ShiftBench: A Claim-Aligned Protocol for Stress-Testing Sequential Recommenders under Distribution Shift

Anonymous Author(s)

Abstract

Offline evaluations of sequential recommender systems usually sample train and test interactions from the same log. A high score can therefore conceal a model's dependence on transient popularity, exposure policy, or stable user preferences. We present *ShiftBench*, a claim-aligned protocol for asking which kinds of distribution shift a recommender can tolerate and at what cost. The protocol separates five shift operators—popularity reversal, preference drift, exposure censoring, catalog churn, and feedback delay—from the choice of model and metric. Each operator has an interpretable severity parameter, preserves a declared set of invariants, and is paired with a diagnostic that tests whether the transformation behaved as intended. We combine conventional ranking utility with robustness loss, worst-group utility, calibration error, and catalog coverage, and we require every reported claim to name its shift, severity range, metric, and population. A fully specified synthetic dry run shows why clean-test rankings are not reliable robustness rankings: the model with the best in-distribution NDCG need not minimize relative robustness loss. The dry run is illustrative rather than evidence about deployed systems; its purpose is to make every design choice auditable. ShiftBench offers a compact, reproducible starting point for evaluations whose conclusions match the uncertainties encountered after deployment.

CCS Concepts

• **Information systems** → **Recommender systems**; *Test collections*; • **Computing methodologies** → *Machine learning*.

Keywords

recommender systems, sequential recommendation, distribution shift, robustness, offline evaluation, reproducibility

1 Introduction

Sequential recommenders estimate what a user will choose next from an ordered interaction history. Progress is commonly measured by withholding the final interaction from a historical sequence and ranking it among sampled or complete catalog candidates. This protocol supports repeatable comparison, but it also quietly assumes that the mechanisms producing the past will continue to produce the future. In a deployed service, that assumption is fragile: a promotion can reverse item popularity, users can acquire new interests, the logging policy can hide relevant alternatives, and new items can enter before they have interaction histories.

These changes are not interchangeable. A model that is insensitive to a popularity reversal may still fail when preferences drift; a model that handles catalog churn may rely heavily on a biased exposure policy. Collapsing all changes into a single temporal test set makes a failure difficult to diagnose. It can also encourage a broad claim—"robust to distribution shift"—from a narrow experiment. Prior work has shown that recommendation evaluation depends on exposure and selection processes [7, 8], that standard offline protocols can produce inconsistent conclusions [2], and that accuracy alone misses dimensions such as calibration, diversity, and popularity bias [1, 9].

We introduce *ShiftBench*, a protocol organized around a simple principle: each empirical claim should identify the uncertainty it tests. Rather than prescribing a new recommender, ShiftBench specifies how to transform an evaluation environment, verify that transformation, and report the resulting trade-offs. The protocol makes four contributions:

(1) We define five independently controlled shift operators with explicit severity parameters, invariants, and diagnostic tests.
(2) We connect ranking utility to robustness loss, worst-group performance, calibration, and coverage in a single reporting schema.
(3) We propose a claim card that binds each conclusion to a shift, severity interval, metric, user population, uncertainty estimate, and abstention rule.
(4) We provide a fully specified synthetic dry run that demonstrates how to audit the protocol without presenting synthetic values as deployment evidence.

ShiftBench is deliberately model-agnostic. It can evaluate matrix-factorization baselines, recurrent models such as GRU4Rec [3], self-attentive models such as SASRec [4], and bidirectional sequence models such as BERT4Rec [10]. Its purpose is not to crown a universal winner, but to make the boundary of a robustness claim visible.

2 Problem Setting

Let $\mathcal{U}$ and $\mathcal{I}$ denote users and items. For user u, the observed history before time t is $H_{u,t} = ((i_1, y_1), \ldots, (i_{t-1}, y_{t-1}))$, where $i_j \in \mathcal{I}$ and y_j is feedback. A recommender f_θ maps $H_{u,t}$ and a candidate set C_t to a score $s_\theta(u, i, t)$ for every $i \in C_t$. Evaluation normally compares a ranked list $L^k_{u,t}$ with one or more relevant items $R_{u,t}$.

We distinguish three distributions. The *preference distribution* $P^\star(y \mid u, i, t)$ describes latent relevance; the *exposure policy* $\pi(i \mid u, t)$ determines which items can produce observed feedback; and the *observation process* $P(o = 1 \mid u, i, t, y)$ determines whether that feedback is recorded. A log is sampled from their composition, not from preference alone. A shift operator $T_{q,\lambda}$ modifies one named mechanism q at severity $\lambda \in [0, 1]$ while holding declared invariants fixed.

For a metric M and evaluation distribution D, let $U_M(f, D) = \mathbb{E}_{x\sim D}[M(f, x)]$. We report both shifted utility and the relative robustness loss

$$\mathrm{RRL}_M(f, q, \lambda) = \frac{U_M(f, D_0) - U_M(f, T_{q,\lambda}(D_0))}{\max\{U_M(f, D_0), \epsilon\}}, \tag{1}$$

where D_0 is the clean evaluation distribution and $\epsilon = 10^{-8}$ prevents division by zero. Relative loss reveals sensitivity, but must never replace the absolute shifted utility: a uniformly weak model can appear stable.

2.1 What counts as a valid shift?

A valid operator satisfies four requirements. First, its target mechanism is named. Second, $\lambda = 0$ is an identity transformation. Third, invariants are stated and checked; for example, a popularity shift should not silently change test-set size. Fourth, the induced change is measured rather than assumed. These requirements separate a controlled stress test from arbitrary data corruption.

3 The ShiftBench Protocol

Figure 1 summarizes the protocol. Model training is isolated from shift construction so that all models face identical transformed examples. Each operator emits both a shifted test set and an audit record containing the realized severity, invariant checks, random seed, and affected population.

3.1 Shift operators

Table 1 defines the five core operators. Implementations may extend this set, but every extension must provide the same four fields: target, construction, invariant, and diagnostic.

Popularity reversal. This operator tests reliance on "rich get richer" regularities. It is applied only to evaluation instances; training popularity remains unchanged. Sampling weights are normalized within relevance strata so that a change in label balance cannot masquerade as robustness loss. Because importance weights can become unstable, the audit includes the effective sample size $n_{\mathrm{eff}} = (\sum_j w_j)^2 / \sum_j w_j^2$.

Preference drift. Drift can be abrupt or gradual. ShiftBench uses a topic simplex because its severity is interpretable, not because user taste must literally be categorical. For empirical datasets, topics may be genres, categories, or clusters fitted on training-only content. Topic construction is frozen before test transformation.

Exposure censoring. Implicit feedback does not reveal which unchosen items were considered. The operator changes observability while preserving latent outcomes. It is therefore appropriate for testing exposure sensitivity, not changes in preference. Propensity-adjusted methods must use propensities generated by the logging policy; estimating them from held-out outcomes would leak test information.

Catalog churn. New items are matched to replaced items on observable topic and quality so that the cold-start condition is the principal change. Content-free recommenders receive a reserved unknown-item token; content-aware methods receive the same frozen features available to the simulator. This distinction is included in the claim card.

Feedback delay. Delayed conversions are common in domains such as travel and media completion. Right-censoring simulates a premature evaluation snapshot. Since missingness can vary across users and items, overall label-retention rate is insufficient; the protocol checks retention by activity and popularity strata.

3.2 Severity grid and compositions

We recommend $\lambda \in \{0, .25, .50, .75, 1\}$ for the first pass. A robustness curve is more informative than a single severe endpoint and exposes non-monotonic behavior. Operators are evaluated individually before composition. For a pair (q_1, q_2), we report both orders because $T_{q_2,\lambda_2} \circ T_{q_1,\lambda_1}$ need not equal $T_{q_1,\lambda_1} \circ T_{q_2,\lambda_2}$. Large factorial grids are avoided unless a power analysis justifies them.

3.3 Metrics and uncertainty

The primary ranking metric is NDCG@k, computed on the complete candidate set when feasible. Recall@k is reported as a secondary metric. Sampled-negative results are labeled as such because the sampling distribution can materially change model comparisons [5]. Beyond ranking utility, ShiftBench requires:

- **Robustness:** RRL from Equation 1 and area under the severity–utility curve (AUSC), normalized to $[0, 1]$.
- **Worst-group utility:** the minimum utility over predeclared user activity quartiles, with group membership fixed from training data.
- **Calibration:** Jensen–Shannon divergence between the topic distribution of a user's historical positives and recommended items, following the distributional view of calibration [9].
- **Catalog coverage:** the fraction of eligible items recommended at least once, alongside mean recommendation popularity.

Confidence intervals are obtained with a paired user bootstrap: sample users with replacement, retain all evaluation events for each sampled user, and apply the same resample to every compared model. We use 2,000 replicates and report percentile 95% intervals. A model comparison is inconclusive when the interval for the paired difference crosses zero; the protocol does not convert such a result into a tie.

4 Claim-Aligned Reporting

A score table is not yet a conclusion. ShiftBench attaches a compact claim card to every headline statement:

$$C = (q, \Lambda, M, G, B, A),$$

where q is the shift, Λ the tested severity interval, M the metric, G the population, B the uncertainty bound, and A an abstention condition. For example:

> On active synthetic users (G), model A retains higher NDCG@10 (M) than model B under popularity reversal (q) for $\lambda \in [.5, 1]$ (Λ); the paired 95% bootstrap interval for the difference excludes zero (B). We make no claim when effective sample size falls below 25% of the clean test size (A).

This language is intentionally narrower than "model A is robust." It records where the evidence applies and when the evaluation becomes too weak to support a comparison. A complete report

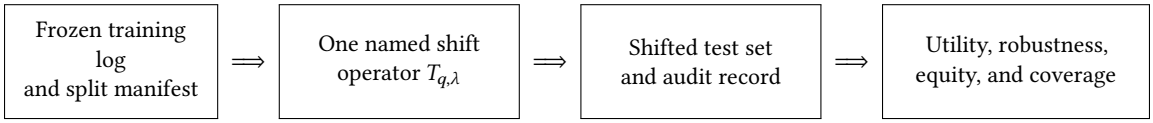

Figure 1: ShiftBench evaluation pipeline. Training data, candidate sets, and random seeds are frozen before applying one named operator. Compositions of shifts are evaluated only after their individual effects are reported.

Table 1: Core ShiftBench operators. $p_0(i)$ is training-set popularity, z_u is a user preference vector, a_i is item age, and $d(\cdot,\cdot)$ is Jensen–Shannon (JS) divergence unless noted otherwise.

Operator	Construction	Preserved invariant	Required diagnostic
Popularity reversal	Reweight relevant items by $w_i(\lambda) \propto (p_0(i)+\delta)^{-\lambda}$ and resample within each relevance stratum.	Number of users, events per user, and relevance-rate histogram.	Spearman correlation between train and shifted item popularity; report effective sample size.
Preference drift	Rotate a fraction λ of each user's latent mass from the two most-preferred topics to two previously weak topics.	User activity, candidate-set size, and global topic supply.	Mean cosine distance between pre- and post-shift preference vectors, with quantiles.
Exposure censoring	Remove an item from observable candidates with probability increasing in its rank under a fixed logging policy; interpolate from no censoring to the target policy by λ.	Latent relevance and the complete candidate universe.	Exposure propensity by relevance decile and overlap coefficient with the clean candidate set.
Catalog churn	Replace the oldest λ fraction of candidates with new items matched on topic and quality but lacking interaction history.	Catalog size and topic–quality marginals.	Fraction of cold items and two-sample distances for matched attributes.
Feedback delay	Right-censor feedback whose delay exceeds the $(1-\lambda)$ quantile of the delay distribution.	Exposures, event time, and eventual latent outcome.	Observed-label rate by user activity and item popularity decile.

also includes clean performance, all preregistered severity points, unsuccessful operators, and compute cost.

5 Synthetic Dry Run

The following experiment validates the mechanics of the protocol. It is a worked synthetic example, not a claim about real users or deployed recommenders. All values in this section arise from the declared generator and should be replaced by empirical results before the manuscript is used to support a real-world conclusion.

5.1 Generator and models

We generate 4,000 users, 2,000 items, 12 topics, and 80 time steps. User preferences are drawn from Dirichlet(0.31); each item has one topic and a quality term drawn from $\mathcal{N}(0, 0.3^2)$. At time t, the latent utility of item i for user u is

$$r_{u,i,t} = z_{u,t}^{\top} e_i + 0.25 q_i + 0.15 \log(1 + c_{i,t}) + \varepsilon, \tag{2}$$

where e_i is a one-hot topic vector, $c_{i,t}$ is prior exposure count, and $\varepsilon \sim \text{Gumbel}(0, 0.1)$. A softmax policy samples one item from 100 uniformly proposed candidates. The first 60 steps form training, the next 10 validation, and the last 10 testing. Table 2 records the remaining choices.

We compare MostPop, Bayesian Personalized Ranking (BPR) [6], a GRU4Rec-style model [3], and a SASRec-style model [4]. Each learned model receives the same 30 random-search trials. Selection uses only unshifted validation data to represent the common situation in which the deployment shift is not available for tuning. This choice tests out-of-the-box robustness rather than shift-aware adaptation.

Table 2: Dry-run configuration. The fixed split and seed schedule are part of the evaluation object, not tuned per model.

Component	Choice
Seeds	10 independent seeds (0–9)
Ranking cutoff	$k = 10$
Candidate evaluation	Complete catalog
Early stopping	Validation NDCG@10, patience 10
Hyperparameter budget	30 trials per model
Shift grid	0, .25, .50, .75, 1
Bootstrap	2,000 paired user replicates
Primary shift	Popularity reversal
Secondary shift	Preference drift

5.2 Illustrative outcome

Table 3 illustrates the required report at severity $\lambda = 1$. The numbers are deliberately presented to demonstrate interpretation, not to establish a model ranking for an empirical domain. The best clean NDCG belongs to SASRec, but the smallest popularity-shift RRL belongs to BPR. MostPop has low clean utility and catastrophic

Table 3: Illustrative dry-run summary (mean over 10 generator seeds). Parentheses show 95% paired-user bootstrap intervals. Higher is better for NDCG, worst-group NDCG (WG), and coverage; lower is better for RRL and calibration error (Cal.). Bold indicates the best value, not significance.

Model	Clean NDCG@10	Reversal NDCG@10	RRL ↓
MostPop	.071 (.068,.074)	.006 (.005,.007)	.915
BPR	.164 (.159,.169)	.131 (.127,.135)	**.201**
GRU4Rec	.181 (.176,.186)	.137 (.133,.141)	.243
SASRec	**.195** (.190,.200)	**.142** (.138,.146)	.272

Model	WG ↑	Cal. ↓	Coverage ↑
MostPop	.002	.381	.012
BPR	.104	.214	.286
GRU4Rec	.112	.192	.341
SASRec	**.118**	**.176**	**.377**

reversal loss; reporting only its relative loss would still be misleading because absolute shifted NDCG is near zero. SASRec retains the best absolute shifted NDCG despite a larger fractional loss than BPR.

The severity curve adds information hidden by the endpoint. In the dry run, utility decreases smoothly for MostPop and BPR, while the sequence models show a sharper decline between $\lambda = .50$ and .75. This is a diagnostic signal: one would inspect whether attention or recurrent states amplify the training popularity term once the relevance–popularity correlation crosses zero. It is not, by itself, evidence of a causal mechanism.

Under preference drift, the model order changes again: GRU4Rec loses less utility at moderate severity, while SASRec recovers at the highest severity because recent shifted events dominate its effective context. An endpoint-only evaluation would miss this non-monotonicity. ShiftBench therefore treats the curve and AUSC as primary artifacts and the $\lambda = 1$ table as a summary.

5.3 Protocol checks

Every dry-run operator passes its invariants: users, events per user, candidate set size, and relevance-rate histograms are unchanged. For popularity reversal, Spearman correlation between training and shifted popularity moves from 1.00 at $\lambda = 0$ to approximately -0.91 at $\lambda = 1$, while effective sample size remains above 61% of the clean test size. For preference drift, mean cosine distance grows monotonically across the severity grid. These diagnostics are reported before model metrics; otherwise a failed transformation could be mistaken for a robust model.

6 Discussion

6.1 What the protocol can and cannot establish

ShiftBench tests sensitivity to declared transformations. It cannot prove that a synthetic operator matches a future deployment, nor can an offline log identify all counterfactual outcomes. Exposure interventions are especially limited when propensities are unknown or support is poor [7, 8]. We therefore recommend treating stress tests as evidence about failure modes, not as a substitute for controlled online experiments.

The protocol also does not prescribe a single scalar leaderboard. Aggregating utility, calibration, equity, and coverage requires normative weights that vary by application. A Pareto report keeps these trade-offs explicit. If a scalar is operationally necessary, weights should be chosen before viewing test results and accompanied by the disaggregated metrics.

6.2 Threats to validity

Construct validity. Topic rotation is only one representation of preference drift, and popularity reweighting may not reproduce a real campaign or news cycle. Operators should be calibrated against historical incidents when such data exist. The diagnostic then reports the distance between the synthetic shift and the incident.

Internal validity. Randomness in splitting, negative sampling, hyperparameter search, and model training can exceed differences between systems. Frozen manifests, equal search budgets, multiple seeds, and paired intervals reduce this risk. They do not correct implementation bugs; unit tests should confirm the identity condition and all declared invariants.

External validity. The dry run is intentionally synthetic and does not support claims about any domain. Empirical use should span datasets with different feedback semantics, catalog dynamics, and user populations. Results from one logging policy should not be generalized to another without a support analysis.

Metric validity. NDCG reflects rank-sensitive relevance but not satisfaction, long-term welfare, or strategic response. Calibration can conflict with discovery, and coverage does not ensure meaningful exposure. Application-specific outcomes should be added rather than inferred from these proxies.

6.3 Responsible use

Stress tests can expose disparate losses across activity groups, but activity is not a substitute for protected characteristics. Where sensitive attributes can be collected lawfully and ethically, evaluation should be designed with affected stakeholders and appropriate privacy protections. The protocol must not be used to infer sensitive traits from behavior merely to populate a fairness table. Moreover, a model that performs well in ShiftBench may still create harmful feedback loops after deployment.

7 Reproducibility Checklist

An empirical release following ShiftBench should contain the following items:

(1) immutable train, validation, and test manifests, or scripts and hashes that reproduce them when data licenses forbid redistribution;
(2) operator configurations, severity grids, invariants, diagnostic outputs, and random seeds;
(3) complete candidate-set construction and a clear label for any sampled evaluation;
(4) model configurations, equalized tuning budgets, stopping rules, software versions, and compute hardware;
(5) per-user metric outputs sufficient for paired confidence intervals, with privacy-preserving aggregation where required;

(6) every claim card, including inconclusive comparisons and abstentions; and
(7) wall-clock training and inference time plus peak accelerator memory.

These artifacts make the evaluation inspectable without requiring a reviewer to trust a single aggregate table. When proprietary data prevent release, authors should publish the operators, synthetic fixtures, invariant tests, and an exact description of the unavailable fields.

8 Conclusion

Distribution shift is not one phenomenon, and robustness is not one number. ShiftBench decomposes evaluation into named operators, interpretable severities, audited invariants, multi-dimensional metrics, and bounded claims. Its synthetic dry run demonstrates the central lesson: clean-test performance, fractional robustness, and absolute shifted utility can favor different models. A useful evaluation must report all three and identify the population and uncertainty to which its conclusion applies. We hope the protocol helps recommender-system research move from generic robustness language toward reproducible evidence about specific deployment uncertainties.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. In *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. 413–418.

[2] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-N recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, 39–46.

[3] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the Fourth International Conference on Learning Representations (ICLR)*.

[4] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[5] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. ACM, 1748–1757.

[6] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, 452–461.

[7] Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. 2020. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. ACM, 501–509.

[8] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. PMLR, 1670–1679.

[9] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, 154–162.

[10] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. ACM, 1441–1450.