Computer Life
ISSN: 1819-4818 | Vol. 15, No. 1, 2026

Probe-Enhanced Drift-Aware Model Updating for Compute-Efficient Sequential Recommendation

Zoe Huang

School of Computing and Information, University of Pittsburgh, PA, USA
Email: 394040530@qq.com

Abstract: Sequential recommenders must adapt to new interactions without repeatedly paying the cost of full-history retraining. Strict drift-triggered updating appears attractive, but can inadvertently suppress useful learning when marginal item frequencies are stable. We develop DAGR-P, a probe-enhanced update scheduler that performs a one-epoch new-data probe when a permutation drift screen is negative and allocates a longer, drift-dependent fine-tuning or recency-replay update otherwise. Candidate deployment follows a paired per-user validation rule. We implement the recency-replay update and the paired admission gate of the SPROUT compute-aware framework, and examine an additional user-matched drift detector. Five independent random seeds are evaluated on each of three explicitly synthetic, globally ordered interaction processes (22,680 interactions, 420 users, and 144 items per seed). Under stationary and abrupt processes, DAGR-P improves over a strict-gate policy from 0.1756 to 0.1857 and from 0.2540 to 0.2588 NDCG@10, respectively. Under gradual drift it obtains 0.2773, compared with 0.2689 for full retraining, with 32.93M versus 118.39M normalized training-score equivalents. A strong new-only updater remains more accurate in all three processes. The study identifies when low-budget probes counteract missed updates, and also shows why matched-user drift tests can be overly conservative. These results support a transparent compute–accuracy analysis, not a claim of field-scale effectiveness.

Keywords: Sequential recommendation; Concept drift; Continual learning; Adaptive training budget; Recency replay; Update admission.

1. Introduction

Sequential recommendation predicts the next item from a time-ordered interaction history. Self-attention methods such as SASRec and BERT4Rec established strong neural sequence encoders [1, 2], after earlier recurrent and convolutional approaches demonstrated the importance of order-sensitive item representations [3, 4, 5]. Time-aware attention and self-supervised sequence objectives subsequently improved representation learning [6, 7, 8]. These developments primarily concern which prediction function to train. A separately important operational question is when that function should be updated after deployment.

That question is difficult because serving data are not drawn from a fixed distribution. Temporal collaborative filtering established that user preferences evolve [9], and evaluation protocol itself can alter the apparent ordering of recommendation methods [10]. Broader work on concept drift characterizes changes in the data-generating process and tests for their occurrence [11]. Adaptive-window detection [12] and streaming recommenders [13] make update timing a first-class design choice. Nevertheless, the update decision is often reduced to a fixed schedule. Such a schedule spends compute during stable periods and can mix obsolete evidence into the next model after a large shift.

Retaining past experience can be valuable but is not unconditionally benign. Sequential meta-learning addresses costly full-history retraining [14], and exemplar replay with distillation reduces forgetting in changing recommendation sessions [15]. Experience replay is a general stability mechanism [16], while regularizing knowledge transfer can discourage forgetting [17]. Conversely, warm starts sometimes generalize less well than fresh optimization [18]. An attractive updater must therefore decide whether to fit new parameters, which examples to use, and whether the proposed update is sufficiently beneficial to deploy.

We study a probe-enhanced policy, DAGR-P, which decouples update intensity from whether item-group drift is statistically detectable. It makes a one-epoch new-only probe when drift is inconspicuous; substantial detected drift receives a longer new-only update, while milder but significant drift activates recency-aware replay. A paired validation gate decides whether a trained candidate should replace the incumbent. This responds to an observed weakness of a strict drift-only screen: a stationary marginal distribution can coexist with substantial room for improved ranking from newly observed interactions.

The contributions are: (i) a budgeted probe-and-replay controller that retains low-cost learning in apparently stable periods; (ii) an implemented user-matched permutation diagnostic separating cohort turnover from population-level changes; and (iii) independent-seed tests against both full retraining and a strong new-only fine-tuning baseline, with cost charged to rejected candidates. These are controlled simulation findings; the reported work does not use observational recommendation logs.

2. Related work and design position

2.1. Temporal recommendation and stability

State-of-the-art attention architectures are capable of modeling substantially richer dependencies than a first-order predictor. Their ability to exploit a large training corpus, however, does not establish that retraining them on every new block is budget-optimal. Under nonstationarity, the model update is a control policy superimposed on the encoder. This distinction motivates an intentionally compact ranker: we can compare update strategies while keeping model architecture,

evaluation candidates, and scoring budget constant. Dual-process memory-augmented sequential recommendation provides a useful example of exploiting evolving interaction histories while evaluating recommendation explanations [27].

The costs of data, capacity, and model refresh are related but distinct. Compute-optimal analyses in language modeling [19] show that parameter count alone is not a meaningful measure of training efficiency. The Transformer architecture [20], network widening by function-preserving transfer [21], and model-reuse schemes for Transformer pretraining [22] offer ways to alter representation capacity. Our experiment does not enlarge a network; instead it allocates training effort across refresh times, making update frequency the decision variable.

2.2. Choosing when and what to replay

A drift detector based on category frequencies is cheap to evaluate before any gradient step. However, an alert is not itself evidence that optimization will improve ranking. DAGR-P therefore distinguishes a data-distribution test from a paired utility test. Replay is also conditional rather than automatic: a large measured shift makes the old-history gradient less representative of the incoming period, whereas milder shifts permit rehearsal as a hedge against forgetting. A one-epoch probe is added to prevent the strong marginal-stationarity assumption from suppressing all learning. The magnitude threshold is fixed from exploratory runs on separate seeds, and the probe extension is assessed on disjoint seeds.

3. Streaming task and backbone

3.1. Time-ordered evaluation

Let $D=\{(u_t,i_t,b_t)\}$ denote an interaction stream with users u, item identifiers i, and globally ordered block labels b. We use six equally sized blocks B0,…,B5. At stage s=1, a cold-start model is fitted on the first 90% of B0, validated on the remaining 10%, and frozen while B1 is served. For $s\geq 2$, the last 10% of $B_{(s-1)}$ serves as validation, the prefix of $B_{(s-1)}$ is available for training, and the resulting deployed model serves B_s. Previously held-out interactions may enter subsequent updates once they have become historical. The candidate never learns the labels of its current validation tail.

During serving, each target can condition on earlier interactions of its user, including earlier events in the same served block, but network weights remain fixed until the next stage. The evaluation ranks all 144 catalog items, including previously consumed items; repeated consumption is a legitimate target in this simulator. NDCG@10 and HR@10 are averaged across a user's served targets and then across users, so a very active user does not dominate the score. The primary endpoint is the arithmetic mean across stages 2–5; stage 1 is a common-start stage included in every reported cumulative training cost.

3.2. Factorized sequential ranker

For a target at time t, let $H_{(u,t)}$ contain up to five previous item identifiers, with zero padding. A width-24 ranker embeds the user, latest item, and short context. With separate context and output item embeddings, the context vector and item logit are

$$h_{(u,t)} = e_u + 0.7\, e_{last} + 0.3\, \mathrm{mean}(e_j : j \in H_{(u,t)}),$$

$$\mathrm{score}(i \mid u,t) = h_{(u,t)}^{\top} v_i + b_i .$$

A full-catalog softmax cross-entropy updates every embedding and item bias. The training configuration is Adam with learning rate 0.007, weight decay 10^{-5}, batch size 256, five cold-start/full-retraining epochs, and three warm-update epochs [23]. The architecture and optimizer settings are identical for all policies. Evaluation is full-catalog rather than sampled ranking. The simplicity of the model makes the simulation a policy test rather than a claim about Transformer-scale recommendation quality.

4. Budgeted drift-aware updating

4.1. Population-level drift detection

For completed blocks $B_{(s-2)}$ and $B_{(s-1)}$, the controller constructs 12-dimensional item-group frequency vectors p and q. With base-two Jensen–Shannon divergence $D_s=JS_2(p,q)$ [24], it uses 79 random pooled-label permutations to estimate $p_s=(1+\#\{D_{perm} \geq D_s\})/80$. The drift screen activates when $p_s \leq 0.05$. This is an inexpensive test of observed marginal turnover, not a direct test of whether optimizing new model parameters will improve next-block ranking. At decision s, both blocks are already complete; the new candidate is evaluated only on the subsequent served block.

The rule does not require drift to be abrupt. A significant small-magnitude shift and a significant large-magnitude shift lead to distinct example sets. We retain $D_s=0.015$ bits as the preselected magnitude boundary. It was chosen on exploratory seeds 11, 17, and 23 before any analysis with the five evaluation seeds. A 79-permutation p-value has a minimum resolution of 0.0125 and rejects at $p\leq 0.05$ when at most three draws exceed the observation.

4.2. Adaptive replay and low-cost probes

A statistically significant shift with $D_s>0.015$ uses only newly eligible examples and three training epochs. If the shift is significant but smaller, the candidate instead trains for three epochs on a recency replay stream: users active in the latest eligible slice each provide up to 20 recent target positions, supplemented by at most $\rho=0.5$ times as many inactive users, sampled without replacement using weights γ^a with $\gamma=0.5$ and inactivity age a measured in blocks. All valid historical positions in the sampled user windows contribute to training loss.

We implement the active-user windows, all-position supervision, exponentially decayed inactive-user sampling, and warm-start with a fresh Adam state from the SPROUT replay update of Shen [25]. Our policy retains these specific operations while allocating replay only in the mild-drift branch. The same replay implementation runs in ReplayPeriodic, a fixed-schedule comparison, allowing its behavior and measured work to be checked independently of the adaptive controller.

When the drift screen is negative, DAGR-P nevertheless fits a one-epoch new-only candidate. This is the probe modification relative to strict DAGR: it acknowledges that a model may still learn from data whose marginal popularity distribution looks unchanged. In this branch the amount of training is one-third of the standard three-epoch warm update. Every candidate incurs its fitting cost, whether later accepted or rejected. The design separates the expensive choice of several epochs and replay from the simpler decision to continue learning at all.

4.3. C. Paired deployment admission and compute accounting

For each validation user u, let δ_u=NDCG@10_u(candidate)−NDCG@10_u(incumbent), evaluated on the same held-out interactions. A candidate is deployed only when $\bar{\delta}$>sd(δ_u)/√n. This adapts the paired one-standard-error gate used by Shen for model-capacity proposals to the different decision of whether to admit a model refresh [25]. The gate is a decision heuristic; it is not an equivalently calibrated two-sided 95% hypothesis test. The training work of rejected candidates remains in the cumulative totals.

We define an exact, fixed-architecture proxy for full-catalog scoring work, C=3|I|Σ_a n_a, where n_a is the number of optimizer examples processed by training attempt a, |I|=144, and the factor three approximates forward plus backward output-layer scoring. Costs are stated in millions of score equivalents (M), not processor FLOPs. The proxy excludes per-batch embedding and history computation, category permutation tests, and full-catalog validation inference; measured optimizer-only CPU seconds and an independent drift-detector profile are recorded separately.

4.4. User-matched alternative and algorithm

An additional diagnostic compares the same users across two completed blocks. For each user present in both, we form a normalized 12-group histogram in each block; group probabilities are averaged over shared users before computing Jensen–Shannon divergence. The reference distribution is obtained by 79 random sign flips that swap the two block histograms at the user level. The null is paired block-label exchangeability. This removes the direct change in cohort membership but can miss learning opportunities arising from repeated observations rather than a change in conditional preference.

ALGORITHM 1. DAGR-P UPDATE AT SERVING STAGE s

Input: deployed model f; completed blocks B_(s−2), B_(s−1)

1: D,p ← group-JS divergence and 79-permutation p-value

2: if p > 0.05: R ← newly eligible data; e ← 1

3: else if D > 0.015: R ← newly eligible data; e ← 3

4: else: R ← recency-aware active/inactive replay; e ← 3

5: f_candidate ← Adam(f, R, e epochs; fresh optimizer)

6: charge all processed examples, even if candidate rejected

7: δ_u ← paired held-out user NDCG difference

8: if mean(δ_u) > SE(δ_u): deploy f_candidate

9: else: preserve f; serve the subsequent block

The final 10% of completed B_(s−1) is held out for candidate comparison, and new training indices end before that validation suffix. Interaction histories are causal: the input to each scored event contains earlier items only, even when earlier items occurred during the same frozen-serving block. The next block B_s is never used to select candidate weights or hyperparameters.

5. Experimental design

5.1. Simulation, data access, and model controls

The timestamped MovieLens benchmarks [26] are the natural external test and are left to future work; this study uses a controlled synthetic stream and contains no observational MovieLens or Amazon interactions. We use a deterministic synthetic stream with six globally ordered blocks B0–B5, 3,780 events per block, 420 users, and 144 items partitioned into 12 groups. Each block contains 210 active users with 18 actions per user, and activity persists from the previous block with an approximately 1.9-fold sampling weight. The complete generated arrays for every final seed are stored alongside the code.

Within each user, next-group logits combine an enduring Gaussian-drawn user-category affinity, a transition matrix biased toward the previous category, and a block-specific group offset. An item is then sampled from its category using item-popularity and user–item preference terms. The stationary setting holds offsets constant; the gradual setting increases an opposing-group contrast linearly across blocks; the abrupt setting makes a persistent large contrast change after B2. The histories used by the predictor contain the five preceding items. These mechanisms produce interpretable next-item dependence but omit exposure bias, item arrival, timestamps within a block, and other properties of industrial logs. Self-evolving software agents with dual memory further illustrate a practical mechanism for preserving prior feedback and directing subsequent updates toward informative cases [28].

TABLE I. Generator and update protocol

Characteristic	Value / implementation
Interaction log	22,680 events × five seeds × three settings
Catalog and users	144 items, 12 groups; 420 users
Serving schedule	B1 first; B2–B5 primary mean
New updates	3 epochs, or 1 for a negative-screen probe
Drift test	79 permutations; α=0.05; JS cut 0.015
User replay	20 targets; ρ=0.5; γ=0.5
Validation	Latest completed-block 10% suffix
Primary metrics	User-macro NDCG@10 and HR@10

5.2. Compared update strategies and independent seeds

Frozen deploys the shared initial model without refresh. FullRetrain trains a fresh ranker for five epochs on all currently eligible targets. NewOnly trains for three epochs on newly eligible targets at every stage. Probe1NewOnly performs the same fixed update with a single epoch. ReplayPeriodic applies the SPROUT recency-replay update of [25] at every stage. Strict DAGR skips all optimizer work on a negative drift screen and otherwise uses adaptive three-epoch training with paired admission. DAGR-P changes only the negative-screen branch to a one-epoch probe. DAGR-P without the paired gate deploys all probe and update candidates. Matched DAGR uses the shared-user permutation detector without probes, isolating detector specificity.

The independent confirmation of DAGR-P uses seeds 73, 79, 83, 89, and 97. Separate exploratory seeds (53, 59, 61, 67, and 71) were used for feasibility checks and do not enter the five-seed statistics. Each policy sees the same chronological stream and identical cold-start checkpoint for a given seed, and matching update types use the same optimizer shuffles. The outcomes aggregate equally over serving stages 2–5, then equally over independent generator seeds. Uncertainty for paired policy differences uses a two-sided 95% Student-t

interval of the five seed-level differences.

Optimization uses a width-24 factorized user/context/item embedding model with full-catalog cross-entropy, Adam at learning rate 0.007 and batch size 256, with the same architecture and cold-start checkpoint for every strategy. This is not a Transformer and its workload estimates cannot be extrapolated into large-model FLOPs. Separate code files implement drift tests, replay sampling, training loops, all compared policies, and offline figure production.

6. Results and ablation

6.1. Main accuracy–work trade-offs

Table II presents the three principal alternatives in the independent five-seed experiment; Fig. 1 also includes the strict-gate comparator. Under gradual drift, DAGR-P reaches 0.2773 NDCG@10 versus 0.2689 for FullRetrain, with 32.93M rather than 118.39M normalized scoring work (72.2% less). Under abrupt drift, its 0.2588 compares with 0.2481 for FullRetrain, using 24.40M (79.4% less). Under the stationary process, it records 0.1857 versus 0.1908 for FullRetrain, using 25.90M rather than 118.39M (78.1% less). The apparent savings refer strictly to the stated training proxy and not total CPU service time.

TABLE II. Final five-seed ndcg@10 / hr@10 / training cost (m)

Setting	Policy	NDCG	HR	Cost
Stat	FullRetrain	0.1908	0.3719	118.39
	NewOnly	0.1922	0.3671	26.94
	DAGR-P	0.1857	0.3533	25.90
Grad	FullRetrain	0.2689	0.5176	118.39
	NewOnly	0.2788	0.5323	26.94
	DAGR-P	0.2773	0.5285	32.93
Abru	FullRetrain	0.2481	0.4815	118.39
	NewOnly	0.2639	0.5016	26.94
	DAGR-P	0.2588	0.4922	24.40

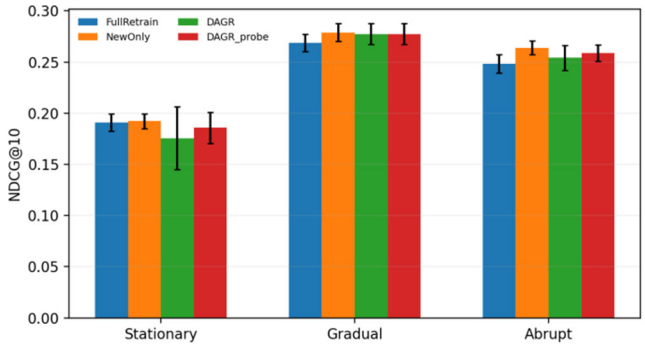


Fig. 1 Held-out seed-averaged NDCG@10 (stages 2–5); error bars are 95% t intervals over five seeds.

NewOnly remains the strongest accuracy baseline on all three processes: 0.1922, 0.2788, and 0.2639 NDCG@10, respectively. It also requires less normalized training work than DAGR-P in gradual drift. Thus the adaptive method is not universally Pareto optimal. The empirical contribution is understanding the consequences of different drift decisions under an explicitly measured budget, including cases where a simple fixed policy is preferable.

6.2. Effect of low-cost probes

With the strict negative-screen skip, DAGR trains 9 of 20 candidate opportunities in stationarity and 11 of 20 under abrupt drift. DAGR-P trains all 20 opportunities, but 11 and 9 of them, respectively, use only one epoch. Its mean NDCG@10 rises from 0.1756 to 0.1857 in the stationary process and from 0.2540 to 0.2588 in abrupt drift, while cost increases from 22.31M to 25.90M and from 21.46M to 24.40M. For gradual drift, all screens are positive; the two algorithms coincide exactly at 0.2773 and 32.93M.

The paired 95% difference intervals show how tentative the gains remain. Relative to DAGR, DAGR-P gains 0.0100±0.0160 under stationarity and 0.0047±0.0053 under abrupt drift. Neither interval excludes zero. Relative to the constant one-epoch baseline, the DAGR-P differences are 0.0097±0.0074, 0.0122±0.0084, and 0.0121±0.0119 in stationary, gradual, and abrupt order. The first two intervals exclude zero, at the cost of additional training examples. These comparisons support a budget–accuracy interpolation rather than blanket dominance.

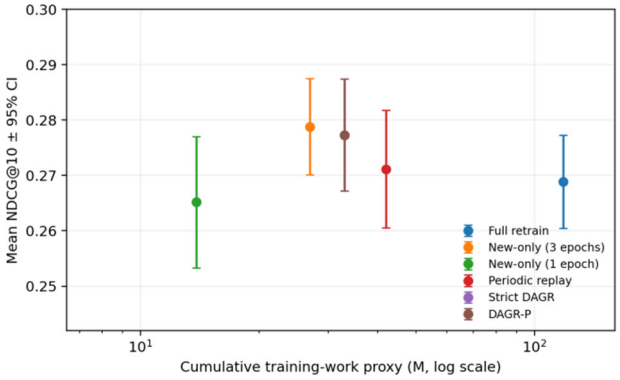


Fig. 2 Gradual-drift accuracy versus cumulative training work for six policies; vertical bars give seed-level 95% intervals.

6.3. Detector specificity versus update utility

Table III isolates the strict marginal detector, the matched-user detector, and the probe/paired-gate choices. The matched detector trains zero candidates in stationary streams, compared with nine for strict DAGR, but its NDCG@10 collapses to the Frozen score of 0.1445. Under abrupt drift, it trains five candidates but obtains only 0.2364, below the 0.2540 of the marginal screen and 0.2588 of DAGR-P. Removing cohort-turnover alerts improves marginal specificity while losing useful updates that accumulate knowledge without an obvious marginal change. A nonsignificant paired-histogram difference is therefore insufficient evidence that the recommender is already optimal.

TABLE III. Component check: mean ndcg@10 / cost (m)

Setting	Ablation	NDCG	Cost
Stat	Strict screen	0.1756	22.31
	Matched screen	0.1445	7.35
	Probe + paired	0.1857	25.90
	Probe, no paired	0.1867	25.90
Grad	Strict screen	0.2773	32.93
	Matched screen	0.2773	32.01
	Probe + paired	0.2773	32.93
	Probe, no paired	0.2779	32.93
Abru	Strict screen	0.2540	21.46
	Matched screen	0.2364	12.25
	Probe + paired	0.2588	24.40
	Probe, no paired	0.2590	24.40

Disabling the paired admission rule slightly raises NDCG in all settings (0.1867 versus 0.1857 in stationary, 0.2779 versus 0.2773 in gradual, and 0.2590 versus 0.2588 in abrupt),

without reducing training work. This indicates that conservative candidate rejection can miss small improvements. Because this gate is paired over users, it can still be useful when noisy candidates regress, but the current generator provides limited evidence of a net quality advantage. The exact training and deployment decisions are preserved per seed and stage.

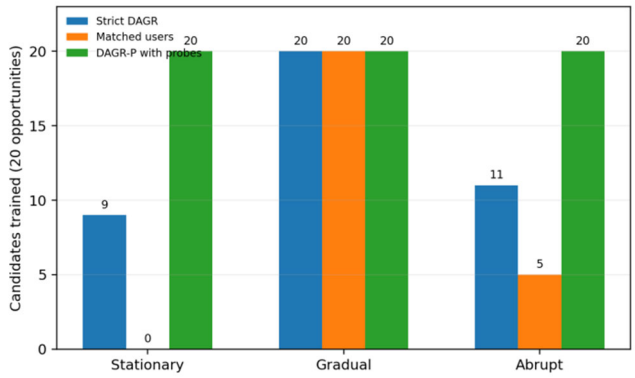


Fig. 3 Total training attempts out of 20 opportunities per process: strict drift screen, matched-user screen, and probe-enhanced policy.

6.4. Temporal causality and cost caveats

The abrupt preference switch occurs at the start of B3, so a model frozen through B3 necessarily encounters it before an update can be trained in response. The detector may use completed B3 to choose the candidate for B4, not retroactively for B3. All reported stage predictions respect that delay. The work proxy counts rejected candidates, yet excludes the controller itself. On the same one-thread CPU used for profiling, a marginal 79-permutation test had median 0.0084 s per stage call across 60 observed decisions, and the matched test had median 0.0041 s. These are detector measurements, not end-to-end per-stage timings, and hardware-dependent timing claims are not used to substitute for objective FLOP measurements.

6.5. Stage-resolved evidence

Aggregate accuracy can conceal a failure to respond at the correct time. Table IV therefore reports the four evaluated serving stages for the abrupt scenario. All entries are measured five-seed means from the saved stage_metrics.csv file; they are not projections from the three-process averages. Stage 2 precedes the designed distribution shift, while the first affected block B3 is evaluated before any method can retrain using events from B3. A causal adaptation to that block can only begin at stage 4. The table illustrates that DAGR-P does not outperform NewOnly immediately after the switch, although it improves over the strict-gate DAGR at the first affected serving stage.

TABLE IV. Abrupt scenario: mean ndcg@10 by serving stage

Stage	Strict	DAGR-P	New	Retrain
2	0.1733	0.1759	0.1778	0.1827
3	0.2333	0.2451	0.2593	0.2327
4	0.3028	0.3045	0.3072	0.2826
5	0.3067	0.3096	0.3115	0.2945

At stage 3 the strict controller averages 0.2333 NDCG while the probe controller reaches 0.2451; NewOnly reaches 0.2593. This difference is important for interpretation: strict DAGR has skipped inexpensive intermediate learning during apparently stable blocks, leaving it less prepared for an unexpected next-period shift. After B3 is observed, the large group-divergence statistic at stage 4 is 0.3483 bits on average and is significant in all five seeds. The resulting three-epoch update can now exploit newly available post-switch training examples. That is why both adaptive controllers improve between stages 3 and 4.

The group divergence is also informative about what is not measured. In the stationary process, mean observed values range only from 0.00207 to 0.00263 bits across stage decisions, but the global permutation detector declares significant change in nine of 20 cases. It is sensitive to the activity-weighted mixture of users and to within-user event dependence, rather than only to the constructed stationary generator parameters. The matched-user sign-flip variant is more conservative in this process, yet such conservatism does not imply stable optimal model weights. These results motivate retaining a small learning budget rather than equating a negative drift test with optimal prediction.

6.6. Auditability and reproducibility checks

The experiments include explicit records of all comparison candidates, accepted deployments, training-example counts, and the stage-specific significance statistics. Each seed generates a unique saved NPZ stream, including global block boundaries and the five-item causal history for every event. Stage results are evaluated with a fixed model throughout each entire served block; predictions may use earlier observed user events in the same block but not later events. Every baseline begins with the same cold-start weights. Training is counted for failed or rejected candidates; no method is credited with a rejected checkpoint at zero computational cost.

Per-seed paired intervals prevent an artificially narrow confidence interval from treating thousands of correlated interactions as independent experiments.

7. Discussion and limitations

The central observation is that the information needed to diagnose dataset shift differs from the information needed to assess the benefit of retraining. The matched-user screen answers a cleaner descriptive question about whether a stable cohort changed its group distribution, but its negative verdict often accompanies ongoing model improvement from newly accumulated interactions. By contrast, the one-epoch probe retains this learning channel at bounded cost. A validation admission gate adds a second decision that asks whether the estimated gain exceeds the user-level empirical noise, although the one-SE rule is conservative and in these simulations sometimes rejects a useful candidate.

The recency-aware replay mechanism is operationally faithful to the imported stream construction, but old targets do not dominate this generator: a new-only updater performs especially well because active users contribute 18 new interactions each update, item identifiers are fixed, and the compact ranker retains much of its past knowledge through warm starts. A weaker new-data supply, cold-start inventory, recurring dormant cohorts, or longer interaction gaps could alter this ranking of methods. Such settings were not verified here and must be tested rather than presumed to favor replay.

The study uses a controlled synthetic generator with item groups assumed known, a small fixed catalog, a lightweight factorized encoder, full-catalog scoring, and just five confirmatory seeds. It has no observational benchmark, no production serving workload, and no evidence for memory or power savings at Transformer scale. The 0.015 threshold was

selected in pilot simulations, the probes were designed after earlier exploratory runs, and the current independent seeds verify only the specified synthetic regimes. These limits narrow the claims to reproducible controller behavior under those regimes.

7.1. Separating three sources of distribution change

The marginal observed frequency at time b may be written as p_b(i)=Σ_u P_b(u)P_b(i|u), which can change when the active-user mixture P_b(u) changes even if each conditional preference P_b(i|u) remains constant. A shared-user comparison partially isolates the second factor by conditioning on repeated user identities, but it sacrifices evidence from newly active users and cannot address improvements from simply observing more events. More importantly, a ranking learner is judged by future conditional prediction loss, not by the p-value of an input distribution test. Therefore, no simple identity between statistical detection and the optimal refresh schedule should be assumed. The empirically preferred update can depend on label availability, history length, representation underfitting, and the costs of candidate optimization and validation.

7.2. Compute normalization and decision overhead

The proxy C is invariant to optimizer implementation for a fixed model and identical example sequence, but it does not compare architectures with different sequence encoders, sampled-softmax designs, or sparse parameter updates. Actual FLOP accounting would include the context embedding arithmetic, reduction and gradient scatter, optimizer state updates, and repeated validation scoring. Likewise, full wall-clock cost includes permutation detection, control logic, checkpoint copying, and model selection. The reported detector timing measurements establish only that those calls were executed and profiled on the declared one-thread CPU; their sizes and ranking relative to training could differ on a GPU. Future work must measure end-to-end energy and refresh latency before using the phrase compute savings in a deployment claim.

8. Conclusion

DAGR-P augments a drift-screened sequential recommender with minimal updates even during apparent stationarity, while retaining adaptive replay and user-paired candidate admission for detected changes. Independent synthetic-stream tests show a measurable cost–accuracy trade-off: probes reduce the accuracy deficit caused by strict refresh skipping, but a strong fixed new-only updater remains difficult to surpass. A matched-user detector further illustrates that greater drift specificity can harm predictive learning when misused as an update veto. The next meaningful validation is a timestamped observational benchmark with end-to-end compute profiling and longer inactive-user cycles.

References

[1] Kang, W.-C., & McAuley, J. (2018). Self-attentive sequential recommendation. In Proceedings of the IEEE International Conference on Data Mining (ICDM) (pp. 197-206). IEEE. https://doi.org/10.1109/ICDM.2018.00035

[2] Teng, D. (2025). TRIAD: Deadline- and reliability-aware scheduling for post-earthquake structural assessment workflows. World Journal of Engineering Research, 3(6), 55-66.

[3] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In Proceedings of the International Conference on Learning Representations (ICLR). arXiv, arXiv:1511.06939. https://doi.org/10.48550/arXiv.1511.06939

[4] Tang, J., & Wang, K. (2018). Personalized top-N sequential recommendation via convolutional sequence embedding. In Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM) (pp. 565-573). ACM. https://doi.org/10.1145/3159652.3159656

[5] Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., & He, X. (2019). A simple convolutional generative network for next item recommendation. In Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM) (pp. 582-590). ACM. https://doi.org/10.1145/3289600.3290975

[6] Li, J., Wang, Y., & McAuley, J. (2020). Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM) (pp. 322-330). ACM. https://doi.org/10.1145/3336191.3371786

[7] Zhou, K., Wang, H., Zhao, W. X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., & Wen, J.-R. (2020). S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM) (pp. 1893-1902). ACM. https://doi.org/10.1145/3340531.3411954

[8] Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., & Cui, B. (2022). Contrastive learning for sequential recommendation. In Proceedings of the IEEE 38th International Conference on Data Engineering (ICDE) (pp. 1259-1273). IEEE. https://doi.org/10.1109/ICDE53745.2022.00099

[9] Koren, Y. (2009). Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (pp. 447-456). ACM. https://doi.org/10.1145/1557019.1557072

[10] Meng, Z., McCreadie, R., Macdonald, C., & Ounis, I. (2020). Exploring data splitting strategies for the evaluation of recommendation models. In Proceedings of the 14th ACM Conference on Recommender Systems (RecSys) (pp. 681-686). ACM. https://doi.org/10.1145/3383313.3418479

[11] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44. https://doi.org/10.1145/2523813

[12] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the SIAM International Conference on Data Mining (SDM) (pp. 443-448). SIAM. https://doi.org/10.1137/1.9781611972771.42

[13] Chang, S., Zhang, Y., Tang, J., Yin, D., Chang, Y., Hasegawa-Johnson, M. A., & Huang, T. S. (2017). Streaming recommender systems. In Proceedings of the 26th International Conference on World Wide Web (WWW) (pp. 381-389). ACM. https://doi.org/10.1145/3038912.3052627

[14] Zhang, Y., Yin, H., Chen, T., Zhang, C., Yu, B., Zhang, W., & Li, X. (2020). How to retrain recommender system? A sequential meta-learning method. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) (pp. 1479-1488). ACM. https://doi.org/10.1145/3397271.3401167

[15] Mi, F., Lin, X., & Faltings, B. (2020). ADER: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In Proceedings of the 14th ACM Conference on Recommender Systems (RecSys) (pp. 408-413). ACM. https://doi.org/10.1145/3383313.3412218

[16] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., & Wayne, G. (2019). Experience replay for continual learning. Advances in Neural Information Processing Systems, 32, 350-360.

[17] Li, Z., & Hoiem, D. (2018). Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12), 2935-2947. https://doi.org/10.1109/TPAMI.2017.2773081

[18] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34-48.

[19] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, B., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., … & Sifre, L. (2022). Training compute-optimal large language models. Advances in Neural Information Processing Systems, 35, 30016-30030.

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008.

[21] Chen, T., Goodfellow, I. J., & Shlens, J. (2016). Net2Net: Accelerating learning via knowledge transfer. In Proceedings of the International Conference on Learning Representations (ICLR).

[22] Chen, C., Zhong, Y., Guo, Z., Zhang, X., & Li, J. (2022). bert2BERT: Towards reusable pretrained language models. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 2134-2148). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.151

[23] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR). arXiv, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

[24] Lin, J. (1991). Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory, 37(1), 145-151. https://doi.org/10.1109/18.61115

[25] Shen, A. (2023). Compute-efficient scaling of large sequential recommendation models under dynamic user behavior. World Journal of Information Technology, 1(1), 62-71. https://doi.org/10.61784/wjit4126

[26] Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems, 5(4), Article 19. https://doi.org/10.1145/2827872

[27] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. Innovation and Technology Studies, 3(1), 27-41.

[28] Mo, T., Zhang, Y., Li, H., Wang, J., & Chen, X. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access, 14, 1-15. https://doi.org/10.1109/ACCESS.2026.3713401