Dear Reviewers,

An earlier version of this paper was submitted (and rejected) to the AAMAS 2026 conference earlier this year. Below, you will find the reviews received for the earlier version. We took these to heart.

Key points raised by the reviewers, and corresponding changes are as follows:

- Experiments reported in the earlier version used data from only a single user. **We now use data for 117 users.**
- Reviews pointed out that the exploration length (number of sessions to converge) was excessive. We emphasized that this is a result of running cold-start experiments, where no priors are given for the learning process. **A new section (5.4) now explores two complementary methods for dramatically reducing the exploration phase length:** one method requires no additional information, while the other assumes some priors on the user preference and boredom thresholds is given (e.g., from similar users or domain expertise).
- We better positioned the contribution in the context of related work, more specifically **identifying the algorithms as intended for session-based recommendations, where each session is strictly sequential:** session ends upon first rejection. **Discussion of related work and potential applications was much extended**.
- **The algorithms were rewritten** to be much clearer, and **the notation was cleaned up**.
- We removed measures that are not informative (such as acceptance rate, which was a deterministic function of the main measure—session length), and instead provided detailed explanations for the remaining metrics and their significance.

AAMAS 2026 Submission Reviews

Review 1: "Learning and Enforcing User Boredom Constraints" Review

Summary:
This paper presents a reinforcement learning algorithm for improving recommender systems. They attempt to overcome the issue of overly repetitive recommendations that RL recommender systems are susceptible to by increasing recommendation diversity, specifically for recommendation systems with small catalogs. The authors' approach uses boredom constraints functions to filter recommendations that occur too frequently. They evaluate their algorithm by simulating user interactions with a real conversational recommendation robot (ElliQ).

Technical Quality: 4. Very Good
Significance And Relevance: 2. Some significance for an AAMAS subfield
Presentation Quality: 4. Very Good
Review:
The paper is well-written and organized. The authors clearly identify their motivation for improving recommender systems with small catalogs to prevent excessive repetition. However, the significance of the work is not very clear to me. The approach and evaluation appear to be tailored to a specific robot use case and thus it's not clear to me that the paper's findings are

transferable to broader uses. Pros: -Statistically significant improvement of suggestion acceptance rate over baseline algorithm Cons: -The authors note that “these results indicate that integrating boredom awareness leads to more sustainable long-term engagement, even when early learning performance appears less favorable”. However, they do not significantly address this tradeoff of less favorable early learning performance. -The scalability to larger catalogs and transferability to other recommender systems appear to be major limitations

Post-rebuttal: I have read and appreciate the authors’ response. After consideration, I have decided to leave my rating unchanged.

Rating: 5: Marginally below acceptance threshold
Confidence: 2: The reviewer is willing to defend the evaluation, but it is quite likely that the reviewer did not understand central parts of the paper

Review 2: Well-written paper, but I have doubts about how well the simulation mirrors reality as well as the overall practical value

Summary:
The authors explore an aspect of recommender systems that is not usually taken into account: the fact that people's choices may be time-dependent due to a desire for variety. They present an algorithm that learns users' preferences along the traditional dimension of intrinsic interest or likability and boredom -- a measure of the user's need for variety, which is particularly of relevance when the number of options in the catalog is small. They study the effect of a few different approaches to managing the tradeoff between exploration and exploitation in terms of the acceptance rate and the session length, using a simulator that emulates a real commercial system that interacts with elderly patients, offering a small catalog of different interaction plans. They find that taking boredom into account can somewhat improve the overall acceptance rate and session length, while the impact on the amount of time it takes for the algorithm to converge is greater.

Technical Quality: 3. Acceptable
Significance And Relevance: 2. Some significance for an AAMAS subfield
Presentation Quality: 4. Very Good
Review:
The paper is well-motivated and clear. The problem is of moderate interest within the realm of recommender systems with small catalogs of actions, such that the user may become bored with repetition. The results show that there are moderate benefits of incorporating boredom into the set of recommended plans.

I wish the actual application had been explained in a bit more detail. Could the authors provide an example or two of the items in the catalog? Are these conversation topics proposed by the system? How similar are the 42 different items in the catalog, i.e. could two different items be considered close enough in topic that they shouldn't be offered in close succession? If so, it seems like the treatment of boredom would have to consider the topic and not just the identity of the item.

I have some concerns about the realism of the experiments, given that boredom is apparently assessed item-by-item as opposed to being treated as topic-related (if I understand correctly; at

least I was led to believe this by the example in Figure 1). Also, as the authors admit, only a single user base profile is used. Other aspects of the simulation seem like they might be rather coarse reflections of the real system.

I also have concerns about practicality. Given that convergence typically takes hundreds to thousands of sessions, what time scale does that translate to in real life? If there are just a few sessions per day, this suggests that the convergence would take months or even years. That doesn't seem practical. Perhaps I'm missing something; I'd be glad to hear what the authors have to say about this.

Rating: 4: Ok but not good enough - rejection
Confidence: 4: The reviewer is confident but not absolutely certain that the evaluation is correct

Review 3: Good paper

Summary:
The work focuses on the domain of recommender systems. Specifically, these systems often struggle with recommendation variety to maintain user engagement, especially systems that rely on reinforcement learning. The authors introduce a novel RL algorithm that improves recommendation variety in small-catalog settings, where repetition is required. The algorithm explicitly models per-user, per-item boredom thresholds, independently from the expected acceptance rate. This allows repeated recommendations of an item, appropriately spaced in time, with a high expected acceptance rate. The problem is modeled as a Constrained Multiarm Bandit. The proposed approach has been evaluated experimentally, and the results hint at a higher acceptance rate and a longer session length.

Technical Quality: 3. Acceptable
Significance And Relevance: 3. Significant for an AAMAS subfield
Presentation Quality: 4. Very Good
Review:
I believe that the paper is a solid contribution to a relevant subfield of AAMAS. The contributions are novel, clearly explained, and the experimental evaluation (while limited) is sufficient to show improvements on the considered metrics. However, I possess low expertise on the topic. Some possible points of improvement are listed in the following:

- One possible point of improvement is to conduct a user experiment, involving actual users, to assess their perception of the system, rather than relying exclusively on user experiments.
- Similarly, the paper would benefit from a comparison with other approaches that aim to increase user engagement in recommender systems or other kinds of diversity constraints.
- Furthermore, providing a replication package of the work, with the code and data used for the experiments, would foster replicability and verifiability.

Rating: 7: Good paper, accept
Confidence: 1: The reviewer's evaluation is an educated guess

Learning and Enforcing User Boredom Constraints in a Conversational Recommender System

Anonymous Author(s)
Submission Id: 235

ABSTRACT

Recommender systems often struggle with recommendation variety, to maintain user engagement. Systems using reinforcement learning (RL) to personalize recommendations are particularly susceptible, due to *exploitation*, repeating previously-successful recommendations. Existing approaches for increasing variety typically assume large catalogs, detailed item metadata, or explicit user feedback; these do not hold in many real-world scenarios. We introduce a novel RL algorithm that improves recommendation variety in small-catalog settings, where repeatition is required. The algorithm explicitly models per-user, per-item boredom thresholds, independently from the expected acceptance rate. This allows repeated recommendations of an item, appropriately spaced in time, with high expected acceptance rate. Modeling the problem as a Constrained Multiarm Bandit, learning works for each user in two stages: First, item- specific boredom thresholds are determined efficiently via a probabilistic binary search. Then, the true acceptance rate is continually improved via familiar multiarm bandit algorithms. The results, evaluated on data from a commercial system, demonstrate clear improvements in acceptance rate and session length.

KEYWORDS

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

ACM Reference Format:
Anonymous Author(s). 2026. Learning and Enforcing User Boredom Constraints in a Conversational Recommender System. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026,* IFAAMAS, 9 pages.

1 INTRODUCTION

Recommender systems are widely used to personalize content recommendations [3, 11, 13, 17]. A fundamental challenge is to vary recommendations (recommendation diversity) while targeting user preferences based on past accepted recommendations [12]. 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 interaction in recommender systems.

Many approaches to promoting diversity typically assume large catalogs, where rich item metadata, item- or user- similarity, or explicit user feedback can be brought to bear [e.g., 9, 20, 21]. Such data enables discovering unseen items that match latent user preferences. Section 2.2 provides a detailed review of prior work.

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. This is a significant challenge to recommendation diversity. As a motivating example, consider ElliQ [5], 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* [16] 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 inherent 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 systems with inherent repetitions, by explicitly modeling—and *learning*—user response as a function of time. The response is 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 [20] 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 learning method using a simulator designed to replicate user-system interactions, with user profiles and a catalog from commercial real-world data. The simulator enables controlled comparison against classical baselines such as UCB1 [2] and Thompson Sampling [18]. The results show that the two-stage learning algorithms surpass these baselines achieving superior performance in converged acceptance rate and session length.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

Learning User Boredom Constraints in Sequential Recommender Systems

Anonymous Author(s)
Submission Id: 50

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.

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

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 (multiple) user interaction histories [1, 3, 11, 13]. A central challenge in recommender systems is recommendation diversity, preventing repetitive suggestions that reduce user satisfaction [12].

While traditional approaches such as collaborative filtering exploit similarities between users [10], recent work employs reinforcement learning (including but not limited to multi-arm bandit models) to capture patterns in the sequential decision-making of each user and adapt to their preferences [1, 7]. 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 task of the recommender system, or when the item catalog is limited.

In Section 2.1 we present a motivating study of a successful beta-experiment with real users, using a multi-arm bandit recommender applied to a commercially deployed conversational robot. 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 [5] is to maintain user engagement over time, promoting cognitive and physical health and decreasing loneliness in elderly users. 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 Contextual UCB, where the context is defined by two time-of-day states: morning and afternoon. In this formulation, the algorithm maintains two independent bandit models, one for each state, without any transition dynamics between them. The boredom constraints were enforced in both states: only plans satisfying the 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. 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 [8].

In a widely used one, diversity is the average dissimilarity between items in a recommendation set [4]. 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 [20]. Both assumptions fail in conversational systems (including the system described above), where suggestions are made sequentially and item metadata is scarce [16].

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 seemingly shifting feedback [12]. Mourão et al. [14] proposes that finding once relevant items to the user and again suggesting them will increase diversity without compromising accuracy. In a sense, 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, Kapoor et al. [9] 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.

3 DIVERSITY CONSTRAINTS

We model the conversational 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.

We distinguish 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*), defined as a sequence of recommendations that terminates on the first user decline. 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. In the *boredom region*

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'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 is 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.

3 DIVERSITY CONSTRAINTS

We model the conversational 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.

We distinguish 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*), defined as a sequence of recommendations that terminates on the first user decline. 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

(time<t), the item is deterministically rejected. In the *engagement region* (time>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. For example, curve **A** corresponds to an item with $t = 3$ and $q = 1$. Assuming spacing is measured in sessions, with 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.

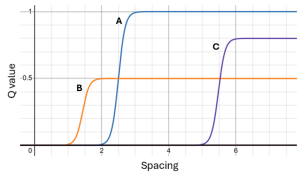

Figure 1: Example of user behavior towards items under spacing constraint. Each curve corresponds to a sigmoid with different boredom thresholds (t) and preference values (q).

Note that *capping* exhibits a sigmoidal pattern in the opposite direction to *spacing*. To maintain consistency between the two boredom measures, we redefine capping by its complementary form, representing 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* as t increases, unfiying algorithmic treatment (see below).

Although we describe the response using a sigmoid, it should be understood as an approximation of a step function. Since boredom values are discrete rather than continuous, 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 [6, 21]. Nonetheless, the framework can extend to any monotonically increasing function capturing the boredom–engagement transition.

4 LEARNING USER BOREDOM THRESHOLDS

The sigmoidal formulation offers a practical advantage: it enables separate learning of the boredom threshold t and the preference parameter q. Specifically, t can be estimated first by identifying the point acceptance transitions from low to high, without requiring precise knowledge of q. Once t is determined, it can be enforced as a hard constraint during the learning of q, improving stability and accuracy by decoupling the estimation of the two parameters.

We therefore adopt a two-stage approach, utilizing a multi-armed bandit as its backbone (Algorithm 1). The algorithm is called with every new session, and maintains two lists of actions (items) between calls: 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. All items are initially placed in the explore list.

Algorithm 1: Action Selection Framework

Input: *explore_list*, *exploit_list*

1 **foreach** *time step* $s = 1, 2, \ldots$ **do**
2 $A_s \leftarrow$ *AvailableActions*(*explore_list*)
3 **if** A_s *not empty* **then**
4 $a_s \leftarrow$ *ExploreAlg.SelectActionAndUpdate*(A_s)
5 **if** *IsCertain*(a_s) **then**
6 *REMOVE*(*explore_list*, a_s)
7 *ADD*(*exploit_list*, a_s)
8 **else**
9 $A_s \leftarrow$ *AvailableActions*(*exploit_list*)
10 **if** A_s *not empty* **then**
11 $a_s \leftarrow$ *ExploitAlg.SelectActionAndUpdate*(A_s)
12 **if** a_s *was rejected* **then**
13 return

At each decision step, items available for exploration are determined using their current boredom values (line 2). The threshold exploration algorithm (Algorithm 2, see below) selects and recommends an item, updates its statistics (line 4) based on the response (accept/reject); a rejection terminates the session. On acceptance, the algorithm checks whether t is confidently known (line 5); if so, the item is transferred to the *exploit* list.

When there are no items available for exploration, selection proceeds via the exploitation algorithm using the items in the *exploit* list (line 11). We use classic multi-arm bandit algorithms for this process, though their arms (potential items) are limited to those whose learned t values are satisfied (line 9). Their own exploration policies now continue, learning the q value for the item. If no item can be selected (e.g., none satisfy the boredom constraint), no recommendation is made (*null* item) presented in Algorithm 2 (line 2). In the second stage, once thresholds are estimated, they are enforced as hard constraints. At this point, basic versions of UCB1 and Thompson Sampling are employed for exploitation, while exploration continues over the q-values.

Exploration of boredom thresholds. 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 $\theta[a]$ is obtained. Since responses are noisy, the algorithm takes rejections and acceptance as evidence (Lines 6-7).

Each item's boredom range is initialized to $[0, w]$, with the midpoint as an initial estimate. In each step, a *scheduler* selects the next recommendation based on the current bounds (line 1). It prioritizes items due for testing and those expected to yield the most informative feedback. If none is available, the scheduler idles (line 2). Upon observing user feedback, the action's current boredom value is retrieved (line 5) and used to update (*action*, *boredom*) statistics.

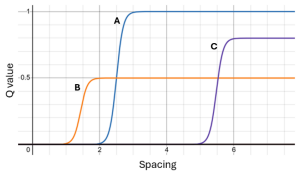

Figure 1: Example of user behavior towards items under spacing constraint. Each curve corresponds to a sigmoid with different boredom thresholds (t) and preference values (q).

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, unfiying 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)

repeat
  $A \leftarrow$ AvailableActions(*explore_list*)
  if A *not empty* **then**
  $a \leftarrow$ ExploreAction(A);
  if *IsCertain*(a) **then**
  Remove(*explore_list*, a) ;
  Add(*exploit_list*, a)
  else
  $A \leftarrow$ AvailableActions(*exploit_list*)
  if A *not empty* **then**
  $a \leftarrow$ ExploitAction(A);
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.

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:

Algorithm 2: Exploration Algorithm

Input: Items A, boredom window w, method $\mathcal{M}$

1 $a_s \leftarrow$ *Scheduler.SelectAction*(A)
2 **if** $a_s = \emptyset$ **then**
3 **return** a_s
4 Execute a_s, observe reward r_s
5 $b \leftarrow \min(get\_boredom(a_s), w)$
 `// Update statistics`
6 $total[a_s, b] \leftarrow total[a_s, b] + 1$
7 $accepts[a_s, b] \leftarrow accepts[a_s, b] + r_s$
 `// Re-estimate threshold using method` $\mathcal{M}$
8 $(lower[a_s], upper[a_s]) \leftarrow \mathcal{M}(a_s, total, accepts)$
9 $\hat{\theta}[a_s] \leftarrow \frac{1}{2}(upper[a_s] + lower[a_s])$
10 **return** a_s

Then, bounds are updated using $\mathcal{M}$, the selected constraint confidence estimate variant (line 8)—discussed below. The search is restricted to the last w sessions. If the true boredom threshold lies outside the window, the algorithm assumes it equals w, causing the item to be rarely suggested. The preference value q is estimated online from the observed samples, but confidence in its value is not assessed: its estimated value will be further learned when it is on the exploit list.

Constraint confidence estimation methods $\mathcal{M}$. The task of the constraint confidence estimation method is to update the lower and upper bounds on the estimated t, if known with sufficient confidence. We present two variants, using the following parameters to balance reliability and efficiency: requiring more samples of the same bound makes exploration longer, but increases reliability.

- The *success rate threshold* τ determines the minimum acceptance probability at which an item is considered valid for recommendation at a given boredom level.
- The *sample size* n controls how many user interactions are observed at each boredom level before a decision is made. Larger n reduces variance but slows down exploration.
- The *error tolerance* ε specifies the maximum deviation allowed between the estimated acceptance rate and the true acceptance rate.
- The *confidence level* δ determines the probability that the acceptance rate estimate lies within the tolerance ε. A higher δ value requires stronger evidence before bounds are updated.

Sample-Based Exploration (Algorithm 3). This method updates thresholds directly based on empirical acceptance rates, once the value has been examined sufficiently according to the sample size n parameter. When the observed rate for a given boredom level b consistently falls below the target threshold τ, the lower bound is increased (line 5); conversely, when it remains above τ, the upper bound is decreased (line 7).

Chernoff-Based Exploration (Algorithm 4). This method refines the update rule by utilizing the Chernoff-Hoeffding bound [15] to account for statistical uncertainty. The bound estimates confidence as

$$\delta \leq 2\exp(-2n\varepsilon^2),$$

Algorithm 3: Sample-Based Update $\mathcal{M}_{sample}$

1 **for** $b' \in [0, w]$ **do**
2 **if** *certainty reached for* $(total[a_s, b'], accepts[a_s, b'])$ **then**
3 $rate \leftarrow accepts[a_s, b'] / total[a_s, b']$
4 **if** $rate < \tau$ *and* $b' > lower[a_s]$ **then**
5 $lower[a_s] \leftarrow \min(b', upper[a_s])$
6 **else if** $rate \geq \tau$ *and* $b' < upper[a_s]$ **then**
7 $upper[a_s] \leftarrow \max(b', lower[a_s])$

where n is the number of samples and ε is the error tolerance. Here, empirical acceptance rates are evaluated against the target τ under confidence parameters $(\varepsilon_{low}, \varepsilon_{high}, \delta_{low}, \delta_{high})$. In line 6, the confidence of the current boredom value is calculated. Bounds are updated only when the observed evidence meets the required confidence level (line 10) and stopping when the confidence is high enough (line 12), ensuring tighter probabilistic guarantees.

Algorithm 4: Chernoff-Based Update $\mathcal{M}_{chernoff}$

1 $rate \leftarrow accepts[a_s, b] / total[a_s, b]$
2 **if** $rate < \tau$ **then**
3 $\varepsilon \leftarrow \varepsilon_{low}$
4 **else**
5 $\varepsilon \leftarrow \varepsilon_{high}$
6 $conf[a_s, b] \leftarrow \text{Chernoff}(total[a_s, b], \varepsilon)$
 `// Update upper bound`
7 $ub \leftarrow w + 1$, $maxConf \leftarrow -1$
8 **for** $b' \in [0, B_{max}]$ **do**
9 $rate' \leftarrow accepts[a_s, b'] / total[a_s, b']$
10 **if** $rate' > \tau$ ***and*** $conf[a_s, b'] > \max(maxConf, \delta_{low})$ **then**
11 $ub \leftarrow b'$; $maxConf \leftarrow conf[a_s, b']$
12 **if** $conf[a_s, b'] > \delta_{high}$ **then**
13 break
14 $upper[a_s] \leftarrow ub$
 `// Update lower bound`
15 $lb \leftarrow -1$, $maxConf \leftarrow -1$
16 **for** $b' \in [0, upper[a_s]]$ **do**
17 $rate' \leftarrow accepts[a_s, b'] / total[a_s, b']$
18 **if** $rate' \leq \tau$ ***and*** $conf[a_s, b'] \geq \max(maxConf, \delta_{low})$ **then**
19 $lb \leftarrow b'$; $maxConf \leftarrow conf[a_s, b']$
20 $lower[a_s] \leftarrow lb$

5 EXPERIMENTS AND RESULTS

We report on a comprehensive evaluation of the boredom constraint learning approach through a series of experiments designed to assess its effectiveness, robustness, and sensitivity to design choices.

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

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 $\mathcal{S} \leftarrow \{b' \mid b' \in [0, w], \delta_{low} \leq d(a, b') \leq \delta_{high}, \text{rate}(a, b') \geq \tau\}$
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 $(\varepsilon_{low}, \varepsilon_{high})$. 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.

[1] Available publicly, url withheld for anonymity.

We present the experiment setup in Section 5.1. We then benchmark the proposed approach against baseline multi-armed bandit algorithms (Section 5.2). We contrast the confidence estimation methods (heuristic sampling and Chernoff-based) in Section 5.3, and contrast different exploitation algorithms in Section 5.4. Next, we investigate the role of boredom metrics by contrasting spacing- and capping-based formulations (Section 5.5). Finally, we extend the analysis to scenarios with a larger item catalog (Section 5.6) to test the scalability and generalizability of the approach.

5.1 Experimental Setup

Evaluating recommendation algorithms requires repeated experimentation under controlled conditions, which are rarely feasible with live users. As a result, simulation platforms have become a standard tool for benchmarking. We developed a simulator grounded with anonymized 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 simulator reflects realistic interaction dynamics while preserving the low-information constraints of the deployed system. Each simulated user was assigned a fixed profile defined by a global acceptance rate and two static item sets: *consistently accepted* and *consistently rejected*. The simulator incorporates boredom modeling via spacing and capping constraints, and user behavior was parameterized by boredom thresholds and preference values. Simulated responses followed a hybrid model: items in the consistent sets produced fixed outcomes, while others were deterministically rejected below their boredom threshold and accepted above it with probability equal to the product of user- and item-level acceptance rates. This determines the user-item q value. Unless otherwise stated, experiments used a catalog of 42 items.

Evaluation Metrics. Each experiment was repeated 50 times with 3,000 simulated sessions per run, using either spacing or capping constraints exclusively. We report results using two complementary evaluation metrics: mean *acceptance rate*, which measures the immediate probability of a suggestion being accepted, and mean *session length*, which captures user engagement in a sequence of recommendations. Both values are measured over the last 500 sessions after the algorithm has converged.

Boredom Constraints. Item-specific boredom constraints were imposed under two configurations: a synthetic setting, with thresholds uniformly sampled within a 14-session (7-day) window, and thresholds from the commercial production spacing rules over a 24-session (2-day) window.

Because the production logs already incorporated manually coded diversity constraints, acceptance rates may be biased upward relative to unconstrained systems.

Parameters. We conducted a series of experiments to determine appropriate parameter values. The parameters fall into four categories: boredom constraints, general confidence parameters, and the confidence estimation method parameters - Chernoff and sampling:

- **Boredom Constraints.** These include w (arbitrarily set at 14). q values were determined as described above. t values were randomly (uniformly) selected for the 42 items.
- **General confidence parameters**. *The success rate threshold* τ defines the preference probability threshold that separates the boredom and engagement regions. Items with acceptance probability below this cutoff are considered to fall in the boredom region. We tested cutoff values between 0.05 and 0.20. The effect on session length and acceptance rate was negligible, but lower cutoffs led to slightly shorter exploration length. Therefore, we used the 0.05 value.
  The stopping rule for exploration (Algorithm 1, line 5) determines when $\bar{t}$ is reliably known. It is defined as the target difference between the upper and lower bounds of the current search range. We evaluated values between 1 and 4. Smaller differences reduced the average error in boredom threshold estimation but increased exploration length. To demonstrate the potential of the algorithm, we chose the most conservative value (1), prioritizing accuracy over learning efficiency. In practical deployments, larger values (e.g., 3) may be more appropriate to balance efficiency and accuracy.
- **Chernoff Parameters**. We applied separate values for ε in the boredom and engagement regions, testing $\varepsilon \in [0.10, 0.25]$ for the boredom region and $\varepsilon \in [0.30, 0.45]$ for the engagement region. The asymmetry reflects the broader range of engagement probabilities (up to 0.95) compared to boredom (≈ 0.05), which makes larger errors tolerable in the engagement region. The combination $\varepsilon_{low} = 0.25, \varepsilon_{high} = 0.45$ yielded the best results across all evaluation metrics.
  We used two δ values to define the confidence interval for updating boredom threshold estimates. The algorithm searches for the most confident boredom values within this interval to update the binary search bounds. We tested $\delta \in [0.05, 0.35]$ for the lower bound and $\delta \in [0.40, 0.80]$ for the upper bound. High δ_{low} values (e.g., ≥ 0.3) significantly degraded performance, while combinations such as $(0.10, 0.40)$ and $(0.10, 0.60)$ for $(\delta_{low}, \delta_{high})$ achieved optimal results. We selected $(0.10, 0.60)$ to provide a wider search range and improve stability.
- **Heuristic Sampling Parameters**. The heuristic parameter n determines how many samples are required for an item at a given boredom value to be considered certain. We tested values between 3 and 10, with both symmetric and asymmetric settings (e.g., requiring more samples in the boredom region). The best performance was obtained with 5 samples for both regions, therefore we used this value in all experiments.

5.2 Comparison with Baseline Methods

We benchmarked the proposed exploration strategies against standard bandit baselines (plain UCB and Thompson sampling), which are not boredom-aware.

Table 1 reports results under both boredom constraints—capping and spacing. For each constraint, four algorithms are presented: the first two correspond to boredom-aware methods, while the remaining two serve as baselines. Performance is evaluated using converged session length and acceptance rate. Best results appear in bold.

For both constraints, examining the last 500 sessions in which the learning has been stabilized, the boredom-aware methods

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 $\bar{t}$). 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.

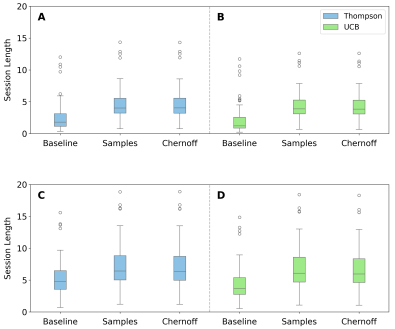

Figure 2: Mean resulting session lengths using the baseline and boredom-aware algorithms. Each quadrant compares the baseline against its boredom-aware variants. Top panes (A, B) show results with spacing constraints, while bottom panes (C, D) show the same for capping. Results are further categorized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

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). 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

Table 1: Performance of boredom-aware exploration methods compared to standard baselines (plain UCB and Thompson sampling). Standard deviations are shown in parentheses. Boredom-aware methods consistently converge to higher long-term performance.

Constraint	Method	Converged Session Length	Converged Acceptance Rate
Capping	Chernoff–Thompson	10.52 (0.57)	**0.913** (0.002)
	Samples–Thompson	10.56 (0.58)	**0.913** (0.001)
	Thompson (baseline)	9.05 (0.56)	0.901 (0.002)
	UCB (baseline)	8.13 (0.52)	0.89 (0.002)
Spacing	Chernoff–Thompson	8.22 (0.33)	0.89 (0.001)
	Samples–Thompson	8.23 (0.34)	**0.892** (0.001)
	Thompson (baseline)	6.25 (0.24)	0.862 (0.002)
	UCB (baseline)	5.9 (0.27)	0.855 (0.001)

achieved higher values in both evaluation metrics, when the samples-thompson variant achieved slightly higher results. Figure 2 of session length over time illustrates the long exploration phase of the boredom-aware algorithm and its convergence to higher values.

These results indicate that integrating boredom awareness leads to more sustainable long-term engagement, even when early learning performance appears less favorable.

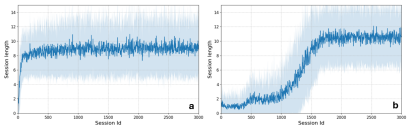

Figure 2: Session length of Thompson Sampling baseline (a) *vs* Sample-based with Thompson exploit (b) using capping constraint.

We further examined the sensitivity of the sample-based Thompson variant using capping to the choice of stopping criterion in the exploration phase. Specifically, we relaxed the conservative stopping rule by reducing the required difference between the upper and lower bounds to 3. This modification accelerated the termination of exploration and, as a result, produced higher total results (session length 9.81, acceptance rate 0.907). Meaning that the method preserved a long-term advantage by converging to a session length which is higher than the baseline Thompson sampling but lower than the more conservative variant. These findings suggest that tuning the exploration stopping rule allows for a principled trade-off between early-phase efficiency and long-term performance.

5.3 Comparison of Exploration Variants

We compare the two exploration strategies used to estimate boredom thresholds: the *Chernoff-based* exploration and the *heuristic-sample-based* exploration. As shown in Tables 2 and 3, the sample-based approach consistently resulted in shorter exploration length (e.g., 888 vs. 1250 for capping, and 519 vs. 624 for spacing) while maintaining similar levels of accuracy in boredom threshold estimation.

Regarding accuracy, Chernoff exploration achieved slightly lower t-error in some settings, reflecting more precise threshold estimation. However, q-error values were largely similar across both exploration methods, and these small differences did not translate into practical improvements in session length or acceptance rate.

In terms of session length, the sample-based exploration achieved slightly higher averages compared to the Chernoff-based approach in the session length (Tables 2,3). Figure 3 illustrates that both methods eventually converged to similar long-term values, with shorter exploration phases under the sample-based strategy.

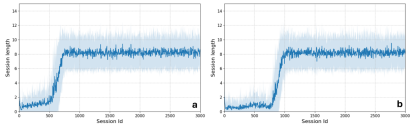

Figure 3: Session length over time in sample-based (a) and Chernoff-based (b) exploration algorithms using spacing constraints.

Overall, the efficiency of the sample-based approach—requiring fewer exploration steps without increasing estimation errors—makes it the more attractive choice. The observed differences in acceptance rate between the two exploration strategies were statistically negligible.

5.4 Comparison of Exploitation Algorithms

We next evaluate the effect of the exploitation algorithm, comparing *UCB* and *Thompson sampling*. Both methods achieved close performance across all metrics. For example, under spacing with Chernoff exploration, UCB reached a mean session length of 7.8 with an acceptance rate 0.886, while Thompson sampling yielded 8.22 and 0.891 (Table 3).

No notable differences were observed in t-error between the two algorithms, while q-error was slightly higher for Thompson sampling, indicating marginally lower estimation accuracy relative to UCB.

These findings suggest that, once boredom thresholds are established, the specific choice of exploitation algorithm has minimal influence on overall performance, with both UCB and Thompson sampling serving as effective and stable exploitation strategies in this context.

5.5 Comparison of Boredom Constraints: Spacing vs. Capping

We next examine the two boredom constraints: *spacing* and *capping*. In both cases, 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., 1226 vs. 629 steps with Chernoff–UCB as shown in Tables 2,3).

Performance differences also reflect how the threshold t is applied. For example, with $\bar{t} = 7$ in a 14-session window, spacing

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.

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 effected 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 excellent session lengths could be achieved using the Samples method (with Thompson used once t is known), the t errors have had no impact in 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

Table 2: Performance comparison of exploration and exploitation variants when learning the Capping boredom constraint. Session length and acceptance rate are measured across the last 500 sessions. Standard deviations are shown in parentheses.

Metric/Method	Chernoff Thompson	Chernoff UCB	Samples Thompson	Samples UCB
Session Length	10.52 (0.57)	10.16 (0.61)	**10.56** (0.58)	10.24 (0.62)
Acceptance Rate	**0.913** (0.002)	0.91 (0.002)	**0.913** (0.001)	0.911 (0.002)
t-Error	0.26	**0.25**	0.72	0.77
q-Error	0.031	**0.017**	0.035	0.019
Mean Exploration	1250	1226	888	**881**
Min Exploration	383	379	**305**	308
Max Exploration	1767	1751	1378	**1369**

Table 3: Performance comparison of exploration and exploitation variants when learning the Spacing boredom constraint. Session length and acceptance rate are measured across the last 500 sessions. Standard deviations are shown in parentheses.

Metric/Method	Chernoff Thompson	Chernoff UCB	Samples Thompson	Samples UCB
Session Length	8.22 (0.33)	7.8 (0.41)	**8.23** (0.34)	7.86 (0.4)
Acceptance Rate	0.891 (0.001)	0.886 (0.001)	**0.892** (0.001)	0.887 (0.002)
t-Error	0.27	**0.26**	0.53	0.57
q-Error	0.052	**0.015**	0.052	0.016
Mean Exploration	624	629	519	**514**
Min Exploration	390	**368**	422	418
Max Exploration	968	970	631	**623**

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 (around 8) compared to capping (around 10).

Overall, these findings suggest that both metrics are learnable, but capping places a heavier burden on the exploration phase, resulting in longer learning times.

5.6 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.

The results shown in Table 4 indicate that increasing the item catalog substantially prolongs the exploration phase, particularly for explore-first algorithms. Nevertheless, the algorithms ultimately converged to higher values of both session length and acceptance rate, with the performance gap between Samples–Thompson and the baselines widening as the caltalog grew.

5.7 Item-Level Behavior Analysis

To gain insight into how item preference (q) and boredom threshold (t) affect learning, we analyze results at the item level. Figure 4 plots the observed acceptance probability for a selection of items as a function of the boredom level; Figure 5 presents box plots of exploration lengths per item.

Table 4: Performance of boredom-aware exploration methods compared to standard Thompson sampling baseline, with 149 items catalog. Standard deviations are shown in parentheses.

Constraint	Method	Converged Session Length	Converged Acceptance Rate
Capping	Samples–Thompson	**28.78** (1.18)	**0.966** (0.000)
	Thompson (baseline)	17.92 (1.46)	0.947 (0.001)
Spacing	Samples–Thompson	**17.71** (0.76)	**0.947** (0.001)
	Thompson (baseline)	10.74 (0.51)	0.915 (0.001)

Stability of learned q and its effect on threshold error. Figure 4 shows that items with high true likability produce very stable estimated acceptance curves. By contrast, low-likability items show noisy, low-valued acceptance curves with wide confidence bands. Consistent with this intuition, items with lower q values tend to incur larger errors in the learned threshold $\hat{t}$. Deterministic items (those that are effectively always accepted or always rejected) are omitted from these comparisons because they are easier to learn.

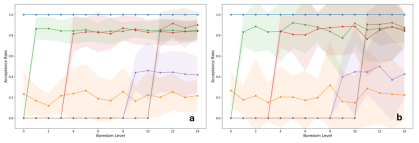

Figure 4: Per-item acceptance probability as a function of boredom level (example items), for Capping (a) and Spacing (b) constraints. Shaded areas denote confidence intervals. High-likability items (top curves) show stable estimates; low-likability items are noisier.

Exploration length and the role of the threshold (t). Figure 5 shows substantial variation in exploration effort across items, ordered by their t value. When using the capping formulation (expressed in the implementation as the complementary metric), items with larger t values typically require less exploration to reach a stable estimate; that is, larger t under capping is associated with shorter median exploration length. Under spacing, the reverse trend is observed: items with smaller t values tend to be resolved more quickly (shorter exploration), again excluding deterministic items. These opposing trends are a direct consequence of the use of the complementary form of capping: an identical numeric threshold maps to different effective restrictions under the two constraints, so the ranges that are closer to the default state are easier to explore.

Synthesis. Taken together, the item-level analysis indicates two separate but related influences on learning. Likability (q) primarily governs the stability of value estimates and is strongly correlated with t-error: lower q produces less informative feedback and higher external source of information, such as by collaborative filtering techniques.

To evaluate the potential of these learning accelaration 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 complimentary. 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.

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

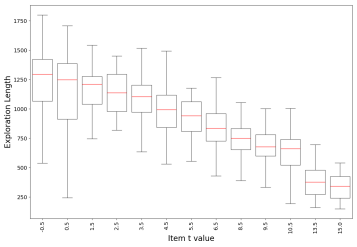

Figure 5: Distribution of exploration lengths for selected items by t value using Capping constraint. Items are ordered by t along the x-axis.

threshold estimation error. The threshold value (t) primarily governs exploration effort, with the effective direction of that influence depending on the boredom metric (capping vs. spacing) because of the complementary representation. These observations explain why some items require substantially longer exploration and why learning accuracy varies across the item population.

5.8 Evaluation with Real Spacing Constraints

To further validate the approach under realistic conditions, we incorporated the actual spacing constraints currently deployed in the system. These constraints, which are hard-coded in the current suggestion mechanism, were translated into the simulator and combined with real user profiles and the full catalog of 149 plans. For adaptation to the used setting, we assumed a two-day boredom window consisting of 24 sessions, with one session every two hours.

We evaluated the Samples–Thompson variant of the method against a standard Thompson Sampling baseline. The boredom-aware algorithm exhibited a markedly stronger converged session length (12.97 vs. 9.62), but suffers from a high exploration phase, with a mean length of 4688.69 sessions.

These findings correspond closely with the results observed in the simulator experiments, demonstrating that the long-term advantages of boredom-aware exploration also hold in realistic application scenarios.

6 DISCUSSION AND CONCLUSION

The presented analysis raises several issues worthy of discussion, and open questions regarding the interaction between parameters and algorithmic behavior. We discuss these below.

User profile impact. All experiments were conducted using a single base user profile. However, the approach remains effective across a spectrum of user behaviors, reinforcing its applicability in real-world conversational settings where user cooperation levels can fluctuate over time. To assess the robustness and generalizability of our findings, we evaluated the algorithms under varying levels of user cooperation, simulating differences in responsiveness and engagement. Results show that the proposed boredom-aware algorithm estimates boredom thresholds more accurately as user cooperation increases, but even under low cooperation, the method continues to promote higher long-term engagement compared to the baselines.

Exploration/Exploitation, Again. One important direction concerns the trade-off between exploration and convergence: to what extent can the exploration phase be shortened without losing the advantage in the accuracy of the learned thresholds? Identifying principled stopping criteria or adaptive schedules may reduce unnecessary exploration while preserving convergence quality.

Joint constraint and preference learning. Another possible direction is the joint learning of constraints and action-value estimates. Rather than treating constraint estimation and q estimation as sequential processes, they could be learned in parallel using selective updates—where certain conditions trigger updates to the Q-function and others to the constraint model.

Summary and Future Work. We presented a two-stage method for learning user boredom constraints while enforcing them within the recommendation process. The results clearly show the benefit of the learning approach. For example, for the *capping* boredom constraint and a single user instance, we compared the proposed algorithm (Samples-UCB-capping) against the Thompson baseline. Samples-UCB achieved an acceptance rate of **91.1%**, compared to **90.1%** for Thompson. Although the absolute improvement is approximately one percentage point, the difference is statistically significant (two-tailed t-test $p < 0.00001$) given the large sample size. This result demonstrates that, under this configuration, the proposed algorithm yields a consistent advantage over the baseline.

An important direction for future work concerns the role of states in learning boredom constraints. The current formulation treats the problem as a stateless multi-armed bandit with constraints. Incorporating stateful dynamics while leveraging the benefits of boredom constraints, like in a Constrained Markov Decision Process [19], may reveal different trends. For example, user boredom or engagement could evolve over time in ways that interact with item selection, making reinforcement learning with state representations more suitable.

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

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

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

[5] 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.

[6] 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.

[7] 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.

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

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.

[9] 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.

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

[11] 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

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

[13] 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

[14] 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.

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

[16] 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

[17] 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).

[18] 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.

[19] 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.

[20] 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).

[21] 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).