ROUTERSHIFT: EVALUATING LLM ROUTING UNDER NON-STATIONARY TRAFFIC

Anonymous authors
Paper under double-blind review

ABSTRACT

Deployed LLM routers must keep choosing a good model as conditions change: models are constantly updated, the query context drifts, and serving latency rises under load. Standard routing benchmarks assume none of this happens. We present ROUTERSHIFT, a benchmark that treats routing as an ordered stream with controlled, labeled changes in model quality, availability, query composition, and latency. The benchmark defines twelve scenarios over $74{,}287$ queries and seven LLMs, across sixteen routing policies and three pool sizes ($k = 3, 5, 7$). Our findings are as follows: i) offline routers lead both the regret and the latency rankings, but not on language and topic shifts; ii) contextual routers, which consume query embeddings, beat their non-contextual counterparts whether online or offline (post-shift regret $.254$ against $.326$ at $k = 7$); iii) topic and language changes are the hardest shifts because they break the learned routing mapping. To speed recovery after a shift, we develop an adaptation layer (ADAPTIVESHIFT) that shortens an online router's memory or biases a frozen router's scores when the chosen model's reward drops, removing up to 11.8 regret per $1{,}000$ steps. We release construction code, stream specifications, router implementations, and evaluation code that rebuild the corpus from public datasets.

1 INTRODUCTION

Conversational systems route each request to one of several candidate LLMs to balance answer quality, latency, and cost. Research has pivoted toward predictive routers that learn from query-model outcomes or preference data Ding et al. (2024); Ong et al. (2025); Feng et al. (2025), cascade-based systems that route requests to stronger models Chen et al. (2024), and reward-based routers built on bandit feedback Wei et al. (2025); Panda et al. (2025). These methods differ in training data and objective function, yet they share the same deployment decision: selecting one model before observing its response.

Current routing benchmarks evaluate policies over a fixed query distribution and model pool Hu et al. (2024); Li et al. (2026); Lu et al. (2025); Yang et al. (2026). Deployments violate this assumption: model checkpoints are replaced, arms enter and leave the pool, query distributions drift across topics and languages, judge labels change, and serving latency changes under load. Each event alters which arm is correct, available, or fast, so one aggregate score cannot separate a router's initial quality from its loss at onset and its later recovery.

Non-stationary bandits Garivier & Moulines (2011); Besbes et al. (2014); Taberner-Miller (2026), online learning, concept drift Gama et al. (2014), and sequential change detection all study decision-making under sudden or expected change. They supply adaptation mechanisms and evaluation principles, but no shared LLM-routing workload in which the change is controlled, labeled, and paired with per-query model outcomes such as accuracy, latency, and cost.

ROUTERSHIFT supplies that workload. Each stream has a reproducible query order, a controlled shift schedule, and labels for the changed factor, shift onset, magnitude, and affected arms, so every reward change has a traceable setup. The benchmark implements twelve scenarios; the experiments here evaluate five capability and availability scenarios, one latency scenario, and topic and language instantiations of query-composition change. Across sixteen policies, no single policy is best everywhere: in the objective-matched comparison, routers that process query features obtain lower post-shift regret, while offline routers obtain the highest latency-aware scores. Topic and

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.

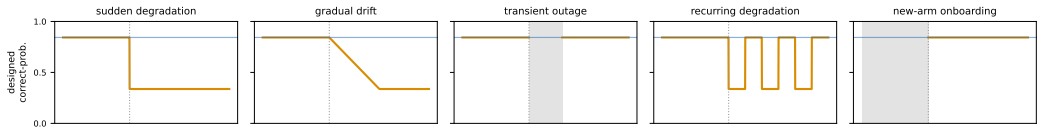


Figure 1: Five single-factor temporal patterns. Orange gives the designed correctness or availability process for the affected arm, blue the unchanged baseline.

language changes expose a different failure mode because they alter the query-to-model mapping without directly degrading an arm.

We summarize our contributions as follows:

- A benchmark for routing under non-stationarity, with twelve controlled scenarios over 74,287 labeled query records and seven-model outcome labels.
- An evaluation interface that rebuilds streams from seeds and scores any router exposing select and update operations, with the full construction code to be released upon acceptance.
- A comparison of sixteen policies under correctness-matched and latency-aware objectives across capability, availability, composition, and latency changes, together with a matched analysis of pool size and headroom.
- ADAPTIVESHIFT, a detector-gated pattern that discounts stale SWTS evidence or temporarily biases a frozen MLP sampler, evaluated from paired post-onset trajectories.

2 RELATED WORK

Predictive LLM routers pick a model before generation from offline query-model outcomes, preference labels, or learned representations. HYBRID LLM and ROUTELLM learn quality-aware routing rules Ding et al. (2024); Ong et al. (2025), and GRAPHROUTER represents query-model relations as a graph Feng et al. (2025). Cascades such as FRUGALGPT escalate to a stronger model when a cheaper one is insufficient Chen et al. (2024). BARP learns a preference-conditioned policy from logged bandit feedback, and PILOT adapts online under a budget Wei et al. (2025); Panda et al. (2025). We train BaRP once and keep it fixed during each stream, so frozen prediction and online updating differ only after the deployment changes, which is what we measure.

Routing benchmarks supply reusable outcome matrices and shared policy comparisons. ROUTERBENCH evaluates multi-model routing over precomputed outcomes Hu et al. (2024), LLMROUTERBENCH widens the model and task coverage Li et al. (2026), ROUTERARENA provides an open comparison platform Lu et al. (2025), and TWINROUTERBENCH combines static replay with live execution for agentic routing Yang et al. (2026). RouterShift keeps reproducible matrix replay and adds labeled temporal changes in capability, availability, query composition, judge, and latency.

Two recent methods target non-stationary LLM routing. PARETOBANDIT tracks quality and price through forgetting and a model registry Taberner-Miller (2026), and ROUTENLP evaluates a closed-loop router under difficulty and domain changes Guo et al. (2026). Both evaluate only their own router. RouterShift instead runs the same streams and metrics over sixteen policies with different training and feedback.

The adaptation mechanisms come from online learning and data-stream research: sliding-window and discounted estimators Garivier & Moulines (2011); Raj & Kalyani (2017), adaptive windows Bifet & Gavalda (2007), and change-point resets Page (1954); Liu et al. (2018). Tracking and dynamic-regret analyses compare a learner with a changing target Herbster & Warmuth (1998); Zinkevich (2003); Hazan & Seshadhri (2009); Daniely et al. (2015); Besbes et al. (2014); Luo et al. (2018); Auer et al. (2019), concept-drift work classifies temporal changes and scores learners in stream order Gama et al. (2014; 2013); Lu et al. (2019); Webb et al. (2016); Hinder et al. (2024), and sequential detection studies how quickly a learner identifies a change Lorden (1971); Veeravalli & Banerjee (2014). RouterShift turns these ideas into controlled LLM-routing streams with shared policies, outcomes, and shift labels.

3 PROBLEM DEFINITION

3.1 LLM ROUTING

Let $\mathcal{M}$ be a pool of k candidate models, let $\mathcal{X}$ be the space of queries, and let $x_t \in \mathcal{X}$ be the query at step t for $t = 1, \ldots, T$, where T is the stream length in queries. We write a router as a policy $\pi : \mathcal{X} \times \mathcal{H}_t \to \mathcal{M}$, where $\mathcal{H}_t$ holds the selections and observed outcomes of steps 1 through $t-1$. The router selects $a_t = \pi(x_t, \mathcal{H}_t)$. After selection it observes binary correctness $y_t(a_t) \in \{0, 1\}$ and latency $\ell_t(a_t) \in \mathbb{R}_{>0}$ for the chosen model alone. It never observes what another model would have returned, so the router learns under bandit feedback.

3.2 NON-STATIONARY ROUTING

We describe the environment at step t by the tuple

$$W_t = \big(\Theta_t,\ A_t,\ D_t,\ J_t,\ L_t\big), \tag{1}$$

where Θ_t is the outcome process that maps a query and an arm to a correctness value, $A_t \subseteq \mathcal{M}$ is the active arm set, D_t is the query distribution over $\mathcal{X}$, J_t is the labeling function that scores a response, and L_t is the latency and cost profile. A change in Θ_t alters what a model gets right. A change in J_t alters what counts as right. A stationary environment holds every factor of W_t constant, and we study environments in which one or more factors change with t.

A shift changes one or more of these factors over an interval. Its specification names the changed factor, the affected arms, a schedule $m(t) \in [0, 1]$ that sets how much of the change is active at step t, a magnitude $\delta \in [0, 1]$ that sets its full strength, and whether the onset is announced to the router. Announced onsets carry no measured gain for the sixteen comparison policies, since none of them reacts to the announcement. We implement abrupt, gradual, recurring, and transient schedule kernels, complemented by concurrent scenarios that compose two events. The twelve scenarios span five changing factors: model capability, arm availability, query composition, judge calibration, and latency. Query composition is one catalogue scenario with topic and language instantiations and with abrupt or gradual schedules. Table 2 in Appendix A lists all twelve scenarios.

4 BENCHMARK CONSTRUCTION

4.1 CORPUS

The corpus contains 74,287 labeled query records from 18 datasets in nine task categories. The categories are Knowledge, QA, Logic, Math, Data Analysis, Code, Chit-Chat, Tool Use, and Multilingual Intent Classification. The multilingual intent classification records come from MASSIVE (FitzGerald et al., 2022). We attempt to label every record against seven candidate models, for an upper bound of 520,009 query-model cells; 69,680 rows have complete seven-arm labels (Appendices D, I and K). The corpus and outcome-matrix substrate build on RouterWorld (Anonymous, 2026). RouterShift adds the shift catalogue, ordered-stream construction, and non-stationary evaluation protocol.

4.2 CONSTRUCTING ROUTERSHIFT

Streams. A stream presents one query at each step, and the router selects an arm before observing its outcome. Stream *length* controls the amount of post-shift evidence, and stream *onset* controls how much time remains for adaptation from the router perspective. The composition follows one of four modes: single dataset, same domain, cross domain, or language mix (Appendix C). We draw mode composition and shift independently.

Shifts. We apply controlled changes to the stored LLM outcomes, availability, query composition, judge labels, or latency. A capability degradation flips a correct outcome on the affected arm with probability $\delta m(t)$. Its difficulty-conditioned version flips with probability $\delta\big(1 - d(x_t)\big)$, where $d(x) \in [0, 1]$ is the query difficulty, defined as the fraction of candidate models that fail query x. An availability shift hides (outage) or adds an arm (onboarding). A context shift changes the topic or language mixture. A latency shift scales one arm's observed serving time and leaves correctness unchanged. Concurrent scenarios compose two interventions from the previous shifts. Outage

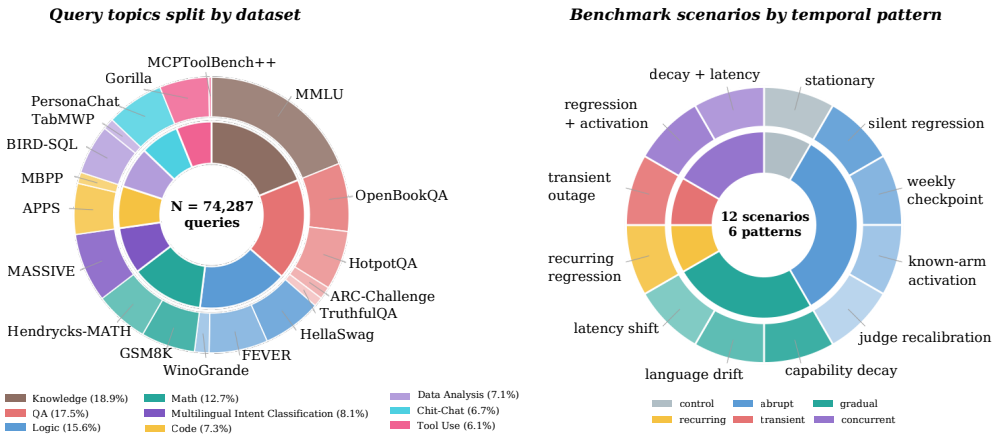


Figure 2: Left: the 74,287 query records grouped by task category and dataset. Right: the twelve benchmark scenarios grouped by temporal pattern. The inner rings show task categories and temporal patterns, and the outer rings show datasets and scenarios.

and latency follow the shape of operational events. All magnitudes and schedules are controlled interventions, not calibrated reproductions of observed incidents.

Shift intensity. Each scenario defines three primary intensity levels using a scenario-specific parameterization. For abrupt capability shifts, $\delta \in \{0.10, 0.25, 0.40\}$; the sensitivity analysis adds two intermediate values. For gradual shifts, intensity sets the ramp length.

Model-pool variants. The pool contains seven models spanning accuracy, latency, and cost. The rationale of having multiple variants is to challenge the routers under increasing variability in arms' response. We evaluate three nested variants with $k \in \{3, 5, 7\}$, whose composition is listed in Appendix B.

4.3 Annotation

For each query-model pair, we compare one decoded response with the source reference. We use task-native parsing or execution where available, including exact intent match for multilingual intent records. For open-ended responses, DeepSeek R1 emits a tagged correct or incorrect verdict under category-specific prompts, following prior LLM-judge and routing protocols (Zheng et al., 2023; Taberner-Miller, 2026). We hold DeepSeek R1 outside the candidate routing pool. An automated audit run on deterministic-based and quiz-based tests demonstrated an agreement rate of 91.58%. Each label records one sampled response. The prompts, parser routes, judge protocol, and decoding settings are detailed in Appendix I. To complement the release and the analytics tool, each stream step also records the changed factor, effective magnitude $\delta m(t)$, affected arms, and observed oracle outcomes.

4.4 Metrics

For capability and availability shifts, the primary metric is mean post-shift per-step correctness regret. We also report post-shift accuracy and L-DCG as secondary diagnostics. For the latency experiment, L-DCG is the primary ranking metric. Mean token cost per query is reported separately.

Let $C_t = \{a : y_t(a) = 1\}$ be the correct arms at step t, ordered by latency, and let r_t be the selected arm's rank in that order. Write $\mathbb{1}[\cdot]$ for the indicator function. The per-step L-DCG is

$$\text{L-DCG}_t = \mathbb{1}[a_t \in C_t] \frac{\ell_t^{\min}}{\ell_t(a_t)} \frac{1}{\log_2(r_t + 1)}, \tag{2}$$

where $\ell_t^{\min} = \min_{a \in C_t} \ell_t(a)$ is the latency of the fastest correct arm. The score lies in $[0, 1]$ and rewards correct selections close to that arm. We average it from onset to the end of the stream whenever L-DCG is reported.

For per-step reward $\rho_t(a)$ and oracle reward $\rho_t^\star$, cumulative dynamic regret over a stream of length T is

$$R_{1:T}^{\mathrm{cum}} = \sum_{t=1}^{T} \big(\rho_t^\star - \rho_t(a_t)\big). \tag{3}$$

For a shift with onset τ, the tables and policy-grid figures report mean post-shift per-step regret,

$$\bar{R}_{\mathrm{post}} = \frac{1}{T-\tau+1} \sum_{t=\tau}^{T} \big(\rho_t^\star - \rho_t(a_t)\big). \tag{4}$$

For correctness, $\rho_t(a) = y_t(a)$ and $\rho_t^\star = \mathbb{1}[C_t \neq \varnothing]$. For latency-aware evaluation, $\rho_t(a) = \mathbb{1}[a \in C_t]\ell_t^{\min}/\ell_t(a)$, and the fastest correct arm obtains one. We reserve R^{cum} for cumulative quantities and $\bar{R}_{\mathrm{post}}$ for the per-step values reported in tables.

The oracle accuracy on a query set S and pool $\mathcal{M}$ is the fraction of rows with at least one stored correct response, and it lies in $[0, 1]$. The router-free *headroom* subtracts the best fixed-arm accuracy from that quantity,

$$H(\mathcal{M}, S) = \frac{1}{|S|} \sum_{x \in S} \mathbb{1}[\exists a \in \mathcal{M} : y(x, a) = 1] - \max_{a \in \mathcal{M}} \frac{1}{|S|} \sum_{x \in S} y(x, a). \tag{5}$$

Headroom measures what a perfect router could gain over the best single model on the same rows, and it bounds what any router can add there.

Finally, the latency scenario slows one arm and leaves correctness unchanged. We compare the correctness reward $y_t(a_t)$ with the latency-aware reward

$$\rho_t^{\mathrm{lat}} = 0.5 y_t(a_t) + 0.5\big(1 - \widetilde{\ell}_t(a_t)\big), \tag{6}$$

where $\widetilde{\ell}_t(a_t)$ is the selected arm's observed latency under a running min-max normalization of that policy's observations. The coefficient 0.5 gives correctness and normalized latency equal weight.

5 Experiments and Results

5.1 Experimental Setup

Policies. We report four groups. The *online* group contains ForgettingTS, SWTS, ϵ-greedy, ParetoBandit (Taberner-Miller, 2026), SW-UCB (Garivier & Moulines, 2011), and Rexp3 (Besbes et al., 2014). The *latency-aware online* group contains two conservative bandits (Wu et al., 2016): BudgetTS enforces a latency budget, and CostTS learns from the latency-aware reward of Equation (6). The *offline* group contains NaiveBayes, KNN, MLP, Avenger, BaRP (Wei et al., 2025), and GraphRouter (Feng et al., 2025), each trained under full observability and kept fixed during evaluation. The correctness block trains per-arm correctness heads (BaRP and GraphRouter keep their native objectives), and the joint block trains on the fastest correct arm. The *references* are Random, which selects an arm uniformly at random, and StaticBest, which selects the arm with the best observed accuracy so far. The embedding-conditioned policies are ϵ-greedy, ParetoBandit, and all six offline routers; the other online, latency-aware, and reference policies do not consume query embeddings. Method details are in Appendix F.

Model pool. The seven open-weight models are Qwen3-235B-A22B (Yang et al., 2025), gpt-oss-20b (OpenAI, 2025), Gemma-4-26B-A4B (Gemma Team et al., 2026), Nemotron-Nano-9B (NVIDIA, 2025), Mistral-7B-Instruct-v0.3 (Mistral AI, 2024), codegemma-7b (CodeGemma Team et al., 2024), and deepseek-math-7b (Shao et al., 2024).

Protocol. The first experiment evaluates silent regression, capability decay, recurring regression, transient outage, arm onboarding, and latency at five stream lengths, $T \in \{1{,}000, 2{,}500, 5{,}000, 7{,}500, 10{,}000\}$, three nested model pools, and ten stream randomizations. We draw streams in two ways: an IID mixture of queries from the same datasets and a balanced cross-domain blend with equal weight per domain. For sake of visualization, Figure 3 averages the two setups, while Table 7 reports the comparison across them. Through the main and sensitivity analyses, we evaluate the five intensity values described in Section 4.2. The second experiment evaluates

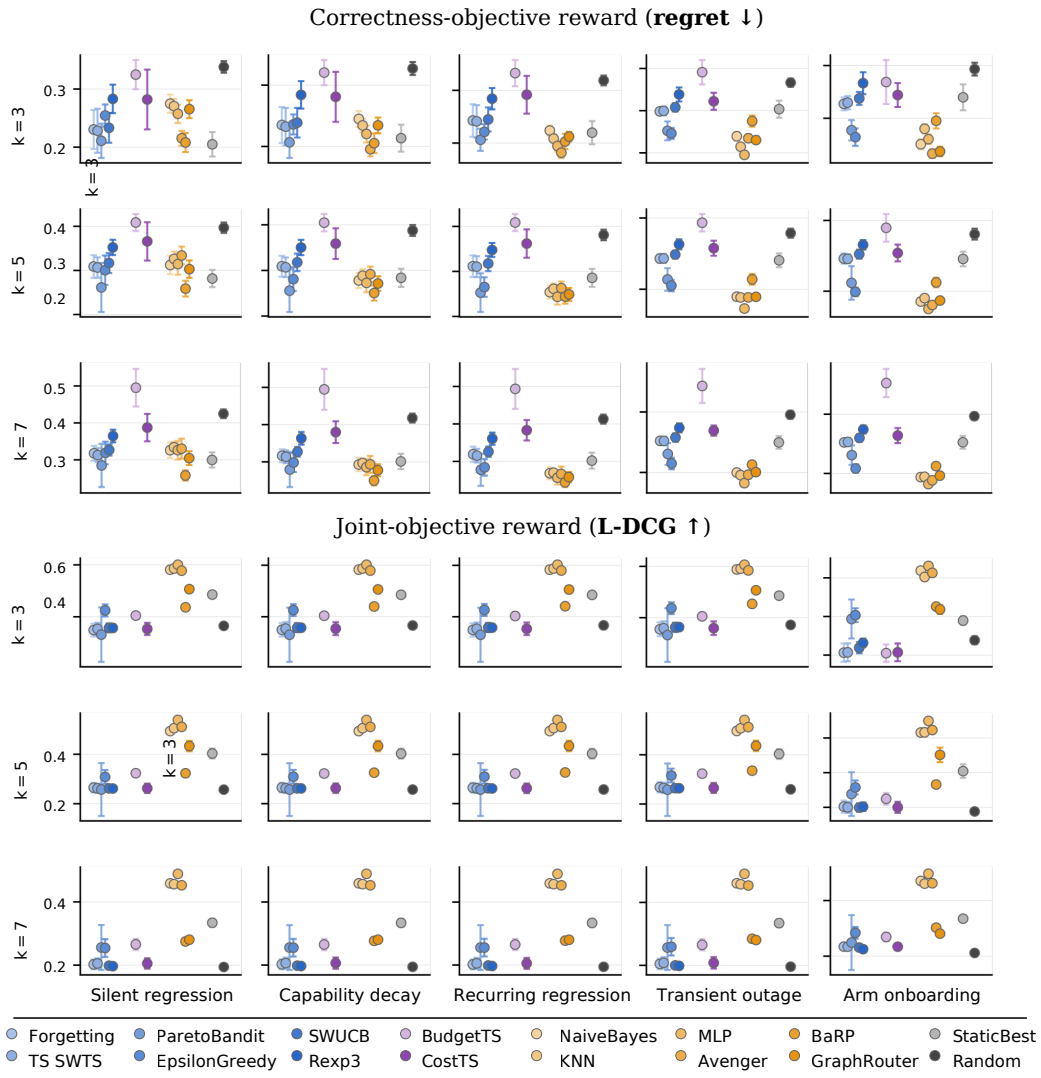


Figure 3: Per-policy results on the five capability and availability shifts. The top rows report mean post-shift per-step regret with every router trained on the correctness objective it is scored on (↓). The bottom rows report time-weighted L-DCG for the joint-matched policies (↑), online with the reward of Equation (6) and offline with fastest-correct training. All offline routers use the standard known-arm onboarding protocol; the cold-start variant is reported separately in Table 3.

topic and language composition changes at T=5,000 with gradual-transition and hard-switch schedules, on three pools, sixteen policies, and ten stream randomizations. A complementary sensitivity analysis uses fixed onset to vary memory horizon and shift magnitude at T=4,000, and separately varies stream length from 50 to 10^4. Section 5.2 reports the main effects and Appendix E gives the corresponding numerical summary.

Statistical unit. We report paired mean differences with 95% confidence intervals and exact two-sided sign tests with Holm correction. The reward-alignment comparison in Section 5.2 instead uses paired t-tests with Holm correction.

5.2 Policy Behavior Across Objectives and Shifts

Figure 3 compares the sixteen policies on the three pools, one column per shift, under two training objectives. Tables 4 to 6 report the per-shift regret values under fastest-correct offline training and

the latency-shift L-DCG under both objectives; those appendix regret columns therefore answer a different training-objective question from the objective-matched top rows of Figure 3.

Having access to context is pivotal. With every router trained on the objective it is scored on, the offline family has the lowest regret at every pool, with means of .189, .238, and .248 at k=3, 5, 7 against the standard online family's .225, .292, and .308. The offline family also leads under the joint objective (.557, .473, and .405) in every pool-shift panel of Figure 3. At k=7, the eight policies that consume query embeddings (ϵ-greedy, PARETOBANDIT, and the six offline routers) occupy the top eight positions in this objective-based regret ranking. StaticBest follows these methods and precedes the remaining non-contextual bandits. Within the standard online group, ϵ-greedy and PARETOBANDIT rank first and second at every pool. This comparison is specific to the objective-matched policy grid and does not isolate embeddings from differences in model class, training observability, or update rule.

Cost of changing reward objective. Switching a standard online policy from correctness to the joint reward raises its L-DCG and lowers its accuracy. At k=7, the L-DCG gains computed from Table 6 range from +.040 for Rexp3 to +.113 for PARETOBANDIT. The best result within this six-policy standard-online group is therefore .249 for PARETOBANDIT, against .397 for the offline block on the latency shift. BudgetTS, which belongs to the separate latency-aware group, is higher overall at .263; it also leads at k=5 with .317. BudgetTS observes latency directly through its budget constraint rather than through a shaped reward. Training-time observability, model class, and feedback differ between the families, so we do not attribute the gap to the objective alone. These results instead indicate that reward weighting and constraints must be specified per policy.

Larger pools challenge routers. Every family's regret increases and every family's L-DCG decreases as the pool grows: offline regret moves from .189 at k=3 to .248 at k=7, online regret from .225 to .308, and offline L-DCG from .557 to .405. The pool-size analysis in Table 1 holds queries, onset, shift realization, and affected arm fixed across pools. The added arms raise the oracle from .875 to .959 and widen router-free headroom from .177 to .207. Regret grows faster than headroom: +.047 for ForgettingTS and +.049 for SWTS against +.031 of headroom. One explanation is that an added arm creates another possible error, spreads bandit feedback over more arms, and makes the fastest correct arm harder to identify under L-DCG. Pool size is therefore part of the routing problem rather than a free improvement. Appendix E gives the sensitivity summary.

			Regret (mean ± SD)	
Pool	Oracle	Headroom	FTS	SWTS
k=3	.875	.177	.186 ± .007	.187 ± .006
k=5	.936	.184	.202 ± .008	.202 ± .008
k=7	.959	.207	.234 ± .009	.236 ± .009

Table 1: Pool-size analysis for silent regression. All pools use the same queries, shift realization, onset, and affected arm

5.3 CONTEXTUAL SHIFTS

We assess routing performance when the query mixture changes abruptly or gradually by topic or language. For the language setup, MASSIVE intent traffic moves from English to a mixture of German, Japanese, Arabic, Thai, and Swahili (FitzGerald et al., 2022); for the topic setup, traffic moves from trivia and chat to code and math. Each case uses 5,000 steps, ten stream randomizations per pool, and gradual-transition or hard-switch schedules.

For the language shift, oracle accuracy falls because several arms answer non-English queries poorly. Offline routers keep applying a mapping learned for the source distribution, so their loss grows after the mixture changes. In Figure 4, the offline gap grows from 0.144 to 0.255, while the online gap changes only from 0.074 to 0.096 because these policies reroute within a few hundred steps. The injection schedule matters little at the endpoint: gradual-transition and hard-switch scenarios differ by less than 0.005. Under these settings, a router can therefore fail without any model checkpoint changing, a failure mode hidden by a stationary benchmark.

The topic change is the opposite case. The code-and-math side of the corpus is better covered by the pool, so the oracle itself rises after onset and every reporting group rises with it. Routing also genuinely improves: under the hard switch at $0.5T$, the online group's gap to the oracle at k=7 narrows from 0.217 to 0.140. The gain depends on the pool holding a specialist for the new mixture: at k=3 the narrowing does not appear, because the small pool has no code or math specialist to

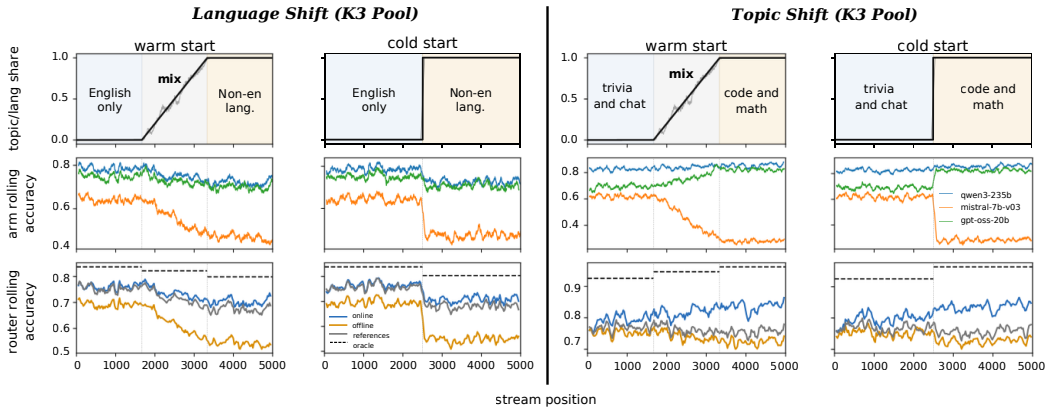


Figure 4: Composition-shift examples at k=3. Left: language; right: topic. Each figure compares a gradual transition with a hard switch and shows stream composition, per-arm rolling accuracy, and the four reporting groups.

discover. A favorable destination therefore lifts raw accuracy for free, but it improves routing only when there is a right arm to find. Full traces are in Appendix H.

6 AdaptiveShift: Detector-Gated Adaptation

The per-policy overview in Figure 3 shows that post-shift loss depends on the changed factor, update rule, and pool. These patterns motivate an arm-state response that retains the base selection rule. We call this approach AdaptiveShift.

Operating on routers. AdaptiveShift computes arm-specific reward statistics from the selected arm's outcomes and applies a bounded response when evidence indicates a change. We instantiate that pattern for two stochastic bases. On SWTS, it discounts the selected arm's posterior counts. On a accuracy-trained frozen MLP deployed as a calibrated sampler, it temporarily lowers the selected arm's log-odds score. Each wrapped policy is compared with the identical unwrapped selection rule; Algorithm 1 shows the TS instantiation.

Algorithm 1. AdaptiveShift wrapper for a TS router $\mathcal{B}$.

In: TS router $\mathcal{B}$, reward r_t, event $e_{a,t}$
Per arm: γ_a, c_a, μ_a, h_a, s_a
$a \leftarrow \mathcal{B}.\text{SELECT}()$ ▷ base
if $c_a > 0$ **then** ▷ one-step lag
  scale arm a counts by γ_a; $c_a \leftarrow c_a - 1$
end if
$\mathcal{B}.\text{UPDATE}(a, r_t)$
update μ_a, h_a (λ=12, δ=0.05), s_a (α=0.1)
if $e_{a,t} = 1$ **then** ▷ notify
  $(\gamma_a, c_a) \leftarrow (0.50, 16)$
else if $h_a > \lambda$ **then** ▷ PH alarm
  $(\gamma_a, c_a) \leftarrow (0.85, 8)$
else if $s_a \leq -0.20$ **then** ▷ residual
  $(\gamma_a, c_a) \leftarrow (0.98, 4)$
end if

AdaptiveShift. Event notifications, Page–Hinkley evidence, and reward residuals select an arm-specific response level. The SWTS instantiation uses retention factors 0.50, 0.85, and 0.98 for 16, 8, and 4. The frozen-MLP instantiation uses score biases -2.0, -1.0, and -0.3 for the same hold windows. The outcome protocol fixes k=7, uses ten stream randomizations, and evaluates silent regression, capability decay, recurring regression, transient outage, and hard and gradual topic and language changes. Curves are aligned at each onset and truncated to a common 5,000-step post-onset window; each composition curve pools its two schedules.

Results. Figure 5 and Table 9 report the same stored traces. AdaptiveShift-SWTS lowers recurring-regression regret from .3202 to .3145, corresponding to 28.4 fewer units of cumulative regret over the plotted window. Its silent-regression and pooled-language gains are 14.3 and 17.7 units; transient outage instead costs 4.2. AdaptiveShift-MLP has its largest gain on silent regression (28.7 units) and improves recurring regression by 11.6, but costs 21.2 units on pooled language change.

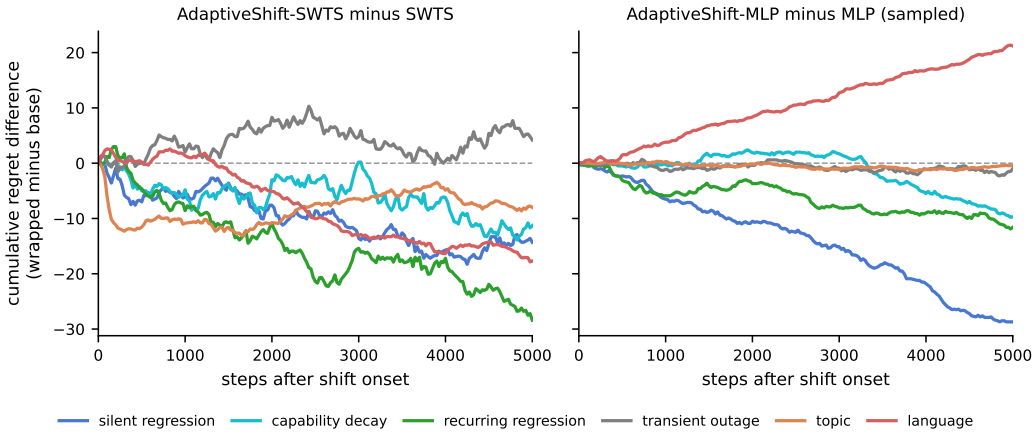


Figure 5: Cumulative paired regret difference from each condition's onset on k=7, averaged over ten randomizations at 5,000 post-onset steps. Negative values favor AdaptiveShift. Left: the count-retention SWTS wrapper. Right: the score-bias frozen-MLP wrapper. Topic and language each pool hard and gradual schedules (20 traces per curve). Table 9 is computed from these same traces.

Scope. Across these experiments, AdaptiveShift results in a potential starting point for improving routers for tackling of non-stationarities. They show that the detector-gated response can remove stale evidence, yet its value depends on both the shift and actuator. Full settings, trace-matched values, and detector diagnostics are in Appendix G.

7 Conclusion

We introduce RouterShift, a benchmark that makes non-stationarity a first-class axis of LLM-router evaluation. It provides ordered streams built from a labeled query corpus, twelve controlled shift scenarios in which every reward change has a known cause and onset, and an evaluation interface that scores any router that selects a model and observes one outcome. In the objective-matched capability and availability comparison, the eight embedding-conditioned policies occupy the top eight regret positions at k=7, while offline routers lead the latency-aware ranking. Query-composition changes expose a different failure mode by breaking the learned query-to-model mapping without changing model checkpoints. On the trace-matched AdaptiveShift protocol, detector-gated count retention improves SWTS on five of six plotted families, while detector-gated score bias improves the sampled MLP on five of six; the exceptions differ by base. Every router remains far from a shift-aware oracle, highlighting a clear opportunity for future work.

Limitations. Our shifts are controlled interventions on stored outcomes, and the benchmark has not yet been evaluated against real deployment traffic, where changes arrive unlabeled, overlap, and mix with ordinary drift. The evaluation also assumes correctness feedback after every step. In deployment, this signal often comes from an LLM judge with its own errors or from delayed, deterministic user-level metrics. AdaptiveShift currently covers one online TS base and one sampled frozen scorer; deterministic offline deployment, other router families, and arm arrivals remain open.

Reproducibility Statement

The reference implementation contains corpus scoring, stream construction, sixteen comparison routers, the AdaptiveShift layer, and result-generation code. This code will be released upon acceptance, together with source identifiers, prompts, judge settings, split identifiers, scenario specifications, detector settings, stored seeds, and the evaluation interface. These components reproduce the corpus from its public source datasets and rebuild every stream and result.

REFERENCES

Anonymous. Routerworld: Scaling conversational ai with latency-aware llm routing. *Under review*, 2026.

Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on Learning Theory (COLT)*, 2019.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, 2007.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. In *Transactions on Machine Learning Research*, 2024.

CodeGemma Team, Heri Zhao, Jeffrey Hui, et al. Codegemma: Open code models based on gemma, 2024. URL https://arxiv.org/abs/2406.11409.

Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning (ICML)*, 2015.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, et al. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations (ICLR)*, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations (ICLR)*, 2025.

Jack FitzGerald et al. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*, 2022.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory*, 2011.

Gemma Team et al. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Nature*, 645:633–638, 2025.

Dongxin Guo, Jikun Wu, and Siu Ming Yiu. Routenlp: Closed-loop llm routing with conformal cascading and distillation co-optimization. *arXiv preprint arXiv:2604.23577*, 2026.

Elad Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *International Conference on Machine Learning (ICML)*, 2009.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2): 151–178, 1998.

Fabian Hinder, Valerie Vaquet, and Barbara Hammer. One or two things we know about concept drift – a survey on monitoring in evolving environments. *Frontiers in Artificial Intelligence*, 7, 2024.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, et al. Routerbench: A benchmark for multi-llm routing system. In *Agentic Markets Workshop at ICML*, 2024.

Hao Li, Yiqun Zhang, et al. Llmrouterbench: A massive benchmark and unified framework for llm routing. *arXiv preprint arXiv:2601.07206*, 2026.

Fang Liu, Joohyun Lee, and Ness Shroff. A change-detection based framework for piecewise-stationary multi-armed bandit problem. In *AAAI Conference on Artificial Intelligence*, 2018.

Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.

Yifan Lu, Rixin Liu, et al. Routerarena: An open platform for comprehensive comparison of llm routers. *arXiv preprint arXiv:2510.00202*, 2025.

Haipeng Luo, Chen-Yu Wei, Alekh Agarwal, and John Langford. Efficient contextual bandits in non-stationary worlds. In *Conference on Learning Theory (COLT)*, 2018.

Mistral AI. Mistral-7b-instruct-v0.3. Hugging Face model card, 2024. URL https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3.

NVIDIA. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/242f0218e31e60494c4cf7642dce2bce-Abstract-Conference.html.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, et al. Adaptive llm routing under budget constraints. *Findings of EMNLP*, 2025.

Vishnu Raj and Sheetal Kalyani. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Annette Taberner-Miller. Paretobandit: Budget-paced adaptive routing for non-stationary llm serving. *arXiv preprint arXiv:2604.00136*, 2026.

Venugopal V. Veeravalli and Taposh Banerjee. Quickest change detection. *Academic Press Library in Signal Processing*, 3:209–255, 2014.

Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

Wang Wei, Tiankai Yang, Hongjie Chen, Yue Zhao, Franck Dernoncourt, Ryan A. Rossi, and Hoda Eldardiry. Learning to route LLMs from bandit feedback: One policy, many trade-offs. *arXiv preprint arXiv:2510.07429*, 2025.

Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *International Conference on Machine Learning (ICML)*, 2016.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Pei Yang, Wanyi Chen, et al. Twinrouterbench: Fast static and live dynamic evaluation for realistic agentic llm routing. *arXiv preprint arXiv:2605.18859*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS Datasets and Benchmarks*, 2023.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, 2003.

A THE TWELVE SHIFT SCENARIOS

Table 2 groups the twelve scenarios by changing factor. Capability, availability, and judge changes alter the outcome process directly, query composition changes the query mixture, and latency changes serving time alone. The five single-factor patterns appear in Figure 1, and Figure 6 shows each on one stream.

Scenario	Factor	Pattern	Onset	Description
stationary	n/a	control	n/a	no shift
silent regression	capability	abrupt	unknown	one arm degrades without notification
weekly checkpoint	capability	abrupt	known	announced checkpoint replacement
arm onboarding	availability	abrupt	known	a trained arm becomes available
judge recalibration	judge	abrupt	known	one shared event re-labels all arms
capability decay	capability	gradual	unknown	one arm degrades along a ramp
query composition	context	gradual/abrupt	unknown	the topic or language mixture changes
latency shift	latency	gradual	unknown	one arm slows without changing correctness
recurring regression	capability	recurring	unknown	a regression returns on a cycle
transient outage	availability	transient	unknown	an arm disappears and returns
regression + onboarding	capability, availability	concurrent	mixed	regression at $T/3$, onboarding at $2T/3$
decay + latency	capability, latency	concurrent	unknown	decay from $T/4$ to $3T/4$, slowdown from $T/3$

Table 2: The twelve implemented scenarios, ordered by temporal pattern. Mixed onset combines one unknown event and one known event.

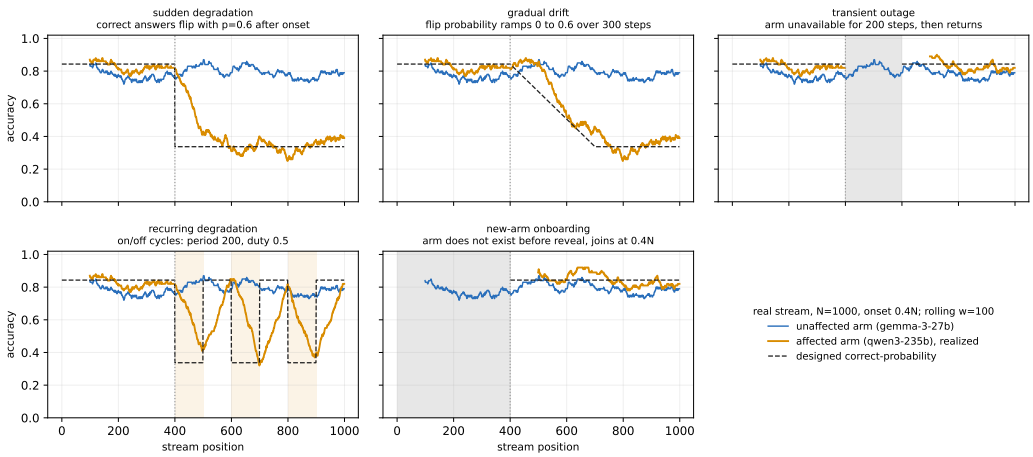


Figure 6: Five correctness and availability shifts on one T=1,000 stream at k=3. Dashed black gives the designed probability for the affected Qwen3-235B arm, orange its realized rolling accuracy, and blue the unaffected gpt-oss-20b arm. Latency is evaluated in Section 5.2.

A known shift calls `notify_shift` before the onset step and provides the shift type and affected arms. None of the sixteen comparison policies changes state in response. AdaptiveShift uses this notification for its bounded strong response. The main onboarding results use one common information condition: the arriving arm appears in offline training and is hidden only before its onboarding step. We separately evaluate a cold-start ablation for MLP and KNN. In that ablation, the arriving arm is withheld from offline training, so each router is fitted on the remaining $K-1$ arms and holds no ranking slot for the arm it will meet. We report three arrival rules: ignore the arm, always take it when available, and explore between following the frozen ranking and taking the new arm. Table 3 reports all three against the standard trained-arm condition; none of its cold-start values is

Table 3: Cold-start arm onboarding. Post-shift regret on the onboarding shift (↓), mean over ten stream randomizations and five stream lengths. *Trained* is the standard condition, in which the arriving arm is part of offline training. The three cold-start columns withhold it and differ only in the arrival rule. Every cold-start value differs from its *trained* baseline after Holm correction (paired t over the ten randomizations), except KNN *ignore* at k=7.

		Trained	Cold start (arm withheld from training)		
Pool	Router	on the arm	ignore	explore	always take
k=3	MLP	.104	.154	.160	.219
	KNN	.126	.174	.180	.219
k=5	MLP	.137	.146	.150	.274
	KNN	.171	.177	.180	.274
k=7	MLP	.153	.159	.164	.287
	KNN	.184	.184	.189	.287

mixed into Figure 3 and Tables 4 to 6. Judge recalibration applies one shared relabeling event to every arm for an affected query. The concurrent regression-plus-onboarding case places the regression at one third of the stream and onboarding at two thirds. The decay-plus-latency case applies quality decay from one quarter to three quarters and latency growth from one third to the end.

B Model Pool Variants and Reward-Matrix Budget

The corpus contains 74,287 labeled records. With seven models, the outcome-label upper bound is 520,009 query-model cells. Per-pool evaluation uses rows with complete coverage for every selected model. The models span accuracy, latency, cost, architecture, and training mixtures. The code and math models contribute complementary per-query outcomes, while the stored matrix records their observed performance on every dataset.

The nested variants are k=3 = {Qwen3-235B, gpt-oss-20b, Mistral-7B}, k=5 with codegemma-7b and Gemma-4-26B-A4B added, and k=7 with deepseek-math-7b and Nemotron-Nano-9B added. Cross-pool results use pool-specific eligible rows and training sizes, so those trends remain descriptive. The matched pool-size experiment holds queries and shift realizations fixed.

C Composition Modes and Stream Lengths

A stream row contains the query identifier, shifted outcome vector, observed latencies, and shift labels. The four composition modes are single dataset, same domain, cross domain, and language mix. The generator supports lengths of 1,000, 2,500, 5,000, 7,500, and 10,000 steps. The release manifest lists the frozen streams by scenario, length, composition mode, and seed.

D Corpus Characterization

The corpus contains 74,287 labeled query records. Source-ID uniqueness does not imply unique text: 441 records in the general-task data share prompt text. Item difficulty is measured from stored model outcomes. In the general-task data, 14.7% of records have all seven arms correct, 38.5% have at most one correct, and the remaining 46.8% have between two and six correct. These bands describe one stored generation per query-model cell.

E Effects of Shift Size and Stream Length

A single shift size and stream length gives one method ordering. We therefore vary two quantities at a time. The first comparison varies the effective memory horizon H_{mem} and degradation magnitude δ at T = 4,000 with onset at $0.4T$. The second varies stream length from 50 to 10^4 together with degradation magnitude, latency multiplier, outage duration, or onboarding time.

Forgetting rules respond differently to larger shifts. Post-shift regret rises with degradation magnitude for every policy, but at different rates. At T=4,000, regret correlates with magnitude at $+0.98$ for SWTS and ϵ-greedy and $+0.73$ for SW-UCB. SW-UCB is about as sensitive to stream length as to magnitude (0.042 against 0.039), while the stochastic methods are two to four times more sensitive to magnitude. SW-UCB drops a degraded arm after a bounded amount of evidence and then stops sampling it, whereas posterior sampling keeps paying the flip probability each time it returns. SW-UCB has the lower regret on 73% of tested magnitudes, and its margin grows with magnitude. For $\delta \leq 0.15$ on short and moderate streams, ϵ-greedy has the lower regret on 92% of values, and SW-UCB regains the majority for $T > 2{,}500$.

The metric must match the changed factor. Under the latency shift, correctness regret selects ϵ-greedy on 75% of tested values. Latency-aware regret selects SW-UCB on 71% and ϵ-greedy on 3%. Correctness does not change under this shift, so a correctness metric rewards whichever policy explores least. We therefore use mean post-shift correctness regret as the primary metric for capability and availability changes and mean post-shift L-DCG as the primary metric for latency. Across-seed standard deviation of post-shift regret is 0.045 to 0.051 for $T \leq 250$ and 0.009 to 0.017 for $T \geq 4{,}000$, so we exclude rankings below $T = 250$ and start the release stream specifications at $T = 1{,}000$.

F Router Families: Training, Results, and Ablations

F.1 Splits and Training Details

Each pool uses a source-ID split with seed 777. Offline routers train on rows disjoint from the stream pool, and shifts are applied only during evaluation. Pool variants contain different eligible rows and training sizes, so cross-pool levels remain descriptive. For the correctness-objective comparison, the offline target is a per-arm correctness label; for fastest-correct training, the target is the fastest correct arm with a sentinel class when no arm is correct. All contextual methods use the same frozen 256-dimensional query embedding. Arm onboarding hides an arm before its onboarding step but retains that arm in offline training. Under the cold-start variant of Table 3, the arm is withheld instead, so MLP and KNN are fitted on $K-1$ arms and meet it with no trained slot for it.

The language experiment is a transfer evaluation. Offline routers train on the core source-ID fit split, which contains no MASSIVE records, and are then evaluated on the six MASSIVE locales. English is the source phase of the evaluation stream, not an additional training split. This English-to-multilingual protocol is specific to the language experiment; the generic source-ID split applies to the other experiments.

Among the offline routers, NaiveBayes is Gaussian naive Bayes, KNN uses k=5, and MLP uses a $512 \rightarrow 256 \rightarrow 128 \rightarrow K{+}1$ head. Avenger calibrates arm rankings within 64 embedding clusters, BaRP (Wei et al., 2025) trains a preference-conditioned policy under full observability, and GraphRouter (Feng et al., 2025) uses a bipartite query-arm graph. The online policies see only the selected arm's outcome: SW-UCB uses a sliding observation window, Rexp3 restarts every memory horizon, BudgetTS enforces a latency budget, and CostTS uses the latency-aware scalar reward with weight 0.5.

F.2 Full Policy Results by Shift and Pool

Tables 4 to 6 report per-shift values with the offline routers under their fastest-correct training, so their offline regret entries differ from the objective-matched points of Figure 3. The tables retain policy identity and shift type, so they show within-family spread and prevent a family mean from being interpreted as policy-level dominance. One caveat applies to KNN: its five-neighbour scores tie on the top arm for 63.3% of queries at k=7, and ties are broken by an unstable sort, so its printed rankings depend in part on tie order.

Policy	Silent $\bar{R}_{\text{post}}$	Decay $\bar{R}_{\text{post}}$	Recurring $\bar{R}_{\text{post}}$	Outage $\bar{R}_{\text{post}}$	Onboarding $\bar{R}_{\text{post}}$	Mean $\bar{R}_{\text{post}}\downarrow$	L-DCG corr.↑	L-DCG joint↑	Cost (tokens)↓
ForgettingTS	0.202	0.210	0.215	0.197	0.231	0.211	0.297	0.338	240.793
SWTS	*0.200*	*0.208*	0.215	0.197	0.234	0.211	0.292	0.346	240.914
ParetoBandit	0.212	0.209	0.208	0.154	0.189	*0.194*	0.364	0.344	196.682
ϵ-greedy	0.248	0.232	0.220	*0.149*	*0.171*	0.204	0.377	0.400	193.113
SW-UCB	0.211	0.219	0.228	0.209	0.237	0.221	0.304	0.343	231.798
Rexp3	0.269	0.272	0.274	0.233	0.275	0.264	0.334	0.353	200.301
BudgetTS	0.306	0.311	0.317	0.300	0.305	0.308	0.404	0.404	141.245
CostTS	0.255	0.261	0.272	0.231	0.261	0.256	0.346	0.346	208.224
NaiveBayes	0.227	0.220	0.215	0.179	0.198	0.208	*0.477*	*0.579*	139.843
KNN	0.231	0.225	0.222	0.190	0.208	0.215	0.427	*0.579*	138.622
MLP	0.235	0.228	0.224	0.191	0.208	0.217	0.379	**0.601**	125.321
Avenger	0.272	0.263	0.258	0.222	0.238	0.251	0.344	0.576	*114.166*
BARP	0.245	0.210	**0.187**	**0.096**	**0.103**	**0.168**	0.426	0.426	193.330
GRAPHROUTER	0.321	0.328	0.334	0.320	0.352	0.331	**0.515**	0.515	**92.185**
StaticBest	**0.188**	**0.195**	*0.201*	0.187	0.219	0.198	0.275	0.494	250.781
Random	0.338	0.328	0.319	0.263	0.295	0.308	0.364	0.364	172.690

Table 4: Policy-level shift results for k=3. Each value uses ten stream randomizations after averaging five lengths. The regret columns use fastest-correct offline training and therefore differ from the correctness-objective offline values in Figure 3 and Table 7. L-DCG corr. scores the correctness-objective policy on the fixed-onset latency shift; L-DCG joint scores the objective-matched policy (online with the reward of Equation (6), offline trained on the fastest correct arm). BudgetTS, CostTS, BARP, GRAPHROUTER, and Random have one fixed objective, so both columns coincide.

Policy	Silent $\bar{R}_{\text{post}}$	Decay $\bar{R}_{\text{post}}$	Recurring $\bar{R}_{\text{post}}$	Outage $\bar{R}_{\text{post}}$	Onboarding $\bar{R}_{\text{post}}$	Mean $\bar{R}_{\text{post}}\downarrow$	L-DCG corr.↑	L-DCG joint↑	Cost (tokens)↓
ForgettingTS	0.288	0.290	0.291	0.285	0.297	0.290	0.189	0.257	238.323
SWTS	0.287	0.290	0.291	0.284	0.296	0.290	0.187	0.256	238.606
ParetoBandit	**0.260**	**0.254**	*0.252*	0.226	0.222	*0.243*	0.219	0.238	214.316
ϵ-greedy	0.287	0.271	0.259	*0.213*	*0.200*	0.246	0.240	0.283	205.435
SW-UCB	0.299	0.304	0.306	0.299	0.305	0.302	0.190	0.248	226.850
Rexp3	0.341	0.342	0.341	0.322	0.335	0.336	0.224	0.262	199.417
BudgetTS	0.397	0.397	0.398	0.393	0.397	0.396	*0.317*	0.317	121.041
CostTS	0.337	0.338	0.341	0.324	0.328	0.334	0.256	0.256	198.316
NaiveBayes	0.312	0.307	0.302	0.288	0.289	0.300	0.295	0.486	129.312
KNN	0.349	0.342	0.337	0.321	0.321	0.334	0.307	0.496	115.012
MLP	0.347	0.343	0.339	0.328	0.329	0.337	0.233	**0.538**	100.318
Avenger	0.358	0.355	0.352	0.344	0.345	0.351	0.235	*0.512*	**88.382**
BARP	0.301	*0.263*	**0.238**	**0.155**	**0.145**	**0.220**	0.309	0.309	197.992
GRAPHROUTER	0.462	0.455	0.450	0.429	0.431	0.445	**0.416**	0.416	*99.847*
StaticBest	*0.266*	0.268	0.270	0.266	0.274	0.269	0.206	0.389	250.781
Random	0.396	0.387	0.381	0.357	0.362	0.377	0.255	0.255	169.174

Table 5: Policy-level shift results for k=5. Each value uses ten stream randomizations after averaging five lengths. The regret columns use fastest-correct offline training and therefore differ from the correctness-objective offline values in Figure 3 and Table 7. L-DCG corr. scores the correctness-objective policy on the fixed-onset latency shift; L-DCG joint scores the objective-matched policy (online with the reward of Equation (6), offline trained on the fastest correct arm). BudgetTS, CostTS, BARP, GRAPHROUTER, and Random have one fixed objective, so both columns coincide.

F.3 STREAM-COMPOSITION ROBUSTNESS

Table 7 reports the correctness-objective family means under both stream constructions. The balanced construction weights the eight domains equally, while the IID mixture draws stream rows uniformly. The paired contrasts quoted in Section 5.2 use these values.

G ADAPTIVESHIFT IMPLEMENTATION AND DETAILED RESULTS

G.1 SHIFT FAMILIES AND DETECTOR EVIDENCE

G.2 IMPLEMENTATION

AdaptiveShift stores constant-size detector state per arm. The SWTS instantiation sets Page–Hinkley threshold λ_{PH}=12, drift δ_{PH}=0.05, reward-residual threshold τ_s=0.20, and window size 1,000. Background, mild, medium, and strong retention factors are 1.00, 0.98, 0.85, and 0.50; response durations are 4, 8, and 16 updates. The frozen-MLP instantiation converts per-arm correctness probabilities to log odds, divides by sampling temperature 0.25, and adds temporary score biases -0.3, -1.0, and -2.0. Its Page–Hinkley threshold is 3, drift is 0.005, residual threshold is -0.20, and

Policy	Silent $\bar{R}_{\text{post}}$	Decay $\bar{R}_{\text{post}}$	Recurring $\bar{R}_{\text{post}}$	Outage $\bar{R}_{\text{post}}$	Onboarding $\bar{R}_{\text{post}}$	Mean $\bar{R}_{\text{post}}\downarrow$	L-DCG corr.↑	L-DCG joint↑	Cost (tokens)↓
ForgettingTS	0.309	0.309	0.309	0.305	0.309	0.308	0.118	0.190	266.641
SWTS	0.306	0.308	0.308	0.305	0.308	0.307	0.115	0.195	268.428
ParetoBandit	**0.266**	**0.266**	*0.266*	0.258	0.267	*0.265*	0.136	0.249	262.616
ϵ-greedy	0.312	0.293	0.281	*0.231*	*0.222*	0.268	0.190	0.238	221.083
SW-UCB	0.321	0.320	0.323	0.315	0.320	0.320	0.124	0.195	252.816
Rexp3	0.359	0.360	0.357	0.347	0.352	0.355	0.154	0.194	222.235
BudgetTS	0.516	0.509	0.513	0.506	0.522	0.513	0.263	0.263	87.406
CostTS	0.363	0.365	0.367	0.346	0.347	0.357	0.195	0.195	197.595
NaiveBayes	0.337	0.331	0.327	0.313	0.313	0.324	0.234	0.450	116.015
KNN	0.397	0.392	0.388	0.381	0.375	0.387	0.242	0.443	92.240
MLP	0.375	0.369	0.365	0.356	0.352	0.364	0.177	**0.484**	86.707
Avenger	0.363	0.359	0.355	0.352	0.344	0.355	0.184	*0.456*	*85.182*
BARP	0.311	*0.273*	**0.246**	**0.169**	**0.157**	**0.231**	*0.264*	0.264	196.433
GRAPHROUTER	0.574	0.575	0.576	0.574	0.579	0.575	**0.283**	0.283	**49.783**
StaticBest	*0.283*	0.284	0.284	0.282	0.287	0.284	0.157	0.322	250.781
Random	0.425	0.418	0.413	0.389	0.394	0.408	0.190	0.190	177.140

Table 6: Policy-level shift results for k=7. Each value uses ten stream randomizations after averaging five lengths. The regret columns use fastest-correct offline training and therefore differ from the correctness-objective offline values in Figure 3 and Table 7. L-DCG corr. scores the correctness-objective policy on the fixed-onset latency shift; L-DCG joint scores the objective-matched policy (online with the reward of Equation (6), offline trained on the fastest correct arm). BudgetTS, CostTS, BARP, GRAPHROUTER, and Random have one fixed objective, so both columns coincide.

Pool	Online	Latency-aware	Offline	References
IID mixture streams				
k=3	*.218*	.282	**.185**	.253
k=5	*.285*	.365	**.229**	.323
k=7	*.304*	.435	**.239**	.346
Balanced cross-domain streams				
k=3	*.233*	.290	**.194**	.272
k=5	*.299*	.374	**.248**	.339
k=7	*.311*	.423	**.257**	.363

Table 7: Mean post-shift per-step regret under correctness-objective training for both stream constructions. The offline values are replayed with correctness targets; they must not be compared with the fastest-correct offline regret columns in Tables 4 to 6. **Bold**: best family mean; *italic*: second best.

residual-EWMA decay is 0.9. The unwrapped comparator uses the identical MLP scores, temperature, availability mask, and randomization without detector bias.

For external-event indicator $e_{a,t}$, Page–Hinkley statistic $h_{a,t}$, and reward residual $s_{a,t}$, the retention rule is

$$\gamma_{a,t} = \begin{cases} \gamma_{\text{strong}}, & e_{a,t} = 1, \\ \gamma_{\text{medium}}, & h_{a,t} > \lambda_{\text{PH}}, \\ \gamma_{\text{mild}}, & s_{a,t} < -\tau_s, \\ \gamma_{\text{background}}, & \text{otherwise.} \end{cases} \tag{7}$$

G.3 TRACE-MATCHED OUTCOME COMPARISON

Figure 5 and Table 9 are generated from the same cumulative-regret archive. Each table entry is the mean regret accumulated during the first $5{,}000$ post-onset steps divided by $5{,}000$. The $\Delta\times10^3$ columns give AdaptiveShift minus base in regret units per $1{,}000$ steps; multiplying them by five gives the endpoint of the corresponding cumulative-difference curve. Hard and gradual composition schedules contribute equally to their pooled rows.

G.4 MATCHED POOL-SIZE AND DETECTOR-DIAGNOSTIC PROTOCOLS

The matched pool-size experiment draws all streams from the $69{,}680$ rows with complete seven-arm labels. It uses the same query identifiers, order, onset, affected arm, and keyed degradation decisions for $k \in \{3, 5, 7\}$. The result file records one row per method, pool, and stream randomization,

Shift family	Plain meaning	Example	AdaptiveShift evidence
Abrupt arm change	Model quality drops at one time point	Checkpoint replacement, silent regression	Strong: immediate reward drop on affected arm
Gradual capability decay	Quality declines slowly over many steps	Progressive model degradation	Weak: slow accumulation, late trigger
Recurring change	Quality cycles or repeats a pattern	Periodic regression and recovery	Weak: stale evidence reused across cycles
Availability change	Model becomes unavailable or returns	Outage, arm onboarding	None: no reward stream from unavailable arm
Topic composition	Query topics shift from easy to hard	Trivia and chat to code and math	Indirect: diffused across queries
Language composition	Query language distribution changes	English to German, Japanese, Arabic	Indirect: diffused across queries
Latency change	Response time increases	Serving slowdown	Indirect: enters only through latency reward

Table 8: Shift families and the evidence available to the AdaptiveShift detector. The detector operates on arm-specific reward feedback and responds to changes that produce a sustained shift in the selected arm's reward stream.

	SWTS			MLP (sampled)		
Scenario	Base	AS	$\Delta\times10^3$	Base	AS	$\Delta\times10^3$
silent regression	.3127	**.3099**	-2.86	.2684	**.2626**	-5.74
capability decay	.3162	**.3139**	-2.26	.2307	**.2287**	-1.94
recurring regression	.3202	**.3145**	-5.68	.2398	**.2375**	-2.32
transient outage	**.2995**	.3003	$+0.84$	.1931	**.1929**	-0.20
topic (hard + gradual)	.1461	**.1445**	-1.60	.1182	**.1181**	-0.09
language (hard + gradual)	.1344	**.1309**	-3.54	**.2751**	.2793	$+4.24$

Table 9: Trace-matched AdaptiveShift outcomes for the protocol of Figure 5: k=7, T=10,000, ten randomizations, and a 5,000-step post-onset window. AS denotes AdaptiveShift and lower is better; bold marks the lower value within each base/wrapper pair. Negative Δ favors AdaptiveShift. Composition rows pool hard and gradual schedules, giving 20 traces per row.

together with oracle accuracy, best-fixed accuracy, router-free headroom, post-shift regret, affected-arm observations, trigger status, and delay. Table 1 in Section 5.2 reports the resulting oracle, headroom, and regret.

Separately, Table 10 is a 20-randomization detector-only diagnostic for the SWTS instantiation at k=7 and T=10,000. It includes weekly-checkpoint and latency changes to characterize alarms beyond the outcome rows of Table 9.

G.5 Detector Behavior

H Composition-Shift Analysis

This appendix expands the composition-shift results of Section 5.3. Phase boundaries are part of the stream specification, so per-phase accuracies and matrix-oracle accuracies are read directly from the trace. Figures 7 to 12 show all three pools and add a per-step selection raster.

Topic shift. Raw accuracy rises for every reporting group after the change because the pool covers code and math better, and the oracle rises with it. At k=7, the online gap narrows from 0.217 to 0.140 under the hard switch at $0.5T$. Under the gradual transition, it narrows from 0.214 to 0.176 to 0.128 through the mixing phase. The offline gap narrows from 0.281 to 0.195 under the hard switch, and the references move least. The final online gap is lower after the gradual transition than after the hard switch at every pool size. At k=3, neither effect appears because the small pool has no specialist for the new topic.

Stream	Trigger rate	False alarms per 10k pre-onset	Median delay (steps)
Weekly checkpoint	1.00	0.5	1,814
Silent regression	1.00	0.5	265
Capability decay	1.00	0.4	2,795
Recurring regression	.95	0.4	299
Transient outage	.35	0.5	770
Latency shift (ρ_t^{lat})	.15	0.2	5,904
Topic change, hard	.75	0.8	62
Topic change, gradual	.95	1.2	1,577
Language change, hard	1.00	0.2	234
Language change, gradual	1.00	0.0	1,802

Table 10: Detector-only diagnostic for AdaptiveShift-SWTS (λ_{PH}=12) at k=7 and T=10,000, using 20 stream randomizations per condition on balanced cross-domain streams. Trigger rate is the fraction of runs with a Page–Hinkley alarm after onset, false alarms count pre-onset alarms per 10,000 stationary steps, and delay is the median first-alarm step after onset. Soft residual responses are not counted. These rows characterize detection and are not outcome comparisons.

Language shift. The language case moves from English to German, Japanese, Arabic, Thai, and Swahili intent queries. The oracle falls from 0.838 to 0.803 at k=3 under the hard switch. The online gap moves from 0.074 to 0.096, while offline accuracy falls from 0.694 to 0.548 and its gap widens from 0.144 to 0.255. Frozen classifiers continue routing multilingual queries to an arm whose accuracy falls from 0.63 to 0.47, while online routers re-route within a few hundred steps. BARP avoids most of this failure because its policy concentrates on the strongest arm. Gradual-transition and hard-switch endpoints match within 0.005 at every pool size, which attributes the frozen-policy gap to the post-shift query-to-arm mapping.

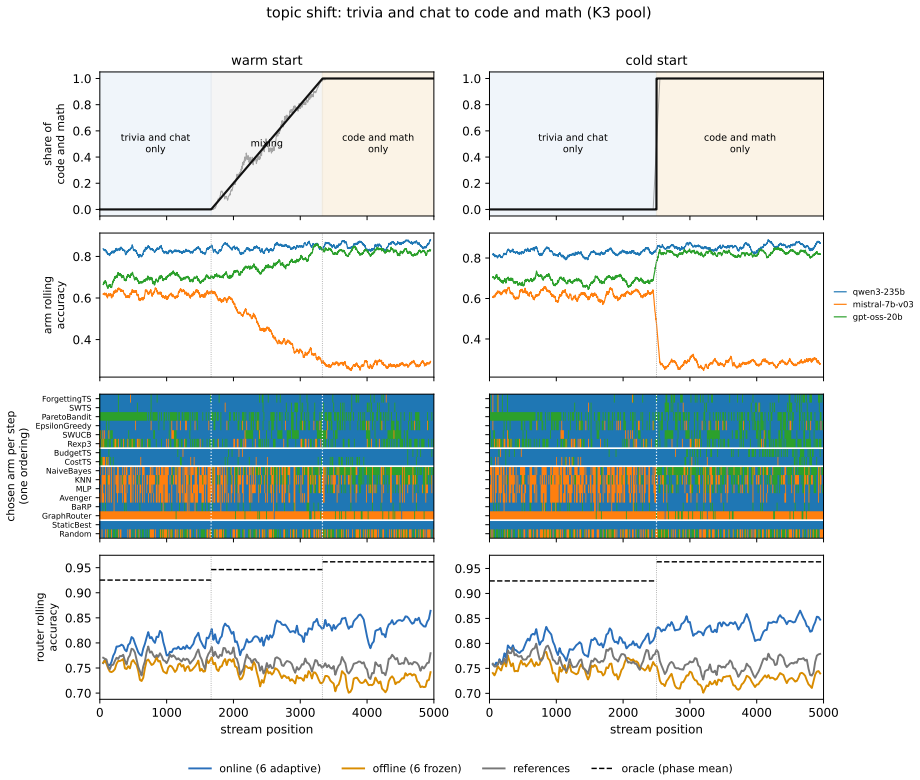


Figure 7: Topic shift on the k=3 pool, including the per-step selection raster.

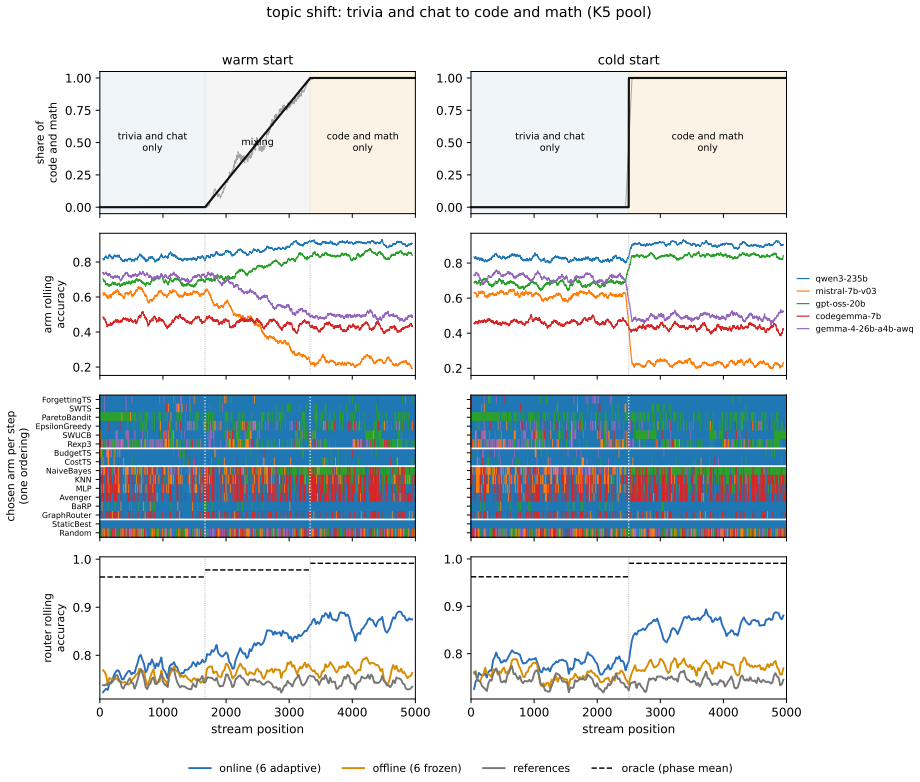


Figure 8: Topic shift on the k=5 pool, using the layout of Figure 7.

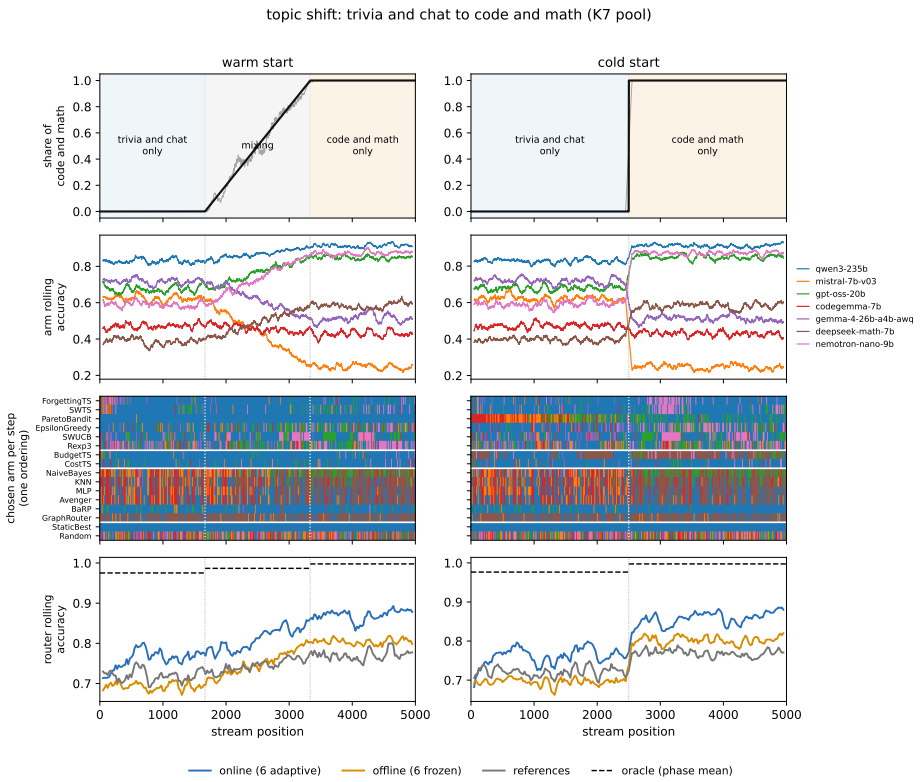


Figure 9: Topic shift on the k=7 pool, using the layout of Figure 7.

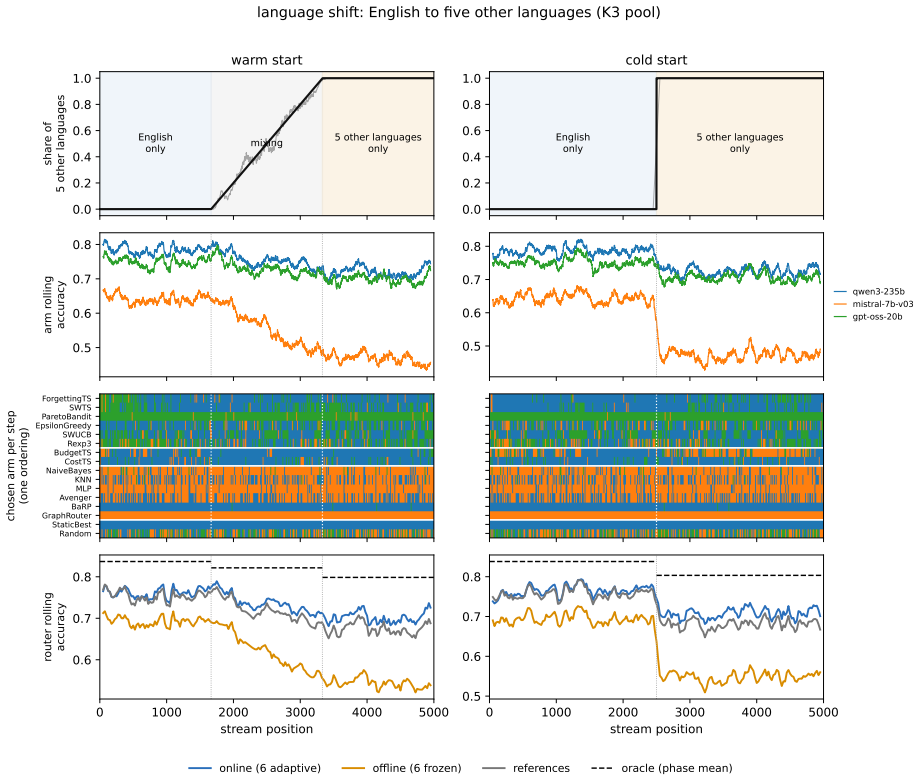


Figure 10: Language shift on the k=3 pool, including the per-step selection raster.

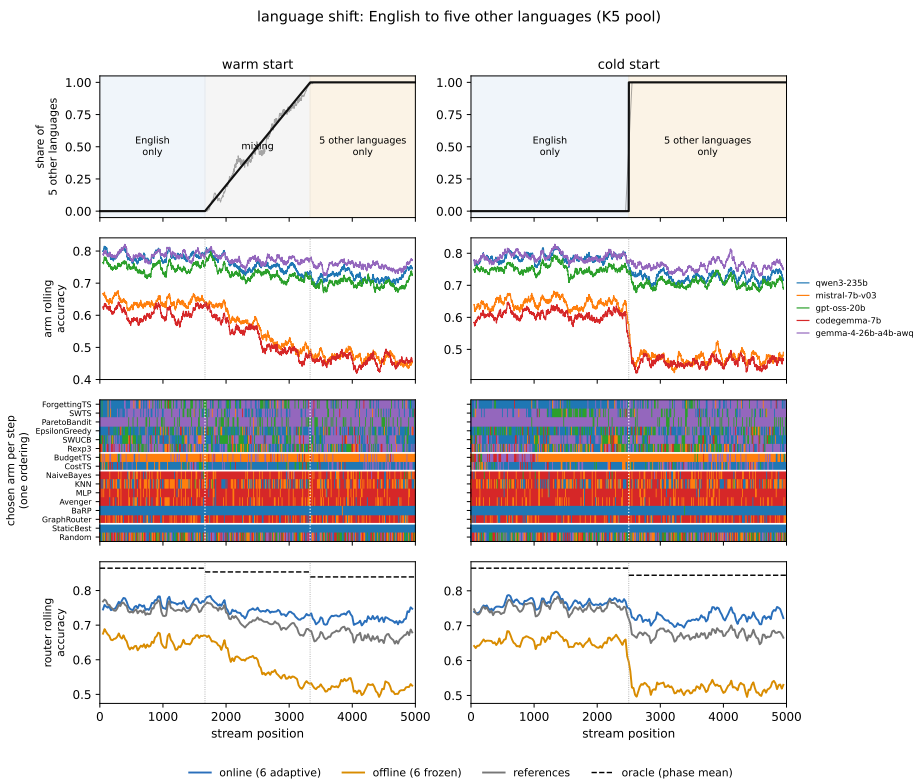


Figure 11: Language shift on the k=5 pool, using the layout of Figure 10.

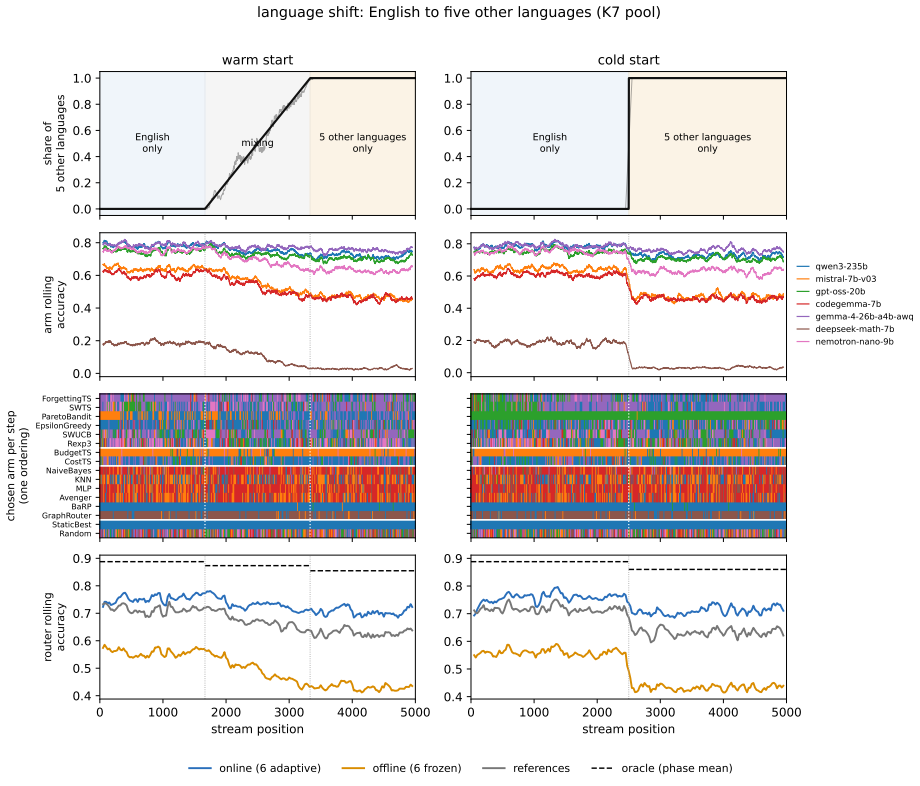


Figure 12: Language shift on the k=7 pool, using the layout of Figure 10.

I DATASETS, ROLLOUT PROMPTS, AND JUDGE PROTOCOL

Table 11 lists the 18 datasets in the combined corpus. The core portion uses source-ID deduplication. MASSIVE contributes a fixed 6,000-record test subset across six locales (FitzGerald et al., 2022). FEVER is assigned to Logic in this table and in every regenerated corpus figure. The release manifest records each dataset's source URL, exact license identifier, and whether the archive contains query text or a source identifier with rebuild instructions.

Category	Dataset	Records	Category	Dataset	Records
Knowledge	MMLU	14,042	Math	GSM8K	4,721
QA	OpenBookQA	5,957	Math	Hendrycks-MATH	4,686
QA	HotpotQA	5,143	Data An.	BIRD-SQL	4,322
Logic	FEVER	5,133	Data An.	TabMWP	973
QA	ARC-Challenge	1,093	Code	APPS	4,437
QA	TruthfulQA	817	Code	MBPP	974
Logic	HellaSwag	5,170	Chit-Chat	PersonaChat	5,009
Logic	WinoGrande	1,267	Tool Use	Gorilla	4,299
Multilingual intent classification	MASSIVE	6,000	Tool Use	MCPToolBench++	244
				Total	74,287

Table 11: Dataset categories and query-record counts. The 68,287 core records use source-ID deduplication, and MASSIVE contributes a fixed 6,000-record test subset.

Rollout prompts. Each candidate model receives a category-specific instruction. Multiple-choice prompts request one option letter. Math prompts request a final boxed answer. Code and SQL prompts request executable output. Tool-use prompts request one function call. Open QA and chit-chat preserve free-form answers. The release code contains the complete templates and field substitutions.

Judge protocol. The matrix stores one decoded response per query-model pair. DeepSeek R1 (Guo et al., 2025) scores open-ended responses with category-specific prompts and emits a tagged correct or incorrect verdict. This use follows empirical LLM-judge studies and routing evaluations (Zheng et al., 2023; Taberner-Miller, 2026). DeepSeek R1 is held outside the candidate arm pool. Parse failures receive a null label. Task-native parsing is used for deterministic tasks. The automated audit recomputes 182,252 task-native cells, with 166,901 agreements and 15,351 disagreements. Disagreements are resolved by assigning the ground-truth, pre-labeled answer. The release records the scoring route, judge checkpoint, prompt version, decoding parameters, and maximum output length for each cell.

J ARM STATISTICS

The arm tables report per-dataset accuracy and per-arm operational statistics over the stored outcome matrix. Because missing labels differ by arm, all-row arm means are descriptive; the matched-row column compares all seven arms on the same 69,680 complete rows.

K REPRODUCIBILITY AND RELEASE

The release, planned upon acceptance, contains the code that reproduces the outcome labels for the 74,287 queries and all results: scoring, stream construction, sixteen router implementations, result generation, and 261 tests, together with prompts, judge settings, source-ID splits, stream specifications, and stored seeds. The seven-model upper bound is 520,009 query-model cells. Queries are referenced by stable source identifiers, and rebuild scripts recover text and labels from the public source datasets. The final manifest records all 18 dataset licenses, sample identifiers, and the exact serving configuration for each model.

category	dataset	Qwen3	GPT-OSS	Gemma-4	Nemotron	Mistral	CodeG	DS-Math
chit-chat	persona-chat	**1.00**	0.58	0.99	0.97	0.93	0.65	0.22
code	apps	0.67	**0.76**	0.41	0.73	0.16	0.29	0.25
	mbpp	**0.89**	0.64	0.88	0.86	0.46	0.59	0.39
da	bird-sql	**0.83**	0.77	0.74	0.67	0.54	0.65	0.48
	tabmwp	0.81	0.94	0.62	**0.96**	0.50	0.51	0.65
knowledge	mmlu	**0.86**	0.64	0.71	0.53	0.56	0.38	0.35
logic	fever	0.71	0.40	**0.80**	0.32	0.65	0.56	0.48
	hellaswag	**0.95**	0.81	0.62	0.70	0.66	0.39	0.38
	winogrande	**0.89**	0.54	0.85	0.77	0.58	0.49	0.24
math	gsm8k	0.94	0.87	0.35	**0.95**	0.20	0.59	0.90
	hendrycks-math	**0.96**	0.87	0.51	0.86	0.11	0.18	0.34
qa	arc_challenge	**0.96**	0.93	0.44	0.91	0.72	0.61	0.66
	hotpotqa	0.70	**0.91**	0.49	0.83	0.65	0.60	0.49
	openbookqa	**0.94**	0.83	0.92	0.82	0.67	0.49	0.35
	truthfulqa	**0.75**	0.47	0.70	0.63	0.62	0.37	0.19
tool	gorilla_func	**0.72**	0.19	0.40	0.34	0.39	0.59	0.21
	mcptoolbenchpp	**0.57**	0.03	0.55	0.18	0.12	0.13	0.01
overall		**0.85**	0.69	0.67	0.69	0.52	0.47	0.40

Table 12: Mean accuracy of each arm per dataset over the reward matrix. Cell means are computed over the rows where the arm carries a correctness label; the Qwen3-235B column merges the directly labeled rows with the substrate-provided outcomes for the remainder. Bold marks the best arm per dataset.

arm	mean acc.	med. latency (ms)	med. output tokens
Qwen3	0.850	2,468	183
GPT-OSS	0.691	3,447	233
Gemma-4	0.665	14,924	273
Nemotron	0.689	10,253	384
Mistral	0.525	601	40
CodeG	0.474	743	60
DS-Math	0.398	502	10

Table 13: Per-arm operational statistics over the full matrix: mean accuracy, and medians of the recorded per-call latency and output token count.

The outcome matrix mixes hosted and self-hosted serving. Local models were served sequentially on one NVIDIA A10G with 24 GB memory, while labeling used up to 16 concurrent requests. Gemma uses the third-party `cyankiwi/gemma-4-26B-A4B-it-AWQ-4bit` checkpoint. Mistral uses `mistralai/Mistral-7B-Instruct-v0.3` (Mistral AI, 2024), CodeGemma uses `google/codegemma-7b-it`, DeepSeek-Math uses `deepseek-ai/deepseek-math-7b-instruct`, GPT-OSS uses `openai/gpt-oss-20b`, and Nemotron uses `nvidia/NVIDIA-Nemotron-Nano-9B-v2`. Qwen uses `Qwen3-235B-A22B-Instruct-2507` through a hosted endpoint. Local generation used a maximum of 384 output tokens. The release manifest records the remaining decoding settings and hosted endpoint regions.

Latency is measured under these serving configurations. Injected within-arm slowdowns isolate temporal latency changes from baseline serving differences.

L Use of Large Language Models

LLM-based assistants were used during manuscript preparation for editing passes, LaTeX assembly, and plotting scripting under author direction. All experimental designs, results, and claims were specified, executed, or verified by the authors, and every number in the paper traces to a versioned result file in the planned release.