Learning User Boredom Constraints in Sequential Recommender Systems

Noam Zvi
Bar Ilan University
Ramat Gan, Israel
noamzvi22@gmail.com

Gal A. Kaminka
Bar Ilan University
Ramat Gan, Israel
galk@cs.biu.ac.il

ABSTRACT

We consider sequential recommender systems that work in multiple sessions, with a fixed catalog. Each session opens with a single recommendation. Acceptance leads to another recommendation. The session ends upon first rejection. The goal is to maximize session length. Myopic exploitation of previously-successful recommendations quickly leads to user boredom. We introduce novel bandit algorithms that improve recommendation variety by learning and enforcing per-user, per-item boredom thresholds. This allows repeated recommendations, appropriately spaced in time, with a high acceptance rate. Learning takes place in two stages: (i) item-specific boredom thresholds are determined; (ii) once the thresholds are known, preference for the item is learned via a standard bandit algorithm. Evaluation using user data from a commercial system demonstrates clear improvements in session length.

KEYWORDS

Boredom, Reinforcement Learning, Recommendation, Diversity, Recommender System, HAI, HRI

1 INTRODUCTION

Recommender systems are widely used to personalize content recommendations [4, 17, 19, 24]. A fundamental challenge is to vary recommendations (recommendation diversity) while targeting user preferences based on past accepted recommendations [18]. Indeed, recommender systems struggle with repetitive recommendations, leading to declining user satisfaction and engagement. Using reinforcement learning (RL) in recommendation systems amplifies this issue: standard RL-based recommenders exploit high-reward items (successful recommendations) and fail to account for user boredom, resulting in disengagement [1]. Addressing this limitation is crucial for sustaining long-term user engagement in recommender systems.

Many approaches to promoting diversity assume large catalogs, where rich item metadata, item- or user- similarity, or explicit user feedback can be brought to bear [14, 30, 31, e.g.,]. Such data enables discovering items that have not been proposed (at all, or recently), and match latent user preferences.

Unfortunately, in many real-world settings, repeatedly recommending the same item may be required due to a limited catalog or because they are inherent to the task of the recommendation system.

As a motivating example, consider ElliQ [7], a commercial home robot for promoting cognitive health of elderly users (see Section 2.1 for details). ElliQ has about 150 interactive activities (called *plans*). As a *conversational recommender system* [23] it uses natural language to propose plans to engage the user, when given an opportunity. Once a plan is accepted and completed, ElliQ can propose a follow-up to prolong the session. Making a good recommendation is critical: the user either accepts or rejects; the session terminates on the latter. Yet, users vary not only in their preference for a plan (e.g., some like music, others like drawing), but also in their preferred gap between repetitions (some prefer drawing every day; others, twice a week). As some plans are routine by design (for health reasons), making a good recommendation is very challenging.

We propose a novel approach to diversify recommendations in sequential systems with inherent repetitions between sessions, by explicitly modeling—and *learning*—user response as a function of time. We accept that the user response appears non-stationary: it is a function of *spacing* (time since the last recommendation was made), or a *cap* on the frequency of the recommendation. We use *boredom constraints* as a general term covering such functions (see [30] for more examples). The recommendation system can use these functions as constraints, filtering repeated recommendations that are likely to be beyond user tolerance.

We develop multiarm-bandit recommendation algorithms that learn, and then enforce, user- and item- specific boredom constraints from repeated interactions with only binary rewards: accept or reject. The algorithms work in two stages, for each user and item. First, the boredom constraint of the item is identified efficiently using a probabilistic binary search algorithm, that uses confidence bounds to estimate where a *boredom threshold* exists, distinguishing likely rejection (user too bored), and acceptance (at some expected rate) due to user preferences. Once the threshold is known with sufficient confidence, a second stage begins exploiting the threshold, using standard multi-arm bandit to continue exploration/exploitation of the true acceptance rate. This process ensures that diversity is enforced endogenously.

We evaluate the proposed learning method using a simulator that replicates user–system interactions based on commercial real-world data, comprising 117 user profiles and a realistic item catalog. This controlled setting enables comparison against classical baselines, such as UCB1 [3] and Thompson Sampling [26]. The results demonstrate that the proposed two-stage learning algorithms consistently outperform these baselines in terms of converged session length. Beyond performance, the experiments provide insight into the dynamics of boredom threshold learning, factors influencing session length, and practical mechanisms for reducing learning overhead.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

2 MOTIVATION AND BACKGROUND

Recommender systems personalize suggestions by leveraging user interaction histories [4, 17]. While traditional collaborative filtering exploits cross-user similarities [16], Sequential Recommender Systems (SRS) specifically model the dynamic evolution of user preferences over time, often segmenting interactions into discrete sessions to capture short-term intents [5, 29].

A central challenge in these interactive environments is recommendation diversity [18]. Recent work employs reinforcement learning (RL) to capture patterns in sequential decision-making [1, 10]. However, by focusing on reward maximization, RL and MAB approaches inherently risk over-exploiting high-reward items, leading to repetitive recommendations and reduced long-term engagement. The challenge is amplified when *some* repetition is inherent to the recommendation task, or when the item catalog is limited, or recommendations are delivered as sequences (or slates) of items within a single session [25, 28].

In Section 2.1 we present a motivating study of a successful beta-experiment with real users applied in a commercially deployed robot. This system operates at the intersection of conversational and sequential recommendation, where interactions occur in distinct sessions and the order of suggestions is critical. The necessity of modeling boredom thresholds is demonstrated in this case study, but extends to various small-catalog domains. For instance, a smart wardrobe system must learn the acceptable interval before suggesting the same outfit [12], a restaurant recommender must identify the optimal period between visits [2, 9, e.g.,], etc. Section 2.2 discusses current approaches for promoting recommendation diversity, and the open challenge we tackle in this paper.

2.1 A Conversational Recommender for a Robot

The task of the ElliQ robot [7] is to maintain user engagement over time, promoting cognitive and physical health and decreasing loneliness in elderly users. ElliQ has About 150 interactive activities it can recommend, varying from conversations and games to health-related check-ins. ElliQ's proactive session initiation, contextual conversation capabilities, and goal-based decisions leverage personalized user data to initiate and maintain user engagement. As a fallback, it employs conversational recommender capabilities that rely on rule-based heuristics, with carefully weighted stochastic decision-making, to vary the recommendation. In addition, the company employs item-specific boredom constraints, common to all users, that prohibit certain items from being recommended too frequently or too soon after the last recommendation.

Together with the company, we conducted an experiment with a group of 34 beta users, where a multi-arm bandit recommender was utilized to improve on the heuristic recommendation, while satisfying the boredom constraints as given. The recommender utilized a simple variant of the UCB1, where only plans satisfying the boredom constraints were allowed for selection.

Despite the absence of item metadata and state attributes, the algorithm achieved clear improvements in acceptance rate over time, from 61% after two weeks to 68.9% after eight weeks. For comparison, the heuristic mechanism used obtained an acceptance rate of 64.7%. These results highlight boredom as a key factor in maintaining user acceptance within sequential interaction loops. The boredom constraints, coupled with a classic bandit algorithm, were sufficient to significantly improve acceptance rates (two-tailed t-test, $p \approx 0.003$), despite the limited catalog and the repetitiveness inherent to the task.

2.2 Boredom as a Driver of Diversity

Recommendation diversity has no one accepted definition [11]. A widely used definition of diversity is the average dissimilarity between items in a recommendation set [6]. This assumes (i) recommendations are returned as a set in each interaction, and (ii) a similarity function exists to compare items through meta-information and descriptive features [30]. Both assumptions fail in interactive sequential and session-based systems (including ElliQ), where suggestions are made sequentially and item metadata is scarce [23].

An implicit motivation for diversity is the non-stationarity of user feedback: the same item may be accepted at one time and rejected at another. This creates a need to vary recommendations, to match the user's shifting feedback [18]. Mourão et al. [21] proposes that finding once relevant items to the user and again suggesting them will increase diversity without compromising accuracy. This would have worked as dynamic boredom constraints on the item. While discussing the *oblivion problem* challenge of finding such items, they do not propose a solution.

Other studies examine the non-stationary user feedback through the lens of modeling the user's psychological state regarding the item, as a function of boredom rather than a change in pure preferences for the item. For example, [14] modeled the user state as a hidden semi-Markov model with two states: sensitization and boredom, so the recommendations are made according to the user's state towards each item and the duration of it. While related, we depart from these approaches by embedding boredom directly as a constraint that is learned (first stage), and then enforced while the pure preferences are made more precise (second stage). This avoids explicit modeling and estimation of the user's boredom state.

Warlop et al. [30] address long-term engagement by modeling rewards as a function of item recency within a MDP framework. While this captures the temporal effect of repeated exposure, the resulting state-based formulation requires planning over histories whose size grows exponentially with the length of the window w (K^w). Although this approach leverages function approximation to mitigate this issue, the underlying combinatorial structure remains challenging, particularly as the catalog size increases. The challenge is further amplified in session-based systems, where multiple recommendations within a single interaction effectively extend the history horizon, making explicit state-space planning impractical.

3 DIVERSITY CONSTRAINTS

We model the sequential recommendation system as a constrained decision problem. Its goal is to maximize the expected cumulative reward while ensuring boredom constraints are satisfied. In this section, we discuss the nature of these constraints. Section 4 discusses how they are learned and enforced.

Generally, in sequential recommender systems, interactions are divided into sessions, where a session is defined as a sequence of recommendations that may terminate upon user rejection (in our settings, this occurs with probability one). Recommendation

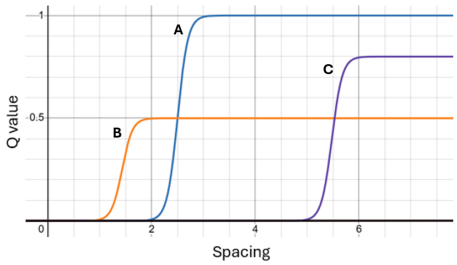

Figure 1: Example of user behavior towards items under spacing constraint. Each curve corresponds to a sigmoid with different boredom thresholds (t) and preference values (q).

diversity can therefore manifest as *intra-session diversity*—variety within a single session—and *inter-session diversity*, variety across sessions. Both avoid repeated exposure to the same items over time. to the degree possible and desired.

We address inter-session diversity, distinguishing two types of boredom constraints: *Spacing constraints* require a minimum of separation in time between recommendations of an item, regardless of acceptance. *Capping constraints* restrict the number of times an item may be recommended within a defined time interval. Requiring that an item cannot be recommended if it appeared less than two days ago is an example of a spacing constraint. Requiring that it cannot be recommended more than three times in two days is a capping constraint.

Practically, for a conversational recommender system, a common constraint is that an item cannot be recommended twice within the same conversation (a *session*). This is a capping constraint that is so common, we assume it is enforced always to avoid irritating the user. We instead focus on measuring time (for both constraints) in session units.

A user's boredom response with respect to a specific item is modeled as a sigmoidal function parameterized by the expected *preference* q for the item (relevant when the user is not bored with it), and the *boredom threshold* $0 \leq t \leq w$ (0 allowing immediate reselection, and w prohibiting selection no matter the time), determining the spacing or capping boredom. The sigmoid function distinguishes two regions in the user behavior, defined using the current boredom value b, which is a function of the occurrences of the item within the session memory window w (discussed later). In the *boredom region* ($b \leq t$), the item is deterministically rejected. In the *engagement region* ($b > t$), the item is accepted with probability q. Both t and q are specific to each user-item pair.

Figure 1 illustrates the user response model under *spacing* constraint. Each curve represents an item characterized by distinct boredom threshold (t) and preference (q) parameters. The boredom value b of each item is determined by when it was last recommended. For example, curve **A** corresponds to an item with $t = 3$ and $q = 1$. Assuming two sessions per day, this implies that the item is deterministically accepted once at least two days have passed since it was last recommended. In contrast, curve **B**, with $t = 2$ and $q = 0.5$, represents an item that is always rejected if less than a day and a half has passed, but accepted with probability 0.5 once that spacing threshold is exceeded.

Note that *capping* exhibits a sigmoidal pattern opposite in direction to *spacing*. To maintain consistency between boredom measures, we redefine capping by its complementary form, using b to count the number of sessions in the window during which the item does *not* appear. This aligns both spacing and capping along a common axis of *boredom relief* b as t increases, unifying algorithmic treatment.

Although we describe the response using a sigmoid, it should be understood as an approximation of a step function. Since boredom values are discrete session counts, the difference between the two formulations is negligible in practice.

This modeling is supported by beta-user experiments, where simple algorithms achieved high acceptance with boredom constraints, as well as prior work employing hard-coded thresholds [8, 31]. Nonetheless, we believe that the framework can extend to any monotonically increasing function capturing a boredom–engagement transition.

4 LEARNING USER BOREDOM THRESHOLDS

The sigmoidal formulation offers a practical advantage: it decouples learning of the boredom threshold t and the preference parameter q. Specifically, we adopt a two-stage approach, utilizing a multi-armed bandit as its backbone. First, t is estimated by identifying the point acceptance transitions from low to high, without requiring precise knowledge of q. Then, once t is determined, it can be enforced as a hard constraint during the learning of q, improving accuracy.

To do this, we maintain two lists of items between sessions, for each user: the *explore* list for items whose t value is not known with sufficient confidence, and the *exploit* list for items whose t values can be confidently used (and whose q values can now be learned). All items are initially placed in the explore list. For each item a, we maintain the number of times it has been recommended ($total[a]$), accepted ($accepts[a]$), and current estimated lower- and upper- bounds ($lower[a], upper[a]$) on its boredom threshold t.

Algorithm 1 is called with every new session. It is given a history of the last w sessions, which it can query to determine which items have been recommended at any session within this window.

Algorithm 1: Single-Session(Last w sessions)

```
1  repeat
2      A ← AvailableActions(explore_list)
3      if A not empty then
4          a ← ExploreAction(A);
5          if IsCertain(a) then
6              Remove(explore_list, a) ;
7              Add(exploit_list, a)
8      else
9          A ← AvailableActions(exploit_list)
10         if A not empty then
11             a ← ExploitAction(A);
12 until a is rejected or no exploit action available;
```

First (line 2), candidate items for exploration are determined using their current boredom relief value b and the item *lower* bound, i.e., if $b \geq lower[a]$. Then, the t-learning (exploration) algorithm

is called in line 4 to recommend an item (Algorithm 2, see below). The algorithm checks using the $IsCertain()$ exploration stopping criterion (line 5) whether t is confidently known, i.e., if a target difference between the upper and lower bounds is achieved. To prioritize accuracy, the most conservative setting is a target difference of 1 (which we used). If so, the item is transferred to the *exploit* list. Rejection by the user terminates the session.

When there are no items available for exploration, selection proceeds via a bandit algorithm to learn q values for items in the *exploit* list (line 11). We use classic multi-arm bandit algorithms (Section 5); their own exploration explores and learns the q value for the item, while enforcing boredom thresholds as hard constraints (line 9). If no item can be selected, the session ends (line 12).

Exploration of boredom thresholds: Learning t. This stage estimates the boredom threshold t for an item. Estimation follows a binary search-inspired process (Algorithm 2) that iteratively refines $lower$ and $upper$ bounds until a reliable boredom threshold is obtained.

Algorithm 2: ExploreAction(List of items$A$)

1 $a \leftarrow$ SelectAction(A);
2 Execute a, observe reward r;
3 $b \leftarrow \min(\text{get_boredom}(a), w)$;
`// Update statistics`
4 $count(a,b) \leftarrow count(a,b) + 1$;
5 $accepts(a,b) \leftarrow accepts(a,b) + r$;
6 $rate(a,b) \leftarrow \frac{accepts(a,b)}{count(a,b)}$;
7 UpdateBounds(a, b);
8 **return** a

Each item's boredom range is initialized to $[0, w]$, and in each step, a *scheduler* selects the next recommendation based on the current bounds (line 1). It prioritizes items due for testing. If more than one such item is available, two tie-breaking rules are possible: prioritize items based on information (preferring items less confidently known), or based on expected satisfaction (q). As a default, we used the information-based rule (but see Section 5.4).

Upon observing user feedback, the item's current boredom relief value b is retrieved (line 3) using *get_boredom* method, which returns the current boredom relief value based on the last w sessions. Thresholds are bounded by w; if the true boredom threshold lies outside the window, the algorithm assumes it equals w, causing the item to be rarely suggested. Since responses are noisy, rejections and acceptances are treated as evidence: Lines 4–6) track the number of attempted recommendations for the item, the number of accepted recommendations (r is 1 if the item is accepted, otherwise 0), and the acceptance rate. We recall that all of these are maintained between sessions.

Finally, the search bounds $lower$ and $upper$ are updated (line 7) using a bound estimation method. We present below two alternative methods to balance reliability and efficiency. Both methods rely on a *success rate threshold* τ, the minimum acceptance probability at which an item is considered valid for recommendation at a given boredom level

Sample-Based Exploration (Algorithm 3). This method updates thresholds based on empirical acceptance rates. It uses *sample size threshold* N that determines how many proposals of a are permitted at each boredom level b, before a decision is made. If the number of proposals ($count[a, b]$) is greater than N, the item has been examined sufficiently. When the observed rate for a given boredom level b consistently falls below the target threshold τ, the lower bound is increased (line 3); conversely, when it remains above τ, the upper bound is decreased (line 5). Larger N reduces variance but slows down exploration.

Algorithm 3: N-Sample UpdateBounds(a, b)

1 **if** $count(a,b) \geq N$ **then**
2 **if** $rate(a,b) < \tau$ *and* $b > lower(a)$ **then**
3 $lower(a) \leftarrow \min(b, upper(a))$;
4 **else if** $rate(a,b) \geq \tau$ *and* $b < upper(a)$ **then**
5 $upper(a) \leftarrow \max(b, lower(a))$;

Chernoff-Based Exploration (Algorithm 4). This method refines the update rule by utilizing the Chernoff-Hoeffding bound [22] to account for statistical uncertainty. The *error tolerance* ε specifies the maximum deviation allowed between the estimated and the true acceptance rate. The *confidence level* δ determines the probability that the acceptance rate estimate lies within the error tolerance ε. A higher δ value requires stronger evidence before bounds are updated. It is calculated as $\delta = 2\exp(-2n\varepsilon^2)$, where n is the number of samples.

Algorithm 4: Chernoff-Based UpdateBounds(a, b)

1 **if** $rate(a,b) < \tau$ **then**
2 $\varepsilon \leftarrow \varepsilon_{low}$
3 **else**
4 $\varepsilon \leftarrow \varepsilon_{high}$
5 $d(a,b) \leftarrow \text{Chernoff}(count(a,b), \varepsilon)$
`// Update upper bound: Lines 6-8`
6 $\mathcal{S} \leftarrow \{b' \mid b' \in [0,w],\ \delta_{low} \leq d(a,b') \leq \delta_{high},\ \text{rate}(a,b') > \tau\}$
7 **if** $\mathcal{S}$ *not empty* **then**
8 $upper(a) = \max_{b' \in \mathcal{S}} d(a,b')$
`// Update lower bound: Lines 9-11`
9 $\mathcal{S} \leftarrow \{b' \mid b' \in [0, upper(a)],\ \delta_{low} \leq d(a,b') \leq \delta_{high},\ \text{rate}(a,b') \leq \tau\}$
10 **if** $\mathcal{S}$ *not empty* **then**
11 $lower(a) = \max_{b' \in \mathcal{S}} d(a,b')$

Here, empirical acceptance rates are evaluated against the target τ, as defined above. We used asymmetric error bounds for the boredom and engagement regions (ε_{low}, ε_{high}). This asymmetry reflects the narrower acceptance probability range in the boredom region (around 0.05) compared to the engagement region (up to 0.95), allowing larger estimation errors when the acceptance probability is high.

In line 5, the confidence of the current boredom value δ is calculated using Chernoff-Hoeffding bound, with the number of samples collected so far n and the relevant error tolerance ε, and saved in $d(a, b)$, which is maintained between sessions. Bounds are updated

for the most confident value in the interval $[\delta_{low}, \delta_{high}]$ if one exists (lines 8, 11).

5 EXPERIMENTS AND RESULTS

We evaluate the boredom-constraint learning approach via a series of experiments examining its effectiveness and sensitivity to parameter choices. We describe the experimental setup (Section 5.1), and then evaluate the boredom-learning algorithms in Section 5.2. We report on factors influencing the exploration phase (Section 5.3), and present approaches for reducing its length (Section 5.4).

5.1 Experimental Setup

Evaluating recommendation algorithms requires repeated experimentation under controlled conditions, which are rarely feasible with live users. As a result, user-simulation platforms have become a standard tool for benchmarking [13, 15, 20, 32]. Our own user simulation[1] was grounded with anonymized real-world data from ElliQ's parent company, *Intuition Robotics*. The data includes item catalog, anonymized user-item preference profiles, and aggregate item-level acceptance statistics.

The user-item preference profile of 117 users (the user-item q values in the engagement region) were established from this data, with respect to 42 activities (items). Separate boredom thresholds for spacing and capping (the user-item t values, determining the separation between boredom and engagement regions) was initialized by sampling uniformly for each item and user, with an exception: an item that was *always* rejected (accepted) by a user as evident in the data, was assigned a threshold of $w = 14$ (0, resp.).

Overall, 117 profiles were tested, varying in their overall acceptance rates, per-item preferences and per-item boredom thresholds. Each single-user experiment (with a specific algorithm) was repeated 50 times with 3,000 simulated sessions per run, under *cold start* conditions (each run starts with no priors on learned q or t). This is a conservative evaluation, so as to avoid biasing the results; see Section 5.4 where we address this in more depth. Results are reported as the mean session length over the final 500 sessions after exploration is finished, capturing steady-state user engagement in a sequence of recommendations. The results reported below are averaged across all users.

Empirical Setting of Parameter Values. We conducted pilot experiments to establish stable parameter values:

The success-rate threshold τ is used in Algorithms 3 and 4 to establish separation of the boredom and engagement regions. Items with acceptance probability below this cutoff are considered to fall in the boredom region. We evaluated values in the range $[0.05, 0.20]$ and observed negligible differences in session length, with slightly shorter exploration for lower values. We therefore set $\tau = 0.05$.

For Chernoff-based confidence estimation, we tested $\varepsilon_{low} \in [0.10, 0.25]$ for the boredom region and $\varepsilon_{high} \in [0.30, 0.45]$ for the engagement region. Across all evaluation metrics, the configuration $\varepsilon_{low} = 0.25$ and $\varepsilon_{high} = 0.45$ consistently performed best. Similarly, we tested $\delta_{low} \in [0.05, 0.35]$ and $\delta_{high} \in [0.40, 0.80]$. Setting $\delta_{low} = 0.10, \delta_{high} = 0.60$ yielded the best results.

For the heuristic N-samples method, we tested $N \in [3, 10]$, including both symmetric and asymmetric configurations between boredom and engagement regions. The best performance was obtained with $N = 5$ in both regions.

5.2 User Engagement

The bottom-line measure of performance of a recommender algorithm is its impact on user engagement. In the case of a sequential system, this is best captured by the resulting converged session length, once learning advances beyond a base exploration phase.

There are four variant boredom-learning algorithms. They vary in their t-learning (exploration) phase lower/upper bound update algorithms (Algorithms 3 and 4), and in their user preference-learning (t exploitation, learning q values) phase algorithms (UCB [3] and Thompson sampling [26]). The latter induce natural baselines, utilizing UCB and Thompson to be used without attempting to learn the boredom threshold of items. So as to make the baselines performance reasonable, all algorithms were prevented from repeating a recommendation within a session: once a recommended item was accepted by the user in a given session, it could no longer be proposed within the same session.

Fig. 2 shows the resulting mean session length of boredom-aware methods against the baselines. Each box-and-whiskers plots results from 117 users, each tested 50 times. In all cases, boredom-aware learning algorithms converged to higher session length. The improvements are quite clear: learning boredom constraints leads to improved engagement.

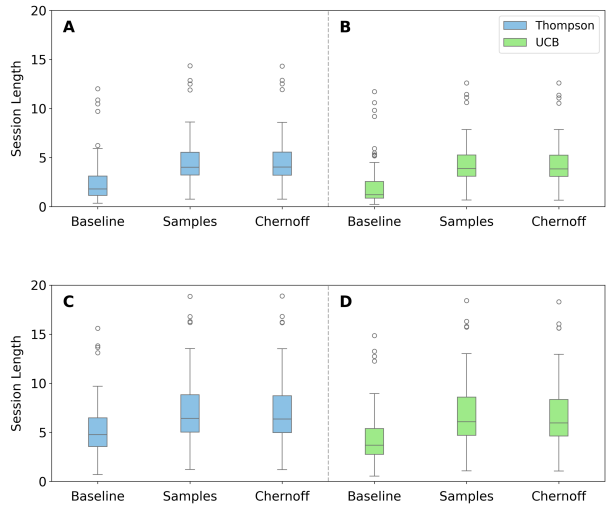

Figure 2: Mean resulting session lengths using the baseline and boredom-aware algorithms. Each quadrant compares the baseline against its boredom-aware variants. Top panes (A, B) show results with spacing constraints, while bottom panes (C, D) show the same for capping. Results are further categorized by exploitation strategy: Thompson Sampling (blue; A, C) and UCB1 (green; B, D).

[1]Available publicly, url withheld for anonymity.

Metric	Chernoff Thompson	Chernoff UCB	Samples Thompson	Samples UCB
Session Len.	7.0 (0.65)	6.61 (0.63)	**7.08** (0.65)	6.70 (0.61)
q-Error	0.028	**0.026**	0.031	0.028
t-Error	**0.35**	**0.35**	1.0	1.01
Mean Explor.	1475	1475	**1009**	**1009**
Min Explor.	799	799	**341**	**341**
Max Explor.	2013	2011	1573	**1571**

(a) Capping. Baseline UCB: 4.34 (0.53), Thompson 5.22 (0.62).

Metric	Chernoff Thompson	Chernoff UCB	Samples Thompson	Samples UCB
Session Len.	**4.58** (0.39)	4.31 (0.40)	**4.58** (0.39)	4.33 (0.40)
q-Error	0.028	**0.025**	0.029	0.026
t-Error	**0.36**	**0.36**	0.81	0.80
Mean Explor.	677	677	**562**	**562**
Min Explor.	369	369	**428**	**428**
Max Explor.	1049	1050	**688**	689

(b) Spacing. Baseline UCB: 2.08 (0.3), Thompson 2.49 (0.27).

Table 1: Performance comparison of t-learning (exploration) and preference (q) learning (t-exploitation) variants under different boredom constraints. Metrics above the horizontal divider reflect preference (q) learning performance, while metrics in the lower part correspond to t-learning performance. Standard deviations are shown in parentheses.

Improvement compared to the baselines. We examine the results in more detail. Table 1 shows mean results from the experiments (Top: capping experiments; bottom: spacing experiments). Each of the two sub-tables shows results in each performance metric (leftmost column), for each of the learning algorithms (remaining columns). Session length is defined as the number of accepted items per session. q-error and t-error are measured as absolute error. Exploration is measured by the number of sessions. Baseline results are shown in the captions.

The table show dramatic improvements in the principal performance measure (session length, first row in each sub-table). Table 1b shows the UCB-based variants *more than double the mean session length* in the sampling experiments (4.31 for Chernoff UCB, 4.33 for Samples UCB, both up from 2.08 for the UCB baseline). The Thompson variants are at 4.54, up from 2.49. Table 1a shows clear improvements as well, by approximately two items per session on average (slightly more than two in the UCB variants; slightly less in the Thompson variants). The standard deviations have increased somewhat, more so for the UCB variants then for the Thompson variants.

Comparing the *best*-performing baseline against the *worst*-performing boredom learning method, demonstrates the significance of the improvements. A paired t-test (two-tailed) yields $p < 0.00001$ for both capping and spacing.

Comparison of preference (q) learning Algorithms. We focus next on the impact of the preference-learning algorithms (the t-exploitation phase), comparing *UCB* and *Thompson sampling*, regardless of the t-learning (exploration) mechanism (*Chernoff* or *Samples*). The relevant results are in columns 3, 5 (from the left) for UCB variants, and in columns 2, 4 for Thompson variants.

The results reveal that in both sets of experiments, the session lengths are higher when using the Thompson variants (e.g., 4.58 using Thompson, compared to 4.31/4.33 using UCB, in Table 1b). Examining the mean error in predicting q value of items (user preference), the trend is reversed: UCB results in slightly higher for Thompson-based variants, indicating marginally lower accuracy relative to UCB. However, the errors themselves are negligible in comparison to the improvements in the session length.

Associated error in predicting t, as well as other metrics that appear below the horizontal divider in the tables are not affected by the algorithm choice for the preference (q) learning phase.

5.3 Duration of Exploration Until t is Known

We now turn to examine factors influencing the duration of the t-learning (exploration) phase.

Spacing vs. Capping Boredom Constraints. The visual presentation of the results in Fig. 2 shows a qualitative difference in the session lengths in capping vs. spacing experiments, achieved under the various conditions. To examine these more closely, we look at the bottom set of metrics in Tables 1a and 1b. The row marked t Error measures the mean error in predicting the correct t threshold of the different items. The mean, minimum and maximum exploration lengths (across all users and items) are shown next.

In both sets of experiments, the algorithms successfully learned the underlying boredom thresholds and converged to stable session lengths. Nevertheless, learning under capping appears more challenging. Exploration phases are nearly twice as long as under spacing (e.g., 1475 vs. 677 steps with Chernoff–UCB (Tables 1a and 1b).

These performance differences result from the interpretation of the threshold t in the different boredom constraint settings. For example, with $t = 7$ in a 14-session window, spacing permits at most two acceptances, whereas capping permits up to seven. The same threshold is therefore more restrictive in spacing, which naturally leads to shorter sessions (just over 4) compared to capping (around 7).

Impact of Exploration Method. We compare the two t-learning exploration strategies used to estimate boredom thresholds, given in Algorithm 3 (heuristic *Samples*) and Algorithm 4 (*Chernoff* bounds). As shown in Tables 1a and 1b, the sample-based approach consistently resulted in shorter exploration length (e.g., 1009 vs. 1475 for capping, and 562 vs. 677 for spacing). These advantages of the Samples method were present across the entire set of items, as evident from the crisp advantage also in the minimum (easiest t threshold to learn) and maximum (hardest t to learn) exploration lengths.

The t errors were clearly higher for the Samples method. However, given that the q-error values were fairly similar, and that high session lengths could be achieved using the Samples method (with Thompson used once t is known), the t errors have had no impact in practice. Indeed, the observed differences in the session lengths achieved by Chernoff-UCB vs. Samples-UCB, and Chernoff-Thompson vs. Samples-Thompson were statistically negligible.

Overall, the efficiency of the sample-based t-learning method of Algorithm 3—requiring fewer exploration steps—coupled with the performance of the Thompson learning algorithm during the t-exploitation (user preference) learning stage makes *Samples-Thompson* the best choice.

5.4 Reducing the Exploration Length

We remind the reader that the results above were generated under *cold start* conditions, where no prior information is used to speed up the t-learning phase. Nevertheless, even using Samples-Thompson combination, the exploration length remains a concern for practical use. We evaluate several approaches to reducing the exploration length.

Max-q Tie-Breaking (MTB). One way to quicken the t exploration stage is by changing the tie-breaking rule used by the action selection scheduler in Algorithm 2, line 1. As described above, in the case of ties, the scheduler greedily prefers recommendations that are more *informative*: they have been tried less times. Instead, one may wish to select actions that have the highest current q-estimation, allowing high-potential items to begin exploitation sooner ("scheduling by q"). This is a general method that requires no user- or item-specific information

Prior-Knowledge Boosting (PKB). A different method relies on prior (approximate) knowledge of the t value for the given user, setting informative lower and upper bounds on t. This requires an external source of information, such as by collaborative filtering techniques.

To evaluate the potential of these learning acceleration methods, we conducted an experiment using a single representative user profile (to allow use of PKB, which is user-specific). The results are shown in Table 2. The column marked *Basic* shows results from Samples-Thompson, using the existing methods described above. The column titled *PKB* shows the results when using the same tie-breaking rule as above, but where the lower- and upper- bounds of each item were initialized to be within 8 sessions of each other, with the ground-truth t in the middle. The column titled *MTB* shows the results without PKB, but where the tie-breaking rule was changed to scheduling by q. The last column (*Combined*) shows the result when both PKB and MTB are used together.

Metric	Basic	PKB	MTB	Combined
Session Len.	8.23 (0.34)	8.23 (0.35)	8.22 (0.33)	8.21 (0.35)
q-Error	0.052	0.016	0.019	0.016
t-Error	0.53	0.328	0.589	0.298
Mean Explor.	519	379	416	299
Min Explor.	422	262	237	129
Max Explor.	631	502	662	448

Table 2: Exploration optimization methods: Prior-Knowledge Boosting, Max-q Tie-Breaking, and both methods combined.

The results demonstrate not only the efficacy of each method by itself, but also in combination. The PKB method reduces mean exploration from 519 to 379 sessions without compromising session length (8.23) or accuracy (in t and q error measures). MTB reduces mean exploration to 416 by prioritizing high-value items for earlier exploitation, thereby sustaining a higher average session length during the learning period. The two methods are evidently complementary. The *Combined* configuration yields results better than its component methods', reducing mean exploration to 299 sessions while achieving almost identical mean session length (see also min. and max. exploration lengths).

5.5 Scalability to Larger Catalogs

To evaluate how well the approach generalizes to broader recommendation settings, we expanded the action space from 42 to 149 items. We tested the proposed Samples–Thompson algorithm against Thompson Sampling baseline, under both spacing and capping boredom constraint.

Table 3 shows the results. Contrasting the session lengths for both capping and spacing constraints show that the Samples-Thompson algorithm offers clear improvements over the baseline, when the catalog is increased in size. The session length doubles for spacing constraints, and nearly doubles (factor of 1.8) for capping constraints. This comes at a cost of longer t-exploration period (see previous section with respect to methods for countering this cost).

Method	Capping	Spacing
Samples–Thompson	**25.07** (1.49)	**14.49** (0.77)
Thompson (baseline)	14.37 (1.45)	7.33 (0.59)

Table 3: Performance of boredom-aware exploration methods compared to standard Thompson sampling baseline, with 149 items catalog. Standard deviations are shown in parentheses.

5.6 Evaluation with Real Production Constraints

We next evaluate the proposed approach under realistic production conditions. Specifically, we incorporate the actual spacing constraints currently deployed in the ElliQ system, and apply them within the simulator using the full 149-item catalog. The constraints are modeled with a two-day boredom window (24 sessions), closely reflecting the operational environment. This experiment is conducted on a single representative user to enable controlled analysis.

We compare the *Samples–Thompson* variant against a standard Thompson Sampling baseline. The results show that the boredom-aware algorithm maintains a clear advantage even under these realistic constraints, achieving a higher session length of **12.97** compared to **9.62** for the baseline - an improvement of 34.8%. The boredom-aware method also achieved a lower q-error value - 0.025 compared to 0.149 for the baseline.

Importantly, this experiment reinforces—and further strengthens—the conclusion from the large catalog setting: explicitly modeling boredom remains beneficial even when constraints are no longer synthetic but derived from real system behavior. Despite the increased complexity and rigidity of production constraints, the algorithm successfully adapts and continues to improve user engagement.

6 DISCUSSION AND CONCLUSION

Sequential recommendation in sessions, with a fixed (small) catalog, is an important challenge: user engagement is only as good as the last recommendation, and repetition is both inherent and unwanted. We presented a two-stage approach for learning user boredom constraints while enforcing them within the recommendation process. The results clearly demonstrate the benefit of the proposed learning approach: inherent repetition is managed, so that user boredom is minimized. Results from extensive experiments with real-world user data show dramatic improvements in session length, and a number of techniques are explored for improving on these results (e.g., reducing the number of learning steps).

A number of challenges are raised, which we intend to pursue in future work. First, incorporating state-based dynamics while preserving boredom-aware filtering may reveal (and allow utilization of) additional factors impacting user preferences, such as boredom evolution within a session, depending on its internal sequence. A Constrained Markov Decision Process (CMDP) [27] may be an appropriate basis for this investigation. Second, an alternative direction is to learn both t and q jointly, for example through selective updates in which different interaction outcomes trigger updates either to the preference model or to the constraint model. Such a formulation may improve sample efficiency and better capture the interaction between user preferences and boredom effects.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Elham Asani, Hamed Vahdat-Nejad, and Javad Sadri. 2021. Restaurant recommender system based on sentiment analysis. *Machine Learning with Applications* 6 (2021), 100114.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.

[4] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[5] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. 2024. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems* 125 (2024), 102427.

[6] Keith Bradley and Barry Smyth. 2001. Improving Recommendation Diversity. *Proc. AICS '01* (09 2001), 75–84.

[7] E Broadbent, K Loveys, G Ilan, G Chen, MM Chilukuri, SG Boardman, PM Doraiswamy, and D Skuler. 2024. Elliq, an ai-driven social robot to alleviate loneliness: progress and lessons learned. *JAR life* 13 (2024), 22.

[8] Abhijnan Chakraborty, Saptarshi Ghosh, Niloy Ganguly, and Krishna P Gummadi. 2017. Optimizing the recency-relevancy trade-off in online news recommendations. In *Proceedings of the 26th International Conference on World Wide Web*. 837–846.

[9] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 815–824.

[10] Gangan Elena, Kudus Milos, and Ilyushin Eugene. 2021. Survey of multiarmed bandit algorithms applied to recommendation systems. *International Journal of Open Information Technologies* 9, 4 (2021), 12–27.

[11] Rong Hu and Pearl Pu. 2011. Helping users perceive recommendation diversity.. In *DiveRS@RecSys*. 43–50.

[12] Syahril Anuar Bin Idris, Syuhada Binti Abdul Rahman, Fathin Najihah Atikah Bt Mohd Farid, Suhaila Binti Abdul Rahman, Raidah Binti Yazid, and Nor Azlinah Binti Md Lazam. 2025. AI-Assisted Personal Wardrobe Management: A Human-Centric Approach to Intelligent Clothing Organization and Style Recommendation. In *2025 IEEE 6th International Conference in Robotics and Manufacturing Automation (ROMA)*. IEEE, 236–241.

[13] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).

[14] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. 2015. Just in time recommendations: Modeling the dynamics of boredom in activity streams. In *Proceedings of the eighth ACM international conference on web search and data mining*. 233–242.

[15] Sinan Keskin and Halil Yurdugül. 2022. E-learning experience: Modeling students'e-learning interactions using log data. *Journal of Educational Technology and Online Learning* 5, 1 (2022), 1–13.

[16] Yehuda Koren, Steffen Rendle, and Robert Bell. 2021. Advances in collaborative filtering. *Recommender systems handbook* (2021), 91–142.

[17] Chintoo Kumar, C. Ravindranath Chowdary, and Ashok Kumar Meena. 2024. Recent trends in recommender systems: a survey. *International Journal of Multimedia Information Retrieval* 13, 4 (Oct. 2024), 41. https://doi.org/10.1007/s13735-024-00349-1

[18] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems–A survey. *Knowledge-based systems* 123 (2017), 154–162.

[19] Yang Li, Kangbo Liu, Ranjan Satapathy, Suhang Wang, and Erik Cambria. 2024. Recent Developments in Recommender Systems: A Survey [Review Article]. *IEEE Computational Intelligence Magazine* 19, 2 (2024), 78–95. https://doi.org/10.1109/MCI.2024.3363984

[20] Martin Mladenov, Chih-Wei Hsu, Vihan Jain, Eugene Ie, Christopher Colby, Nicolas Mayoraz, Hubert Pham, Dustin Tran, Ivan Vendrov, and Craig Boutilier. 2021. Recsim ng: Toward principled uncertainty modeling for recommender ecosystems. *arXiv preprint arXiv:2103.08057* (2021).

[21] Fernando Mourão, Claudiane Fonseca, Camila Souza Araujo, and Wagner Meira Jr. 2011. The Oblivion Problem: Exploiting Forgotten Items to Improve Recommendation Diversity.. In *DiveRS@ RecSys*. 27–34.

[22] Jeff M Phillips. 2012. Chernoff-hoeffding inequality and applications. *arXiv preprint arXiv:1209.6396* (2012).

[23] Dhanya Pramod and Prafulla Bafna. 2022. Conversational recommender systems techniques, tools, acceptance, and adoption: A state of the art review. *Expert Systems with Applications* 203 (2022), 117539. https://doi.org/10.1016/j.eswa.2022.117539

[24] Shaina Raza, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi, Ananya Raval, Farshad Navah, and Amirmohammad Kazemeini. 2024. A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice. *arXiv preprint arXiv:2407.13699* (2024).

[25] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten De Rijke. 2019. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4806–4813.

[26] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning* 11, 1 (2018), 1–96.

[27] Rahul Singh, Abhishek Gupta, and Ness B Shroff. 2022. Learning in constrained Markov decision processes. *IEEE Transactions on Control of Network Systems* 10, 1 (2022), 441–453.

[28] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[29] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2019. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 6332–6338.

[30] Romain Warlop, Alessandro Lazaric, and Jérémie Mary. 2018. Fighting boredom in recommender systems with linear reinforcement learning. *Advances in Neural Information Processing Systems* 31 (2018).

[31] Tom Zahavy, Brendan O'Donoghue, Andre Barreto, Volodymyr Mnih, Sebastian Flennerhag, and Satinder Singh. 2021. Discovering diverse nearly optimal policies with successor features. *arXiv preprint arXiv:2106.00669* (2021).

[32] Kesen Zhao, Shuchang Liu, Qingpeng Cai, Xiangyu Zhao, Ziru Liu, Dong Zheng, Peng Jiang, and Kun Gai. 2023. KuaiSim: A comprehensive simulator for recommender systems. *Advances in Neural Information Processing Systems* 36 (2023), 44880–44897.