Serendipity-Aware Recommendation via Candidate Generation and Density-Based Rejection Sampling

Vivian Akoth Wagumba[1], Joseph Gathithi Wathome[2], Isaiah Onando Mulang'[1,3] and Evanson Mwangi Karanja[1]

[1]Jomo Kenyatta University of Agriculture and Technology (JKUAT)
[2]Dedan Kimathi University of Technology
[3]SAP SE

Abstract

Recommender systems are predominantly optimized for accuracy, a regime that systematically favours popular, already-familiar items and reinforces filter bubbles at the expense of *serendipity* i.e. the delivery of items that are both unexpected and relevant. We study serendipity as an explicit two-stage recommendation task. In the first stage, a diffusion-based collaborative filtering model (DiffRec) generates a ranked candidate set from a user's interaction history. In the second stage, a learned *rejection sampler* accepts or declines each candidate according to its surprise potential, estimated from item content features (genres and tags) and popularity-based novelty. We instantiate the rejection sampler with two model families: density estimators (Gaussian Mixture Models, normalizing flows, and a Bayesian estimator) that score candidates by an accept/decline log-likelihood ratio, and discriminative classifiers (random forest, neural network, XGBoost); each is paired with two decision rules (a fixed score threshold and a per-user percentile rule). We view this filtering stage as reshaping the distribution of recommended items towards a target distribution of serendipity outcomes, in the spirit of data-consistent model inversion (DCMI), and make this viewpoint operational through density-ratio-based acceptance tests. We define the task, the problem assumptions, and a reproducible evaluation protocol on the Serendipity-2018 dataset, covering fourteen pipeline configurations, a dual accuracy/serendipity metric suite, a positive/negative surprise decomposition, and threshold and decision-rule sensitivity analyses. The goal of this paper is to specify the task and a rigorous, reusable methodology for measuring serendipity gains without collapsing accuracy.

Keywords

Serendipity, Recommender systems, Diffusion models, Density estimation, Rejection sampling, Evaluation methodology

CEUR-WS.org/Vol-4341/paper8.pdf

1. Introduction

Recommender systems are extensively trained and evaluated to maximize accuracy: precision, recall, and ranking quality against held-out interactions. While useful, accuracy is a poor proxy for user value because it can be trivially satisfied through blockbuster items that a user can discover anyway [1, 2]. Optimizing accuracy in isolation creates feedback loops that narrow the user's experience over time [2], a phenomenon widely discussed under the heading of filter bubbles [3] and popularity bias [4].

Serendipity: recommending items that are at once unexpected and relevant [5, 6], is the beyond-accuracy objective that directly targets discovery. The central difficulty is that unexpectedness and relevance pull in opposite directions: the items most likely to be relevant are exactly those the user already expects and vise vasa. A serendipity method must therefore navigate a trade-off rather than optimize a single quantity. This makes serendipity both an underexplored *task* and a *measurement* challenge which is precisely the kind of setting that motivates a methodology-first treatment.

We formulate serendipity as a two-stage task. A generative recommender (DiffRec [7]) produces ranked candidates, and a learned rejection sampler retains those with sufficient surprise, estimated from item content (genres, tags), and popularity-derived novelty [8]. The rejection-sampling infuses a natural acceptance test to shift a generative distribution towards the serendipitous region of the item space [8]. Consider user *105791* from Serendipity-SAC2018 Dataset [9] whose recorded history

FRAME'26: Methodology First - Rethinking Research Assessment in RecSys Workshop, September 28, 2026, Minneapolis, Minnesota, USA

vivwagumba@gmail.com (V. A. Wagumba); wathomejosephgathithi@gmail.com (J. G. Wathome); isaiah.mulang@jkuat.ac.ke (I. O. Mulang'); mkaranja@icsit.jkuat.ac.ke (E. M. Karanja)



© 2026 Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-26

consists of just two films namely: *Total Recall, action–science-fiction title, and Friends with Benefits, a romantic comedy.* A recommender completes this history with mostly canonical films: *The Shawshank Redemption, The Godfather, Inception, The Matrix, Fight Club, The Dark Knight, Forrest Gump, and Schindler's List*, alongside *(500) Days of Summer* and *Avatar.*, yet only only the last two appear in held-out interactions. These two recommendations match the user's mixed profile while lying farther from the globally popular core, motivating a separate surprise-aware filter.

Recent work on *data-consistent model inversion* (DCMI) for mechanistic models [10, 11, 12], frames inference as changing a prior distribution on model inputs so that its push-forward through a forward map matches a target distribution of observations while remaining close to the prior. In our setting, the diffusion recommender acts as a relevance-biased proposal over candidate items, a surprise/relevance read-out defines an outcome space of serendipity features, and a learned accept/decline function implements a change of measure that steers the empirical outcome distribution towards a target serendipity distribution. Unlike DCMI, our forward map is non-differentiable, and we work per-user over finite candidate pools rather than on continuous parameter spaces for physical models, but the distributional viewpoint provides a useful lens for designing and analysing the rejection sampler.

Our contributions are: (i) a task definition and explicit assumptions; (ii) a modular method that decouples candidate generation from surprise-based filtering and can be instantiated with several density-estimation and discriminative back-ends; and (iii) a reproducible protocol covering datasets, partitioning, leakage controls, baselines, metrics, model selection, ablations, and sensitivity analyses. Together they expose when filtering helps, is neutral, or harms recommendation quality.

2. Related Work

Serendipity and beyond-accuracy objectives. Research has portrayed serendipity as comprising unexpectedness and relevance, with several operationalizations of unexpectedness based on distance from a user's expectations or from a primitive prediction model [6, 2]. Surveys catalogue the many, sometimes conflicting, definitions and metrics, and stress that serendipity is harder to evaluate consistently than accuracy [5, 1]. Auralist [13] is an early system that explicitly injects serendipity by re-ranking, while more recent work learns directional or explainable serendipity signals [14]. Novelty and diversity metrics provide complementary, item-centric notions of discovery [15]. Our work differs in treating surprise filtering as a separable, model-agnostic stage applied on top of a generative recommender, and in specifying the evaluation protocol as a first-class artifact.

Generative and diffusion-based recommendation. Generative models of user–item interactions, including variational autoencoders [16] and, more recently, diffusion models [17], have become competitive collaborative filtering backbones. DiffRec [7] adapts denoising diffusion to recommendation by injecting limited noise into corrupted interaction vectors, preserving personalized signal while retaining the generative flexibility we exploit for candidate diversity. We use DiffRec as the proposal generator and do not modify its training objective.

Density estimation and rejection sampling. Rejection sampling corrects a tractable proposal distribution toward a target by probabilistically accepting samples [8]. We borrow the acceptance-test idea and learn the accept/decline boundary with density estimators—Gaussian Mixture Models [18], normalizing flows [19, 20], and a Bayesian estimator—as well as with discriminative classifiers [21, 22]. To our knowledge, the explicit framing of serendipity filtering as a learned rejection sampler over a generative recommender's candidates, with a shared protocol to compare density-based and discriminative back-ends, has not been systematically evaluated.

3. Task Definition and Problem Formulation

Task. We address *serendipity-aware top-k recommendation.* Given a user u with interaction history $H_u \subseteq \mathcal{I}$ over an item catalogue $\mathcal{I}$, the system must return an ordered list $R_u = [r_1, \dots, r_k]$, $r_i \in \mathcal{I} \setminus$

H_u, that maximizes a joint objective trading off accuracy against serendipity. Unlike standard top-k recommendation, the target is not relevance alone but *positive surprise*.

Assumptions. We make the following assumptions for our evaluation. (A1) Relevance is observable only through held-out interactions; an item is relevant to u if it appears in u's ground-truth set G_u. (A2) Unexpectedness is estimated from content and popularity signals; it's not directly observable and there is no ground-truth labels. (A3) Items already in H_u are known to the user and are excluded from both candidates and ground truth, so that serendipity is measured only over genuinely novel items.

Serendipity formulation. For a recommended item p and user u, we define an unexpectedness score $\mathrm{unexp}(p,u) \in [0,1]$ and a relevance indicator $\mathrm{rel}(p,u) \in \{0,1\}$ with $\mathrm{rel}(p,u) = \mathbb{1}[p \in G_u]$. Unexpectedness aggregates three complementary sources of surprise,

$$\mathrm{unexp}(p,u) = w_p\, s_{\mathrm{pop}}(p) + w_c\, s_{\mathrm{cat}}(p,u) + w_f\, s_{\mathrm{feat}}(p,u), \tag{1}$$

where $s_{\mathrm{pop}}(p) = 1 - \widehat{\mathrm{pop}}(p)$ is popularity-based surprise (with $\widehat{\mathrm{pop}}$ the catalogue-normalized interaction frequency), $s_{\mathrm{cat}}(p,u) = 1 - \mathrm{Jaccard}(C_p, C_u)$ is category novelty over genre sets, and $s_{\mathrm{feat}}(p,u) = \frac{1}{2}\big(1 - \cos(\mathbf{x}_p, \bar{\mathbf{x}}_u)\big)$ is feature distance between the item profile $\mathbf{x}_p$ and the user's mean profile $\bar{\mathbf{x}}_u$. The weights (w_p, w_c, w_f) are fixed a priori (Section 6). An item is *serendipitous* iff $\mathrm{unexp}(p,u) \geq \tau_u$ and $\mathrm{rel}(p,u) \geq \tau_r$. We further distinguish a *positive surprise* (unexpected and relevant) from a *negative surprise* (unexpected but irrelevant); minimizing the latter is an explicit objective, since a method can otherwise inflate unexpectedness by recommending irrelevant long-tail items.

Pipeline objective. DiffRec yields an ordered candidate set $P_u = [p_1, \ldots, p_m]$. A learned acceptance function $a_\theta(p,u) \in \{0,1\}$ filters P_u to produce R_u. The task is to choose a_θ so that the serendipity of R_u increases relative to the unfiltered baseline P_u without an unacceptable loss in ranking accuracy.

4. A Distributional View of Serendipity-Aware Filtering

We now formalize the role of the rejection sampler inspired by DCMI in stochastic inverse problems [10, 11, 12, 23], but adapted to the specifics of serendipity-aware recommendation.

4.1. Proposal and outcome distributions

For a user u, let $P_u = [p_1, \ldots, p_m]$ denote the ranked candidate from DiffRec after masking items already consumed by the user. We view P_u as a finite support approximation to a user-specific proposal distribution P_X^u over item feature vectors $\mathbf{x}_p \in \mathbb{R}^d$. In the simplest case, P_X^u is the discrete uniform distribution over P_u, but it may also reflect the normalized DiffRec scores. Each candidate p is mapped to an outcome vector

$$\mathbf{y}_p = M(\mathbf{x}_p, u) \in \mathbb{R}^{d'} \tag{2}$$

that collects serendipity-relevant quantities. In our experiments, M returns the components of the unexpectedness score in Eq. (1) together with the overall unexpectedness and the relevance indicator $\mathrm{rel}(p,u) = \mathbb{1}[p \in G_u]$. The empirical distribution of $\mathbf{y}_p$ over $p \in P_u$ is the push-forward of P_X^u through M; we denote it by P_Y^u. The target Q_Y and class-conditional densities are estimated from the user-relative, model-agnostic accept/decline labels defined in Section 6.

4.2. Change of measure via density ratios

The filtering stage selects a probability measure Q_X^u on the finite set P_u that trades off fidelity to the proposal and proximity to the outcome distribution Q_Y. This can be expressed through f-divergences,

$$Q_X^u \;\approx\; \arg\min_{\widetilde{Q}} \Big\{ w_X D_f\big(\widetilde{Q} \,\|\, P_X^u\big) + w_Y D_f\big(\widetilde{Q} \circ M^{-1} \,\|\, Q_Y\big) \Big\}, \tag{3}$$

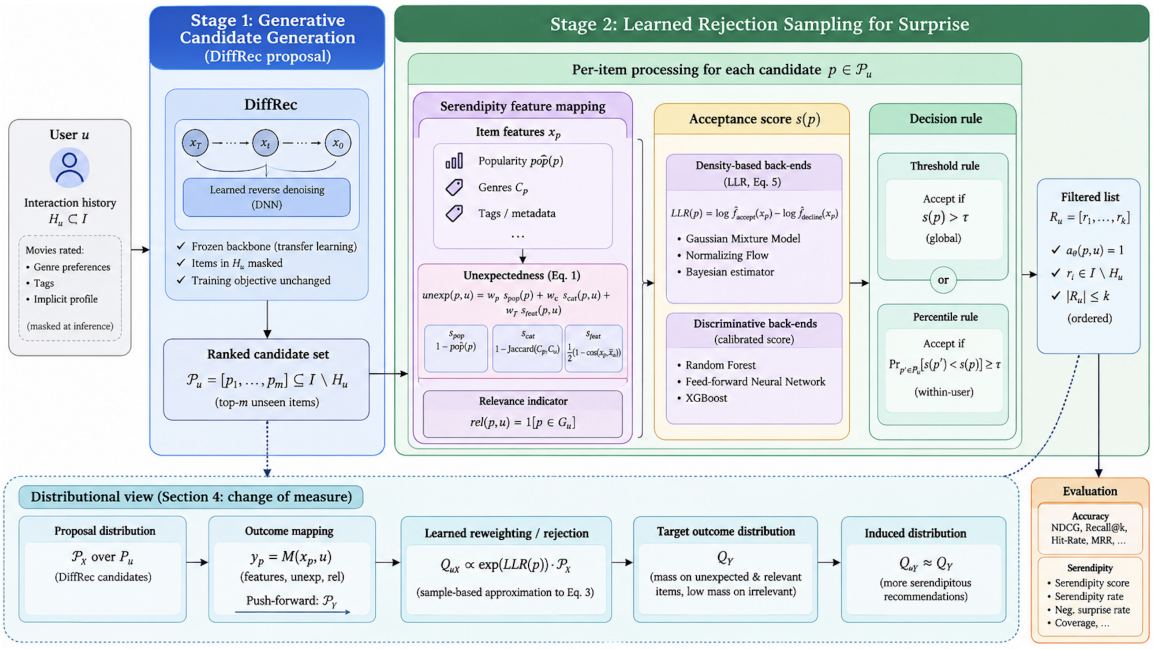

Figure 1: Two-stage pipeline: DiffRec proposes $\mathcal{P}_u$; M maps candidates to serendipity features; density-based or discriminative scores and a threshold or percentile rule produce R_u, shifting P_X^u toward Q_Y.

where $\widetilde{Q} \circ M^{-1}$ denotes the push-forward of $\widetilde{Q}$ through M, and $w_X, w_Y \geq 0$ control the strength of the two terms. We do not attempt to solve Eq. (3) directly but view the rejection sampler as a discrete change of measure on P_u by reweighting or discarding candidates according to learned density ratios. In continuous settings, such density ratios are the Radon–Nikodym derivatives that describe how one measure is obtained from another; they play a central role in DCMI algorithms that adjust prior distributions so that their push-forward matches observed data [10, 12]. Concretely, suppose we have two class-conditional densities over item features, $\hat{f}_{\text{accept}}$ and $\hat{f}_{\text{decline}}$. These densities are estimated offline from labelled examples, as described in Section 5. For a candidate p with feature vector $\mathbf{x}_p$, the log-likelihood ratio statistic

$$\text{LLR}(p) = \log \hat{f}_{\text{accept}}(\mathbf{x}_p) - \log \hat{f}_{\text{decline}}(\mathbf{x}_p) \tag{4}$$

measures how much more typical p is under the accept distribution than under the decline distribution. High values of $\text{LLR}(p)$ indicate candidates that lie within serendipitous-like items space. If we assign to each candidate a weight proportional to the exponential of $\text{LLR}(p)$ and then renormalize, we obtain a reweighted version of P_X^u that places more mass on serendipitous-like candidates. Rejecting candidates whose $\text{LLR}(p)$ falls below a threshold, or whose within-user percentile rank is too low, yields a sparse approximation to this reweighted distribution. Accordingly, the rejection sampler proffers a discrete approximation to the change of measure suggested by Eq. (3).

4.3. Connection to data-consistent inversion

DCMI for mechanistic models starts from a prior distribution on continuous-valued model parameters, pushes this through an expensive biophysical model to obtain a distribution over observables, and then uses density-ratio-based rejection to adjust the parameter distribution so that its push-forward matches the empirical distribution of observed data [10, 11, 12]. We instead use finite per-user candidate pools, a cheap engineered read-out, and one-pass filtering. The empirical outcomes of labelled accepts approximate Q_Y, while $\hat{f}_{\text{accept}}$ and $\hat{f}_{\text{decline}}$ define its contrast with declined items. DCMI is therefore a principled design analogy rather than a literal inversion: it explains how Eq. (4) enforces consistency among the proposal, serendipity read-out, and target distribution. Iteratively retraining on accepted candidates and resampling is a natural extension.

5. Methodology

The method is a two-stage pipeline (Figure 1): a fixed generative *proposal*, and a learned *rejection*.

5.1. Stage 1: Generative Candidate Generation

DiffRec [7] generates proposals and models the user interaction vector with a forward noising process and a learned reverse denoising process parameterized by a DNN that predicts the clean interaction vector x_0. At inference, the reverse process produces a score over the catalogue from which the top-m unseen items form P_u. We start from the public DiffRec implementation trained on MovieLens-1M and adapt it to the Serendipity-2018 domain with a memory-efficient transfer-learning scheme that keeps the original denoising-diffusion objective unchanged and freezes all diffusion parameters while training only a thin projection head onto the target item space. Model selection over the fine-tuning runs is by validation loss and ranking metrics (Section 6); the selected checkpoint is then frozen for all downstream experiments.

5.2. Stage 2: Rejection Sampling for Surprise

Given P_u, the sampler applies $a_\theta(p, u)$ using popularity/novelty attributes, multi-hot genres, and tags where available. Without explicit user IDs or embeddings, conditioning enters in three ways: P_u is generated from H_u; $s_{\text{cat}}(p, u)$ and $s_{\text{feat}}(p, u)$ are relative to C_u and $\bar{\mathbf{x}}_u$; and the percentile rule ranks each candidate within that user's pool. Thus, both features and decisions remain user-relative.

Density-estimation vs. Discriminative back-ends. For the *Density-estimation variant*, we learn accept and decline feature densities and score candidates; the proposal P_u from DiffRec is accepted in proportion to its fit to the target serendipitous density. As shown with the LLR in Eq. (4). We instantiate them with (i) Gaussian Mixture Models [18], (ii) normalizing flows [19, 20], and (iii) a Bayesian estimator. Higher LLR values indicate candidates closer to the surprising-yet-fitting accept distribution and implement the density-ratio reweighting in Eq. (3). Alternatively, the *Discriminative back-ends* are a contrasting family in which a binary classifiers—random forest [21], a feed-forward neural network, and XGBoost [22]— is trained to predict accept/decline directly, using their calibrated score in place of LLR. These back-ends test whether modelling the decision boundary directly is preferable to modelling the two densities.

5.3. Decision Rules

Each back-end produces a real-valued score $s(p)$ (an LLR or a classifier score). We convert scores to decisions with two rules, both evaluated for every back-end:

- *Threshold rule:* accept iff $s(p) > \tau$, for a global threshold τ. This applies a uniform surprise bar across users and items.
- *Percentile rule:* accept iff the within-candidate percentile rank of $s(p)$ is at least τ, i.e. $\Pr_{p' \in P_u}[s(p') < s(p)] \geq \tau$. This adapts the bar per user, accepting the relatively most surprising candidates regardless of absolute scale.

The threshold is simpler but scale-sensitive; the percentile is scale-free and controls acceptance per user. Validation scores are linearly rescaled to $[0, 1]$ before applying the global thresholds $\tau \in \{0.5, 0.9, 1.0\}$ allowing consistent interpretation for both unbounded log-likelihood ratios and calibrated probabilities. At $\tau = 1.0$, only maximum-scoring items pass and many threshold-based lists are empty.

5.4. Aggregation across back-ends

When multiple back-ends are combined (the full pipeline), an item is retained only if it is unanimously accepted; if no item is accepted but some are rejected, the list is empty for that user; if no back-end returns a decision, the pipeline falls back to the raw DiffRec candidates. We track unanimous-accept, unanimous-reject, and mixed-decision sets to characterize back-end agreement.

6. Experimental Methodology

This section specifies the protocol in enough detail to reproduce it. Following the methodology-first emphasis of the venue, we fix all design choices before observing outcomes.

6.1. Data, Particitioning, and Baselines

Dataset, Item Contextualization and Preprocessing. The **Serendipity-2018** dataset [9] provides ratings, genres, tags, and tag-genome features required by Eq. (1). After preprocessing, the interaction matrix spans roughly 9.5×10^4 users and 1.8×10^5 items. For robustness, we also validate DiffRec on clean and noisy MovieLens-1M variants [24].

Item popularity is computed from both global rating frequency and per-user rating behaviour, yielding the popularity-novelty signal in s_{pop}. To build accept/decline supervision for the rejection sampler we organize items into rating-pattern subsets (e.g. globally-high vs. globally-low rated, items rated highly by a user but not globally, and niche items with high ratings from few users), and derive content features (genres, tags) for each item. Within these subsets we treat items that are simultaneously highly rated by the focal user and atypical in global popularity or content profile as surrogate *accept* examples, and items that are either popular but poorly rated or rarely rated and content-distant as surrogate *decline* examples, so that all back-ends learn from user-relative yet model-agnostic labels rather than from DiffRec scores. All preprocessing scripts, subset definitions, and labelling rules are released with the code so that feature construction and supervision are auditable.

Data Partitioning and Leakage Avoidance. For each user we hold out a per-user ground-truth set G_u and expose only a fixed-size input history (the default uses a small history window per user with a minimum interaction count); the remaining interactions form G_u. Items in H_u are masked from candidate generation so that already-consumed items can never be recommended or counted as hits, eliminating history leakage. The rejection sampler is trained only on training-split items and applied to held-out users at evaluation time. Popularity statistics used in s_{pop} are computed on the training split to avoid target leakage from the test interactions.

Baselines. The primary baseline is DiffRec-only: unfiltered top-k candidates that isolate the rejection sampler's contribution. Since all configurations share the same frozen proposal, differences are attributable to filtering. We report every single back-end and the full unanimous pipeline. A non-learned novelty-aware re-ranker would provide further context but is left to future work.

6.2. Evaluation Metrics

We evaluate two complementary axes at a fixed cut-off k (default $k = 10$). *Accuracy:* Precision@k, Recall@k, F1@k, Hit-Rate@k, NDCG@k [25], MRR, and catalogue Coverage. *Serendipity:* mean unexpectedness, mean relevance, a serendipity score (unexp $\times$ rel), the serendipity rate (fraction of items that are serendipitous), and the positive/negative surprise rates. We also report the average number of items returned and the acceptance rate, which together expose over-filtering. Reporting accuracy and serendipity jointly is essential: either axis alone can be gamed (accuracy by recommending popular items; serendipity by recommending irrelevant long-tail items), so a method is credible only if it moves favourably on both, or improves one without materially harming the other. When filtering yields fewer than k items for a user, we compute Precision@k and NDCG@k using the actual list length $|R_u|$ in place of k, treat Recall@k as the fraction of ground-truth items recovered regardless of $|R_u|$, and assign zero to all top-k accuracy metrics when R_u is empty, while still including such users when averaging serendipity, acceptance, and coverage statistics.

6.3. Configurations and Model Selection

The unexpectedness weights are fixed a priori at $(w_p, w_c, w_f) = (0.3, 0.3, 0.4)$ and the serendipity gates at $\tau_u = \tau_r = 0.5$, so that the metric definition is held constant across all configurations and is not

Table 1
Accuracy vs. discovery for representative configurations (comprehensive evaluation; same frozen DiffRec proposal stage). Higher is better for all columns.

Configuration	NDCG	Serendipity	Serendipity Rate
DiffRec-only (baseline)	0.3403	0.2379	0.33
Normalizing Flow (percentile)	**0.3514**	**0.2459**	**0.34**
GMM (percentile)	0.3484	0.2452	**0.34**
Neural Network (threshold)	0.3371	0.2302	–
Random Forest (percentile)	0.3325	0.2305	–
Normalizing Flow (threshold)	0.3341	0.2341	0.33

Table 2

Mean metrics by configuration type.

Type	NDCG	Hit-Rate	Serendipity	Neg. Surprise
Baseline	0.3403	0.60	0.2379	0.67
Full pipeline	0.3166	0.50	0.2149	0.70
Individual	0.2211	0.37	0.1535	0.66

Table 3
+ve/-ve surprise decomposition at threshold 0.5. Acc. is the acceptance rate.

Configuration	Pos. Surp.	Neg. Surp.	NDCG	Acc.
DiffRec-only	2.42%	13.11%	0.3446	100%
GMM (threshold)	2.45%	13.01%	0.3449	100%
XGBoost (threshold)	6.03%	37.52%	0.0992	30%

tuned to favour any method. These values were chosen to give roughly comparable contributions from popularity, category, and feature-based surprise under the Serendipity-2018 distribution, and we treat them as fixed design commitments in this initial study; sensitivity to these weights and gates is left to future work, while the threshold sweeps already expose how different acceptance regimes trade accuracy against serendipity. DiffRec fine-tuning selects the checkpoint by lowest validation loss together with ranking metrics (Recall@$\{5, 10, 20\}$, NDCG@$\{5, 10, 20\}$) under early stopping. For the rejection samplers, the decision parameter τ is treated as a controlled variable rather than tuned for best results: we sweep it (Section 7) and report the full curve. Density and discriminative back-ends are trained on the same features and supervision to keep the comparison fair.

We evaluate **fourteen configurations**: the DiffRec-only baseline; twelve single back-ends (Gaussian Mixture Model, normalizing flow, Bayesian, random forest, neural network, and XGBoost, each with the threshold and the percentile decision rule); and the full unanimous-acceptance pipeline. Each configuration is run at multiple decision points (thresholds in $\{0.5, 0.9, 1.0\}$) and at two user-scale settings (on the order of 3.9×10^4 and 9.7×10^4 users) to assess stability. For every non-baseline configuration we separately evaluate the filtered pipeline output and the raw DiffRec output, so the effect of filtering is always measured against an identical proposal set.

7. Results

We summarize the evaluation across configurations. The same frozen DiffRec proposal stage underlies every row, so differences are attributable to the rejection sampler and its decision rule. The main results shown in Table 1 reports accuracy (NDCG) and discovery (serendipity score and rate). Two configurations; the normalizing-flow and GMM under the percentile rule, improve *both* NDCG and serendipity over the baseline. Most other single back-ends improve serendipity only by sacrificing accuracy, consistent with the trade-off anticipated in Section 3. Table 2 compares configuration type, showing that the baseline retains the highest accuracy and serendipity on average; the full unanimous-acceptance pipeline loses both because it over-rejects; and individual back-ends vary widely. This confirms that aggressive unanimous filtering is counter-productive under the current supervision.

For clarity, the serendipity rate reported in Table 1 denotes the fraction of recommended items across all users whose unexpectedness and relevance scores simultaneously exceed the serendipity gates, whereas the positive and negative surprise percentages in Table 3 are computed as the proportions of recommendations labelled as positive or negative surprises out of all recommended items at the given threshold. The small differences between the baseline and GMM-threshold rows in Table 3 despite

a rounded 100% acceptance rate arise from finite-precision thresholding and from the fact that the GMM filter operates on slightly different candidate-score distributions, and the baseline NDCG values change across tables because the user subsets used in the corresponding aggregations differ in size and composition.

Table 3 decomposes surprise at threshold 0.5 ($\sim 3.9 \times 10^4$ users). The GMM threshold filter slightly increases positive surprise while *reducing* negative surprise and preserving NDCG. XGBoost nearly triples positive surprise but only by accepting ~30% of candidates, which explodes negative surprise and collapses NDCG. This is exactly the negative-result pattern the protocol was designed to surface. Across back-ends, the percentile rule is consistently safer than the threshold rule because it controls the acceptance rate per user and avoids the empty-list failure that absolute thresholds produce when scores are poorly calibrated. As the threshold increases towards 1.0, several configurations drive the acceptance rate (and hence accuracy) to zero, confirming the scale-sensitivity of the threshold rule.

8. Discussion

The experiments support a nuanced reading of the rejection-sampling approach. Density-based back-ends paired with the percentile rule can yield small simultaneous gains in accuracy and serendipity, which is encouraging given that the proposal stage is frozen and the filter sees only content features. However, the dominant lesson is methodological: the most "serendipitous-looking" configuration (XGBoost) is the worst on negative surprises and accuracy, and it would have looked attractive under a serendipity-only metric. This is direct evidence for evaluating accuracy and serendipity jointly and for reporting acceptance rates and negative-surprise rates alongside any serendipity gain. The over-rejection of the full unanimous pipeline indicates that the current accept/decline supervision is too conservative when intersected across back-ends; softer aggregation (e.g. majority vote or score averaging) and learning the acceptance boundary against observed serendipitous interactions rather than heuristic subsets are natural next steps. The unexpectedness score is heuristic and content-based (A2), so absolute serendipity numbers should be read as protocol-relative rather than ground truth; we mitigate this by fixing the metric a priori and by reporting the full sensitivity curves.

Limitations and ethical considerations. Our operationalization of surprise relies on popularity, genre, and tag-based proxies, which may not fully capture users' experienced serendipity and may influence exposure patterns for niche providers. In particular, long-tail items that are rare and content-distant can be favoured by the unexpectedness components even when their true relevance is uncertain, and the rejection sampler's filtering could therefore alter the balance between head and tail exposure in ways that merit careful scrutiny. Moreover, because the degree of exploration is controlled by internal thresholds and percentile rules, users do not directly observe or adjust the level of serendipity in their recommendations, which raises questions about transparency and autonomy that we do not address in this initial study. We regard these exposure, transparency, and user-control aspects as important topics for future work rather than as solved issues in the present methodology.

9. Conclusion

We framed serendipity as an explicit two-stage recommendation task and proposed a modular method that filters a diffusion recommender's candidates with a learned rejection sampler, instantiated with density-based and discriminative back-ends and two decision rules. Our primary contribution is a precise task definition and a reproducible, dual-axis evaluation protocol—spanning fourteen configurations, a positive/negative surprise decomposition, and decision-rule, threshold, and scale ablations—that exposes when surprise-based filtering helps and when it silently harms recommendation quality. We release code, dataset preparation scripts, and configurations so that the task and protocol can serve as shared infrastructure for future serendipity research.

Declaration on Generative AI

During the preparation of this work, the author(s) used a generative AI assistant in order to: grammar and spelling check, image generation, and editing assistance for formatting. After using these tool(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: Evaluating recommender systems by coverage and serendipity, in: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys), ACM, 2010, pp. 257–260. doi:10.1145/1864708.1864761.

[2] M. Kaminskas, D. Bridge, Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems, ACM Transactions on Interactive Intelligent Systems (TiiS) 7 (2016) 1–42. doi:10.1145/2926720.

[3] T. T. Nguyen, P. Hui, F. M. Harper, L. Terveen, J. A. Konstan, Exploring the filter bubble: The effect of using recommender systems on content diversity, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 677–686.

[4] H. Abdollahpouri, Popularity bias in ranking and recommendation, in: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES '19, ACM, New York, NY, USA, 2019, p. 529–530. doi:10.1145/3306618.3314309.

[5] D. Kotkov, S. Wang, J. Veijalainen, A survey of serendipity in recommender systems, Knowledge-Based Systems 111 (2016) 180–192. doi:10.1016/j.knosys.2016.08.014.

[6] P. Adamopoulos, A. Tuzhilin, On unexpectedness in recommender systems: Or how to better expect the unexpected, ACM Transactions on Intelligent Systems and Technology (TIST) 5 (2014) 1–32. doi:10.1145/2559952.

[7] W. Wang, Y. Xu, F. Feng, X. Lin, X. He, T.-S. Chua, Diffusion recommender model, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), ACM, 2023, pp. 832–841. doi:10.1145/3539618.3591663.

[8] J. von Neumann, Various techniques used in connection with random digits, National Bureau of Standards Applied Mathematics Series 12 (1951) 36–38.

[9] D. Kotkov, J. A. Konstan, Q. Zhao, J. Veijalainen, Investigating serendipity in recommender systems based on real user feedback, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC), ACM, 2018, pp. 1341–1350. doi:10.1145/3167132.3167276.

[10] T. Rumbell, C. Wanjiru, I. O. Mulang', S. Obonyo, J. Kozloski, V. Gurev, Sequential data-consistent model inversion, in: NeurIPS 2023 Workshop on Deep Learning and Inverse Problems, 2023.

[11] T. Rumbell, J. Parikh, J. Kozloski, V. Gurev, Novel and flexible parameter estimation methods for data-consistent inversion in mechanistic modeling, Royal Society Open Science 10 (2023) 230668. doi:10.1098/rsos.230668.

[12] T. Butler, J. Jakeman, T. Wildey, Combining push-forward measures and bayes' rule to construct consistent solutions to stochastic inverse problems, SIAM Journal on Scientific Computing 40 (2018) A984–A1011. doi:10.1137/16M1087229.

[13] Y. C. Zhang, D. Ó. Séaghdha, D. Quercia, T. Jambor, Auralist: Introducing serendipity into music recommendation, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM), ACM, 2012, pp. 13–22. doi:10.1145/2124295.2124300.

[14] X. Li, W. Jiang, W. Chen, J. Wu, G. Wang, Directional and explainable serendipity recommendation, in: Proceedings of The Web Conference (WWW), ACM, 2020, pp. 122–132. doi:10.1145/3366423.3380100.

[15] P. Castells, S. Vargas, J. Wang, Novelty and diversity metrics for recommender systems: Choice, discovery and relevance, in: Proceedings of the International Workshop on Diversity in Document Retrieval (DDR), 2011, pp. 29–36.

[16] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the World Wide Web Conference (WWW), ACM, 2018, pp. 689–698. doi:10.1145/3178876.3186150.
[17] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, in: Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020, pp. 6840–6851.
[18] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[19] D. J. Rezende, S. Mohamed, Variational inference with normalizing flows, in: Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015, pp. 1530–1538.
[20] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, Journal of Machine Learning Research 22 (2021) 1–64.
[21] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32. doi:10.1023/A:1010933404324.
[22] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
[23] I. O. Mulang', S. Obonyo, T. Rumbell, V. Gurev, C. Wanjiru, Feature importance random search for hyperparameter optimization of data-consistent model inversion, in: NeurIPS 2023 Workshop on Deep Learning and Inverse Problems, 2023. URL: https://openreview.net/forum?id=qvLWGWsMyq.
[24] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Transactions on Interactive Intelligent Systems (TiiS) 5 (2015) 1–19. doi:10.1145/2827872.
[25] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446. doi:10.1145/582415.582418.