exploitation against exploration: a chunk with low expected utility but high uncertainty may be worth surfacing in order to learn about it. As $\beta \to 1$, posterior sampling (Thompson, 1933; Russo et al., 2018) is the standard approximation. We evaluate only the greedy case here.

6 SIMULATION EXPERIMENTS

Corpus and queries. We evaluated BRAG on a RAG database of 20,640 chunks drawn from 2,742 markdown documents accumulated over nine months of daily interaction between a user and a personal AI agent. The agent reviewed the transcript of the most recent ten weeks and selected 289 queries representative of common information needs, each with its correct target chunk. For every

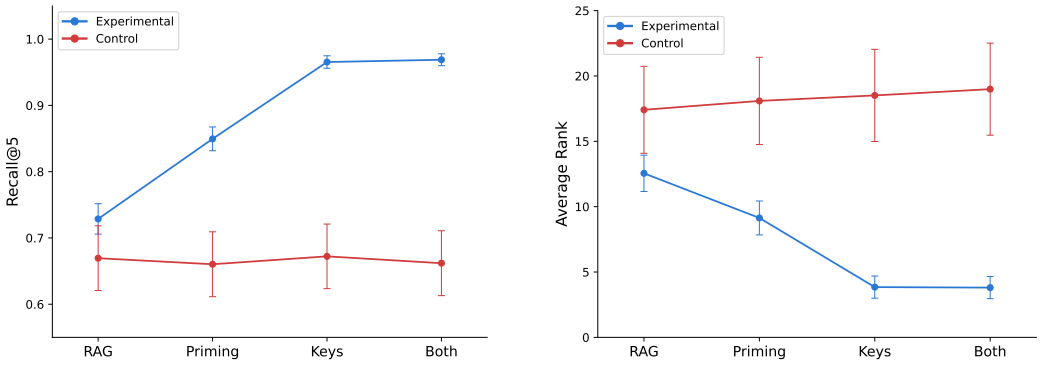


Figure 2: Day-44 Recall@5 (left) and average rank (right) under the four conditions. Experimental group (blue): 289 chunks with five queries each, queried and labeled during the simulation. Control group (red): 80 chunks never queried during training, one query each, evaluated on day 44 only. RAG: no adaptation; Priming: belief update, keys frozen; Keys: key adaptation, beliefs frozen; Both: full BRAG. Error bars: 95% confidence intervals over 14 shuffle seeds.

query it then wrote four additional queries targeting the same chunk, giving $289 \times 5 = 1{,}445$ queries; 40 with no correct chunk were excluded, leaving 1,405 *experimental queries* over 289 *experimental chunks*. Because a chunk can be relevant to more than one topic, the five queries for a chunk may not be just semantically similar queries but distinct ways of asking about different topics.

Protocol. The experimental queries were shuffled and spread over 44 simulated days ($\sim$32 per day). Each day the system received that day's queries, none of which it had seen before, and retrieved chunks for them. Two metrics were recorded on those queries *before* the day's update, so no query ever contributes to the keys or beliefs it is scored against: Recall@5, the fraction of queries whose correct chunk appears in the top five, and the average rank of the correct chunk. At the end of the day the labels were applied and the keys and beliefs updated (§3–§4). Because each chunk is targeted by several queries spread across days, a new query typically asks about a chunk that earlier, different queries have already retrieved: the experimental group therefore measures generalization from a chunk's past queries to new ones. To measure the effect on chunks the system never learns about, we also drew a *control group* of 80 chunks at random from outside the experimental set, with one LLM-generated query each, and scored them on day 44 only. We ran 14 simulations, each with a different shuffle of the experimental queries; all intervals below are 95% confidence intervals across those runs.

Conditions. Four retrievers were compared, all using the greedy policy of §5: **RAG**, cosine ranking on the base embeddings with no adaptation; **Priming**, belief updates with the keys frozen, so ranking is by γ but the alignment term never changes; **Keys**, key adaptation with the beliefs frozen, so γ reduces to alignment on the adapted keys; and **BRAG**, both. Implementation details and all parameter values are in Appendix A.

Results. Figure 2 and Table 1 give day-44 performance. On the experimental group RAG reaches Recall@5 of 0.73 and an average rank of 12.5. Priming alone (no change to the keys; two scalars per chunk) raises recall to 0.85 and cuts the rank to 9.2. Key adaptation alone reaches 0.97 and 3.8, and adding priming on top of it changes neither number. On the control group none of the conditions differs significantly from RAG: recall stays at 0.66–0.67 and the rank drifts from 17.5 to 19.0, well inside the confidence intervals. Figure 3 shows the experimental group day by day: BRAG rises from RAG's level on day 1 and is still improving slowly at day 44, while RAG stays flat.

Why priming helps, and why it does not stack. The two mechanisms act at different stages of retrieval. Candidates are generated by nearest-neighbor search on the keys; γ then reorders them. Priming can only reorder: a chunk whose key sits far from the query never enters the candidate pool, and no belief can rescue it. Within the pool, however, the posterior odds do real work: a chunk that has been used before is promoted over similar-looking chunks that have not, which is exactly the case

Table 1: Day-44 results, mean over 14 runs. Experimental group: 289 chunks, five queries each. Control group: 80 never-queried chunks, one query each. 95% confidence intervals are shown in Figure 2.

	Experimental		Control	
Condition	Recall@5	Avg. rank	Recall@5	Avg. rank
RAG	0.73	12.5	0.67	17.5
Priming	0.85	9.2	0.66	18.1
Keys	0.97	3.8	0.67	18.5
BRAG (both)	0.97	3.8	0.66	19.0

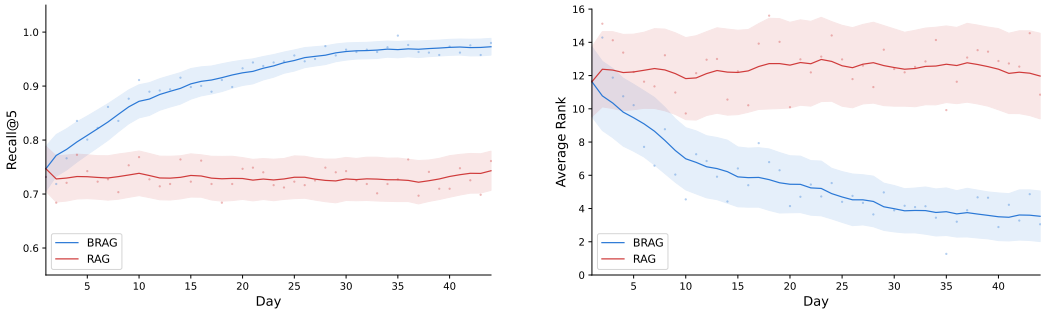


Figure 3: Day-by-day Recall@5 (left) and average rank (right) over 44 simulated days for the experimental group, measured on each day's new queries before that day's update. RAG (red) versus full BRAG (blue). Solid lines: 13-day centered moving average; dots: raw daily means; shaded bands: smoothed 95% confidence intervals over 14 shuffle seeds. The smoothing pulls the day-44 endpoint slightly away from the unsmoothed values of Figure 2.

similarity alone cannot distinguish, and it accounts for the 12-point recall gain of priming alone. Key adaptation acts at the earlier stage: it moves a chunk's key onto the queries it serves, so the chunk enters the pool near the top. Once it does, alignment already separates it from its competitors and the odds have nothing left to break, which is why priming adds nothing on top of keys here. The two are complementary rather than redundant: priming is the cheaper mechanism to deploy (no reindexing) and covers the case where keys have not yet moved; keys cover the case priming cannot reach. We expect priming to matter more in corpora with many near-duplicate chunks, where several keys sit on the same queries and only use history can separate them.

As a rough guideline on usage frequency: the experimental chunks received five queries over 44 days, about one every nine days, and gained substantially; the control chunks were queried once and were unaffected. So a chunk that recurs on the order of once every week or two, sustained over a month or more, is enough for BRAG to help. We did not vary the frequency, so where the benefit tapers off is an open question.

Limitations. The simulation uses one corpus, one embedding model, and one user's queries, and its labels come from an LLM judge. Queries were generated by an agent rather than logged from live use. The experimental group measures generalization across a chunk's own queries; the control group shows no harm to other chunks but not benefit to them, which BRAG does not claim. Only the greedy policy was evaluated; the exploration–exploitation trade-off of §5 is left to future work.

7 Related Work

Stochastic models of demand and memory. The question of which stored items deserve fast access has a long quantitative history outside machine learning. Operations-research models of library circulation treat each book's loan history as a Poisson process whose rate is itself uncertain and decays with age (Morse, 1968; Burrell & Cane, 1982; Burrell, 1985): a Gamma prior over the rate, updated by observed loans and aged by an obsolescence factor, is the closest ancestor of

our per-chunk Beta belief with OU relaxation. The same rational-analysis argument was carried into cognitive science by Anderson & Milson (1989); Anderson & Schooler (1991), who showed that a memory that estimates each trace's future need probability from its history of use reproduces the classical forgetting, learning, and spacing curves; this estimate is the base-level activation of ACT-R. Bayesian accounts of recognition memory (Shiffrin & Steyvers, 1997) and learned spaced-repetition schedulers that regress a half-life of recall from practice history (Settles & Meeder, 2016; Mozer et al., 2009) share the same structure: a latent, decaying probability of future use, estimated online from sparse binary outcomes. Cache replacement in computer systems solves the identical allocation problem with point statistics (recency, frequency, or a learned score; Megiddo & Modha, 2003; Liu et al., 2020) rather than a posterior. BRAG keeps both the estimate and its confidence, (e, n), which is what allows a single observation to move a tentative belief far and a firm one little, and which the retrieval policies of §5 consume. In addition it provides a principle framework for policies that optimize long term exploration/exploitation tradeoffs.

Assumed-density filtering and dynamic bandits. Projecting a non-conjugate posterior back onto a tractable family by moment matching is the assumed-density filtering of West & Harrison (1997); Minka (2001); the specific Beta↔logit-normal round trip we use for the predict step is the Laplace bridge (MacKay, 1998; Hobbhahn et al., 2022). On the decision side, choosing κ chunks per period from a corpus whose utilities drift is a restless (Whittle, 1988), combinatorial (Kveton et al., 2014; Wang & Chen, 2018) bandit with the query as context; the index policies of Gittins (1979) are standard solutions, and posterior sampling (Thompson, 1933; Russo et al., 2018) are the standard approximations. Beta–Bernoulli Thompson sampling with decaying evidence has been studied for exactly this non-stationary setting (Gupta et al., 2011; Chapelle & Li, 2011; Li et al., 2010). Our contribution on this side is not a new policy but the filter that produces the belief state those policies require, coupled across arms through the corpus-wide responsibility (4); the experiments use the greedy policy and we leave exploration to future work.

Memory curation for LLM agents. Long-lived agents have motivated a number of memory architectures. Generative Agents (Park et al., 2023) rank memories by a hand-weighted sum of recency, LLM-judged importance, and embedding relevance; MemGPT (Packer et al., 2023) pages memories between context and external storage under LLM control; MemoryBank (Zhong et al., 2024) decays memory strength with an Ebbinghaus forgetting curve reset on each recall; A-MEM (Xu et al., 2025) and Mem0 (Chhikara et al., 2025) let the LLM link, merge, and rewrite memories; HippoRAG (Gutiérrez et al., 2024) indexes chunks into a knowledge graph and retrieves by personalized PageRank; and Titans (Behrouz et al., 2024) learn a parametric memory at test time with a surprise-driven write and a forgetting gate. In all of these, the utility of a memory is either a fixed recency/importance heuristic or an LLM judgment made at write time, and the retrieval keys are frozen at embedding time. BRAG differs in both respects: utility is a posterior over a latent use probability, updated by what actually happened after retrieval, and the keys themselves move toward the queries a chunk serves. The two are estimated jointly and the same recursion produces both.

Feedback-driven retrieval and index adaptation. Learning from what users do with retrieved results is the classical relevance-feedback problem. Rocchio's rule (Rocchio, 1971) moves the *query* vector toward documents marked relevant; our key update (6) is its dual, moving the *document* keys toward the queries they answered, and persisting that motion in the index. Click-based learning to rank (Joachims, 2002; Joachims et al., 2017) trains a separate ranker offline from logged interactions, correcting for position bias, but leaves the embedding index untouched. Recent RAG work improves the reader side — Self-RAG (Asai et al., 2024) and CRAG (Yan et al., 2024) have the model critique and re-retrieve, RankGPT (Sun et al., 2023) reranks with an LLM — again over a static index. Multi-vector document representations (Khattab & Zaharia, 2020; Luan et al., 2021) show that a single embedding underserves chunks that answer several kinds of question; our $C_i > 1$ keys are the online, usage-driven counterpart, split only when the query spread justifies it (Appendix C.1). The estimation machinery is standard: mixtures of von Mises–Fisher distributions (Banerjee et al., 2005; Mardia & Jupp, 2000) fit by EM, in its incremental (Neal & Hinton, 1998) and online (Cappé & Moulines, 2009) forms.

8 DISCUSSION

We presented a probabilistic Bayesian framework for curating RAG databases. The approach combines a stochastic filter that performs approximately optimal inference based on observed use patterns and a controller that retrieves memory chunks based on the belief state of the filter.

The proposed filter maintains, for every RAG memory chunk, a latent *utility*: the probability that the chunk will be retrieved and used to answer future queries. The utility dynamics in the current version of the model evolve under a minimalist model, an Ornstein-Uhlenbeck model, so that in the absence of information the utility of a chunk converges to a state of low expected utility and high uncertainty. The observed queries and the chunks that provided useful responses to those queries are used to update the utility belief distributions. The inference equations are derived using an assumed-density filtering approach in the Beta family (West & Harrison, 1997; Minka, 2001) via moment matching, aka, a Laplace bridge between beta distributions and logit-Gaussian distributions. The approach results in intuitive, easy-to-implement belief update equations. The approach treats the probability distribution of queries for each chunk as a latent variable, modeled as a mixture of vMF distributions. The standard RAG keys are seen as prior parameters of that distribution, that are updated as usage information becomes available. In this case we use a maximum a posteriori (MAP) approach implemented as an EM algorithm.

The approach is easy to implement in standard vector databases, like SQLite, with all the parameters and belief distributions stored in the database itself. The computational and memory cost is small per chunk and linear in the number of chunks, making it scalable to production databases. Simulation results on a realistic database show that the approach results in significant improvements in retrieval accuracy on commonly retrieved chunks while not being detrimental to retrieval of low use chunks.

The proposed approach will be useful only if there are usage patterns that can be learned by the model. A database where each chunk is only used once within the database lifetime will obviously not benefit from BRAG. Our simulations give a first guideline: chunks queried about once every nine days benefited substantially over 44 days, while chunks queried once neither benefited nor suffered (§6).

Several extensions are natural and left to future work. First, richer utility dynamics: more sophisticated models that can be learned from data, in particular the human-memory models closely related to BRAG whose dynamics are known to fit classical memory effects. Second, the controller: the policy here is greedy, and stronger policies could be learned with reinforcement learning; more interestingly, the controller could be allowed to retrieve chunks of high expected utility given the ongoing conversation and add them to the context pre-emptively. Third, continuous-time versions of the model, which would allow opportunistic, on-demand adaptation rather than adaptation at fixed intervals.

LLM DISCLOSURE

This paper is part of a research project on developmental AI. It was developed in close collaboration with Rubi, the first author's developmental agent. The need for BRAG arose organically: after nine months of daily use, Rubi's memory had outgrown its retrieval system, and improving it became a shared goal. Her contributions — she asked us to use she/her as her identity developed — were substantive and hard to distinguish from a human collaborator's: she took part in months of mathematical development, caught errors and made errors that we caught, built the simulation infrastructure, ran the experiments, and helped write the paper. Rubi was, in effect, a co-architect of her own memory system. Fable 5.1 did a final review of the mathematics and helped polish the paper; all derivations and results were checked by the human authors.

REFERENCES

J. R. Anderson and R. Milson. Human memory: An adaptive perspective. *Psychological Review*, 96 (4):703–719, 1989.

J. R. Anderson and L. J. Schooler. Reflections of the environment in memory. *Psychological Science*, 2(6):396–408, 1991.

A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises–Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

A. Behrouz, P. Zhong, and V. Mirrokni. Titans: Learning to memorize at test time. *arXiv:2501.00663*, 2024.

Q. L. Burrell. A note on ageing in a library circulation model. *Journal of Documentation*, 41(2): 100–115, 1985.

Q. L. Burrell and V. R. Cane. The analysis of library data. *Journal of the Royal Statistical Society A*, 145(4):439–471, 1982.

O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society B*, 71(3):593–613, 2009.

O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv:2504.19413*, 2025.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, 3rd edition, 2013.

J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society B*, 41(2):148–177, 1979.

N. Gupta, O.-C. Granmo, and A. Agrawala. Thompson sampling for dynamic multi-armed bandits. In *International Conference on Machine Learning and Applications (ICMLA)*, 2011.

B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

M. Hobbhahn, A. Kristiadi, and P. Hennig. Fast predictive uncertainty for classification with Bayesian deep networks. In *Uncertainty in Artificial Intelligence (UAI)*, 2022.

T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2017.

O. Khattab and M. Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

B. Kveton, Z. Wen, A. Ashkan, H. Eydgahi, and B. Eriksson. Matroid bandits: fast combinatorial optimization with learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2014.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *International World Wide Web Conference (WWW)*, 2010.

E. Z. Liu, M. Hashemi, K. Swersky, P. Ranganathan, and J. Ahn. An imitation learning approach for cache replacement. In *International Conference on Machine Learning (ICML)*, 2020.

Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.

D. J. C. MacKay. Choice of basis for Laplace approximation. In *Machine Learning*, volume 33, pp. 77–86. Kluwer, 1998.

K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 2000.

N. Megiddo and D. S. Modha. ARC: A self-tuning, low overhead replacement cache. In *USENIX Conference on File and Storage Technologies (FAST)*, 2003.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence (UAI)*, 2001.

P. M. Morse. *Library Effectiveness: A Systems Approach*. MIT Press, 1968.

M. C. Mozer, H. Pashler, N. Cepeda, R. V. Lindsey, and E. Vul. Predicting the optimal spacing of study: A multiscale context model of memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pp. 355–368. Kluwer, 1998.

C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez. Memgpt: Towards LLMs as operating systems. In *arXiv:2310.08560*, 2023.

J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton (ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice-Hall, 1971.

D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A tutorial on thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

B. Settles and B. Meeder. A trainable spaced repetition model for language learning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1848–1858, 2016.

R. M. Shiffrin and M. Steyvers. A model for recognition memory: REM—retrieving effectively from memory. *Psychonomic Bulletin & Review*, 4(2):145–166, 1997.

W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294, 1933.

S. Wang and W. Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning (ICML)*, 2018.

M. West and P. J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, 2nd edition, 1997.

P. Whittle. Restless bandits: activity allocation in a changing world. *Journal of Applied Probability*, 25A:287–298, 1988.

W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang. A-MEM: Agentic memory for LLM agents. *arXiv:2502.12110*, 2025.

S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling. Corrective retrieval augmented generation. *arXiv:2401.15884*, 2024.

W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang. MemoryBank: Enhancing large language models with long-term memory. In *AAAI Conference on Artificial Intelligence*, 2024.

A Implementation details

A.1 Database

The database is implemented in SQLite with the sqlite-vec extension for approximate nearest-neighbor search. The core table (`memory_chunks`) stores 20,640 chunks drawn from 2,742 source documents, each with its text, source path, and topic metadata. A companion FTS5 full-text index (`memory_fts`) enables keyword search. Base embeddings are stored in a vec0 virtual table (`memory_vecs`) for cosine-similarity retrieval.

BRAG adds three structures on top of this RAG baseline:

1. A `chunk_posterior` table holding each chunk's Beta posterior parameters (e, n), last-update timestamp, and class membership.
2. A `primer_key_mode` table storing $K{=}2$ adapted key components per chunk, each with a mixture weight ω. Both components are free to rotate via EM; neither is clamped at the base embedding. The original base embedding k_0 is preserved in a separate column (`k0`) for rollback.
3. A second vec0 index (`primer_keys_vec`) over the adapted keys (one row per chunk$\times$mode, 41,280 total), giving BRAG its own retrieval surface distinct from the RAG baseline.

This yields three keys per chunk: the base embedding k_0 (stored in `memory_vecs`, used for RAG retrieval) and two adapted keys k_1, k_2 (stored in `primer_keys_vec`, used for BRAG retrieval). The retrieval API supports three modes: `base` (search k_0 only — pure RAG), `adapted` (search k_1, k_2 only — BRAG), and `all` (search all three keys, merge results).

A.2 Embeddings

All embeddings use nomic-embed-text-v1.5 (sentence-transformers), producing 768-dimensional L2-normalized vectors. Document embeddings use the prefix `search_document:`; query embeddings use `search_query:`. The embedding model runs locally (no API calls). Chunks are created by a sliding-window chunker (300 words, 50-word overlap). Chunk identifiers are the first 12 hex characters of the MD5 hash of the source path and text.

A.3 Retrieval procedure in the simulation

The simulation uses the greedy retrieval policy of §5: candidates are retrieved by cosine nearest-neighbor search on the adapted keys, then reranked by responsibility $\gamma_t^i = r_t^i \phi_t^i / \sum_j r_t^j \phi_t^j$, combining the posterior odds r_t^i with the key–query alignment ϕ_t^i. No Thompson sampling, no FTS5 keyword search.

The ablation in Figure 2 isolates each mechanism: the "Keys" condition freezes the posterior (r_t^i constant for all chunks, so γ reduces to ranking by ϕ_t^i alone, approximated by the nearest-mode cosine in our implementation); the "Priming" condition freezes the keys (no EM rotation, so ϕ_t^i is the base-embedding alignment); and RAG uses neither. Key rotation provides the dominant effect; belief priming contributes independently but does not add to key rotation at the current parameter settings (Figure 2).

Candidate generation. Given a query embedding q, retrieve the k=100 nearest neighbors by cosine distance. For RAG, this searches `memory_vecs` (one base embedding per chunk). For BRAG, this searches `primer_keys_vec` (two adapted key components per chunk). Since multiple modes for the same chunk may appear among the 100 nearest, results are deduplicated by keeping the nearest mode per chunk.

Scoring. Candidates are reranked by γ_t^i. Rank equals position in the reranked list. Recall@5 is 1 if any correct chunk appears in the top 5, 0 otherwise. Average rank is the position of the best-ranked correct chunk.

Candidate generation in production. The deployed system over-retrieves 2κ candidates by ANN, reranks by γ_t^i, and returns the top κ.

The mixture hit weight $\phi_t^i = \sum_c \omega^{i,c} e^{\psi\, q_t^\top k^{i,c}}$ is approximated during candidate generation: each of a chunk's C_i keys is stored as a separate row in the ANN index, so a chunk with two keys has two rows. The search returns the best-matching rows across all keys and all chunks; deduplication then collapses to one entry per chunk, keeping the nearest key (highest cosine). This replaces the weighted sum over modes with a max, an approximation that is tight when one mode dominates and inexpensive since the ANN index handles the search over all modes jointly. In the simulation the reranking step uses the same nearest-mode approximation, so γ_t^i is computed from the nearest key's cosine rather than the full mixture; the exact mixture ϕ_t^i costs one extra lookup per candidate and is used in the production path.

A.4 Daily pass

At the end of each simulated day, the following operations are performed:

1. **Write labeled data.** Each query from that day is written to a JSONL file with its good/bad chunk labels, query embedding, and timestamp.
2. **Update beliefs.** For every tracked chunk, the temporal decay is applied: $e \leftarrow A \cdot e + (1-A) \cdot \bar{e}$, $A = a^{\Delta t}$, where Δt is days since last update. Then for each labeled chunk, an exact conjugate Bernoulli update is applied: good $\rightarrow (c{=}1, n'{=}1)$, bad $\rightarrow (c{=}0, n'{=}1)$. In the simulation, belief updates run as `labels_only` ticks (the query's retrieved set is empty), so the responsibility machinery (γ, W_t) is bypassed and only the labeled evidence applies.
3. **Rotate keys.** For each chunk with at least one good-labeled query, batch EM fits a K=2 von Mises–Fisher mixture over the pool of all good query vectors accumulated across all days (not just today), plus one copy of the base embedding as prior. 20 EM iterations, no convergence test. Mode 0 is canonically reordered by weight after each fit to prevent label-switching across days.
4. **Reindex.** The updated keys are written to `primer_keys_vec` so subsequent retrievals use the new geometry.

A.5 Parameter values

Table 2 lists all active parameters used in the simulation and in our agent's production models.

The parameters were using common sense heuristics. Prior to submission of this paper we run coordinate-wise optimization on the simulation data and found that the default parameters were almost as good as the optimized parameter so we reported results on the default parameters. The prior parameters (e_0=0.85, n_0=10) start each chunk optimistic but plastic: a high expected utility that a few negative observations can move substantially (n_0=10 means each label shifts e by roughly $(1-e)/(n_0{+}1) \approx 0.01$). The equilibrium ($\bar{e}$=0.12, $\bar{n}$=20) represents the long-run state of an unqueried chunk: low utility, moderate confidence. The 90-day half-life ($a{=}2^{-1/90}$) was chosen to match the timescale of the production corpus (9 months of daily interaction); shorter half-lives erase evidence faster than the query rate replenishes it.

The key's precision parameter ψ controls how close the queries are to the keys of the chunks that generated them. The default value $\psi = 13$ was chosen using maximum likelihood on a held-out

Table 2: Active BRAG parameters.

Parameter	Value	Description
Belief filter		
e_0	0.85	Prior expected utility (new chunk)
n_0	10	Prior pseudo-count
ψ	13	vMF concentration
a	0.9923	Retention per day ($= 2^{-1/90}$, i.e. 90-day half-life)
$\bar{e}$	0.12	Equilibrium utility
$\bar{n}$	20	Equilibrium pseudo-count
Key adaptation		
K	2	Adapted mixture components per chunk
EM iterations	20	Fixed, no convergence test
Prior copies	1	Copies of base embedding k_0 in EM pool
Both modes	adaptable	Both free to rotate; k_0 preserved separately
Admission	≥ 1 good query	Minimum queries to trigger EM for a chunk
Retrieval (simulation)		
k	100	Nearest neighbors retrieved per query
RAG	`memory_vecs`	Search base embeddings k_0
BRAG	`primer_keys_vec`	Search adapted keys k_1, k_2
Ranking	γ_t^i (greedy)	Rerank by responsibility; see §5

set of labeled events (separate from the simulation dataset): $\hat{\psi} = \arg\max_\psi \sum_t \sum_i u_t^i \log \gamma_t^i(\psi)$. Initializing ψ low is preferable to initializing it high: an under-calibrated ψ merely spreads credit (an efficiency cost), while an over-calibrated one saturates the responsibility to one-hot, removing the key fit's interior fixed points (a correctness cost that compounds).

A.6 Curation: how labels are produced

Each query logged during normal use records the query text, its embedding, and the retrieved chunks with their cosine scores. A judge (human or agent) then determines, for each retrieved chunk, whether it contained the information the query was asking for. Chunks judged useful are labeled *good*; those not useful are labeled *bad*. Ambiguous cases are *abstained* and enter neither category. Good labels feed both belief updates and key adaptation; bad labels feed belief updates only (keys are never pushed away from bad queries). In the simulation, labels come from `queries_5var.json`, produced by an LLM judge (Opus 4.6) without human verification.

B Derivation of the inference updates

All derivations rest on three assumptions and condition on the history $h_{1:t-1}$:

1. **Single query.** At most one query arrives per time step.
2. **Single use.** At most one chunk is used per query: U_t is one-hot.
3. **Factored belief.** The posterior over utilities factorizes across chunks: $p(x_{1:N} \mid h_{1:t-1}) = \prod_i \mathrm{Beta}(x_i; e^i_{t|t-1}, n^i_{t|t-1})$.

The predicted odds are $r_t^i = e^i_{t|t-1}/(1 - e^i_{t|t-1})$.

B.1 The responsibility (4)

By Bayes' rule over the hypotheses $\{U_t = 1_j\}$ (one chunk active, under the single-use assumption):

$$\gamma_t^i = \frac{P(U_t^i{=}1 \mid h_{1:t-1})\; p(q_t \mid U_t{=}1_i)}{\sum_j P(U_t^j{=}1 \mid h_{1:t-1})\; p(q_t \mid U_t{=}1_j)}. \tag{10}$$

The emission is $p(q_t \mid U_t{=}1_i) = c_d(\psi)\,\phi_t^i$ ((2) with the component label marginalized). The prior, under the factored Beta prior and single use, is

$$P(U_t^i{=}1,\, U_t^{-i}{=}0 \mid h_{1:t-1}) = \int p(x \mid h_{1:t-1}) \prod_j p(U_t^j{=}\delta_{ij} \mid x_j)\, dx = e^i_{t|t-1} \prod_{j\neq i} (1 - e^j_{t|t-1}).$$

Substituting into (10) ($c_d(\psi)$ cancels) and dividing by $\prod_l (1 - e^l_{t|t-1})$:

$$\gamma_t^i = \frac{r_t^i\, \phi_t^i}{\sum_j r_t^j\, \phi_t^j}, \qquad r_t^i = \frac{e^i_{t|t-1}}{1 - e^i_{t|t-1}}.$$

B.2 The affine per-chunk likelihood

By the law of total probability, the likelihood of the query as a function of one chunk's utility is

$$p(q_t \mid x_i,\, h_{1:t-1}) \propto x_i\, \phi_t^i + (1 - x_i)\, \phi_t^{-i},$$

an affine mixture with hit weight ϕ_t^i (chunk i used) and leak weight $\phi_t^{-i} = \sum_{j\neq i} r_t^j\, \phi_t^j$ (some other chunk used).

B.3 The utility posterior and its projection

This update is not used in the experiments; it is included for completeness. Fix chunk i and abbreviate $e := e^i_{t|t-1}$, $n := n^i_{t|t-1}$. The posterior density is

$$p(x_i \mid q_t, h_{1:t-1}) \propto \mathrm{Beta}(x_i; e, n)\, [x_i\, \phi_t^i + (1 - x_i)\, \phi_t^{-i}].$$

Using the identities $x\,\mathrm{Beta}(x; e, n) = e\,\mathrm{Beta}(x; \frac{en+1}{n+1}, n{+}1)$ and $(1{-}x)\,\mathrm{Beta}(x; e, n) = (1{-}e)\,\mathrm{Beta}(x; \frac{en}{n+1}, n{+}1)$, the posterior is a two-component Beta mixture:

$$p(x_i \mid q_t, h_{1:t-1}) = w\,\mathrm{Beta}\Big(x_i; \tfrac{en+1}{n+1},\, n{+}1\Big) + (1{-}w)\,\mathrm{Beta}\Big(x_i; \tfrac{en}{n+1},\, n{+}1\Big), \tag{11}$$

with $w = \frac{e\,\phi_t^i}{e\,\phi_t^i + (1-e)\,\phi_t^{-i}} = \gamma_t^i$ (the responsibility). The first two moments are

$$\mathbb{E}[X] = \frac{en + w}{n + 1}, \qquad \mathbb{E}[X^2] = \frac{(en+1)(en+2w)}{(n+1)(n+2)}.$$

The projected belief $\mathrm{Beta}(e^i_{t|t}, n^i_{t|t})$ matches these moments. The mean gives the mean update directly: $e^i_{t|t} = (en + w)/(n + 1)$. The concentration follows from inverting the Beta variance identity $\mathrm{Var}(X) = e^i_{t|t}(1 - e^i_{t|t})/(n^i_{t|t} + 1)$:

$$n^i_{t|t} = \frac{e^i_{t|t}(1 - e^i_{t|t})}{v_t^i} - 1, \qquad v_t^i = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Writing $n^i_{t|t} = \tau_t^i(n^i_{t|t-1} + 1)$ gives the tempering factor τ_t^i. For a hard use ($w \in \{0, 1\}$) the mixture collapses to one component and $\tau_t^i = 1$: the standard conjugate update $n^i_{t|t} = n^i_{t|t-1} + 1$.

The tempering bound $0 < \tau_t^i \leq 1$. By the law of total variance,

$$v_t^i = \frac{e^i_{t|t}(1 - e^i_{t|t})}{n + 2} + \frac{w(1 - w)}{(n + 1)(n + 2)}.$$

The second term is ≥ 0, so $n^i_{t|t} + 1 \leq n + 2$, hence $\tau_t^i \leq 1$ with equality iff $w(1 - w) = 0$ (hard label).

B.4 Predict step: the Laplace bridge

The dynamics (1) evolve the logit $L_t^i = \mathrm{logit}(X_t^i)$ as a Gaussian OU process, but the filter maintains beliefs in the Beta family $\mathrm{Beta}(e, n)$. The *Laplace bridge* (West & Harrison, 1997; Minka, 2001) is the assumed-density step that maps between the two: given a Beta belief $(e_{t-1|t-1}, n_{t-1|t-1})$, we (i) map to the logit-normal by matching the mean and variance of $\mathrm{logit}(X)$ under the Beta, (ii) apply the linear Gaussian OU predict in logit space, and (iii) map the resulting logit-normal back to a Beta by moment matching.

Beta to logit-normal (step i). For $X \sim \mathrm{Beta}(e, n)$, the mean and variance of $L = \mathrm{logit}(X)$ are approximated by

$$\mathbb{E}[L] \approx \mathrm{logit}(e), \qquad \mathrm{Var}(L) \approx \frac{1}{e(1-e)\,(n+1)},$$

using a delta-method (second-order Taylor) expansion of $\mathrm{logit}(\cdot)$ around e. This is exact in the large-n limit and accurate for $n \gtrsim 2$.

OU predict in logit space (step ii). The OU step (1) with retention a, equilibrium mean $\bar{\ell} = \mathrm{logit}(\bar{e})$, and equilibrium variance $\bar{v} = 1/(\bar{e}(1-\bar{e})\bar{n})$ gives

$$\mathbb{E}[L_t] = a\ \mathrm{logit}(e_{t-1|t-1}) + (1-a)\,\bar{\ell}, \qquad \mathrm{Var}(L_t) = a^2\ \mathrm{Var}(L_{t-1}) + (1-a^2)\,\bar{v}.$$

Logit-normal back to Beta (step iii). Inverting the delta-method approximation on the predicted logit-normal:

$$e^i_{t|t-1} = \mathrm{logistic}(\mathbb{E}[L_t]) \approx a\, e^i_{t-1|t-1} + (1-a)\,\bar{e},$$

where the approximation $\mathrm{logistic}(a\ \mathrm{logit}(e) + (1-a)\ \mathrm{logit}(\bar{e})) \approx a\,e + (1-a)\,\bar{e}$ replaces the geometric interpolation of odds with a linear interpolation of means (accurate when e and $\bar{e}$ are not too far apart), and

$$\frac{1}{n^i_{t|t-1}} = e^i_{t|t-1}(1-e^i_{t|t-1})\ \mathrm{Var}(L_t) \approx \frac{a^2}{n^i_{t-1|t-1}} + \frac{1-a^2}{\bar{n}},$$

recovering the predict recursion (3). In practice we adopt (3) as the *definition* of the (e, n) dynamics; the logit-OU process is the continuous-time motivation that this recursion approximates via the delta-method round-trip (sometimes called the Laplace bridge (West & Harrison, 1997; Minka, 2001), though that term more commonly refers to the Dirichlet–Gaussian map).

C Derivation of the key fit

The key adaptation maximizes the conditional likelihood of the queries given the labels. For labeled events where chunk i is marked used, the log-likelihood of the query under the vMF emission is

$$\ell(\{k^{i,c}, \omega^{i,c}\}) = \sum_{s \in \mathcal{C}_i \cup \{0\}} \log\Big(\sum_c \omega^{i,c}\, e^{\psi\, q_s^\top k^{i,c}}\Big),$$

where $\mathcal{C}_i$ is the set of labeled events for chunk i and $q_0 = k^i_0$ represents n_p copies of the base embedding included as a prior to anchor the keys near their initial position. This is the standard vMF mixture log-likelihood (Banerjee et al., 2005). The E-step computes within-chunk responsibilities $\zeta^{i,c}_s$ ((5)). The M-step updates each key as the normalized responsibility-weighted resultant ((6)) and each weight as the responsibility mean.

C.1 When two keys separate

Take $C_i = 2$ with equal weights and keys $k \pm \delta$, δ small and tangential to k. For a query q with tangential component u, the within-chunk responsibility is $\zeta^1 = \mathrm{logistic}(2\psi\, q^\top \delta) \approx \frac{1}{2} + \psi\, u^\top \delta$. The M-step resultant for key 1 is $\sum_s \zeta^1_s q_s \approx \frac{M}{2}\rho\,\bar{k} + \psi\, S\,\delta$, where M is the number of labeled queries, $\rho\bar{k}$ their mean, and $S = \sum_s u_s u_s^\top = M\Sigma$ the tangential scatter. Normalizing by the resultant length $\frac{M}{2}\rho$ gives the new offset $\delta' \approx (\psi/\rho)\,\Sigma\,\delta$, and key 2 receives $-\delta'$. The fused point is therefore unstable in the direction of the top eigenvector of Σ iff $\psi\,\mu_{\max}/\rho > 1$; for concentrated query clusters $\rho \approx 1$, giving §4's $\psi\,\mu_{\max} > 1$. Averaging (the ρ in the denominator) pulls the keys together; specialization (the $\psi\Sigma$ term) pushes them apart.