Bubble Bursting Implementations and Dynamics in Recommender Systems

Distributed REsearch Apprenticeships for Master's Program
Project Report

Madison G. Thantu
Columbia University
mgt2143
`madison.g.thantu@columbia.edu`

Jannik J. Wiedenhaupt
Columbia University
jjw2196
`j.wiedenhaupt@columbia.edu`

Abstract

Recommender systems (RSs), which offer personalized suggestions to users, power many of today's social media, e-commerce, and entertainment services. Despite the utility and ubiquity of RSs, there is growing concern about the harmful effects that propagate from these systems, such as filter bubbles, user homogenization, and unfairness, and the fundamental trade-off that exists between accuracy and diversity. Although these concerns have become mainstream, our understanding of how these detrimental effects emerge in the complex RS ecosystem remains incomplete, and the present study aims to contribute to this important line of research. We first operationalize user behavior homogenization and filter bubbles, defining a set of attributes for which our subsequent metrics are tailored to. We then implement several recommender models to represent the accuracy-diversity prioritization spectrum, including an accuracy-driven myopic algorithm, a diversity-promoting algorithm inspired by the xQuAD framework, and a naively exploratory random algorithm. We conclude our report with a comparative analysis of how the three aforementioned phenomena emerged, evolved, and interacted with one another in our simulation ecosystem as the prioritization of accuracy versus diversity is varied. Our implementation and experiments can be found at `https://github.com/madisonthantu/T-RECS-RS-research`.

1 Introduction

The information economy has altered how information is valued, manufactured, and distributed. While the accessibility of information is fundamentally positive, the paradox of choice [17] explains how an overabundance of information can complicate decision-making. As such, by reducing a data set to items that are user-relevant, recommender systems (RSs) function as a valuable tool for navigating this problem of information overload. They are increasingly influencing how the layperson participates in this new economy—helping to guide users' decisions and attention.

However, aside from their utility, RSs have also gained increased attention due to their downstream impacts on information consumption. RSs have been shown to facilitate filter bubbles, wherein users are recommended and subsequently consume content that aligns with their preexisting views [6]. Such filter bubbles have been linked to increases in misinformation and polarization [7].

Standard recommender system (RS) policies are myopic: The recommendation algorithm greedily optimizes for its pre-defined objective (e.g., maximizing user engagement or dwell time). This implementation creates an implicit filter such that the presented content or information is: (a) in alignment with the user's beliefs, opinions, etc., and (b) homogeneous.

Consequently, exploring RS implementations that mitigate rather than exacerbate filter bubbles is crucial. As such, the present research seeks to characterize the effects of bubble-bursting strategies in a dynamic RS ecosystem. Utilizing an RS simulation, we estimate long-term user behavior under various recommendation algorithms. These include myopic and several exploration-facilitating implementations. We then conduct a comparative analysis of these models and their effects on accuracy, recommendation quality, homogeneity, and fairness.

2 Related work

Fairness. Existing research generally defines fairness with respect to either users [2, 23] or items [19, 25], where item fairness can also be formulated at the group level w.r.t. providers. There is research, although limited, that attempts to address fairness with respect to multiple entities [24, 15]. Mladenov et. al. [15] aim to achieve user and provider fairness using a model that optimizes for the "socially optimal equilibrium of the ecosystem" w.r.t. content provider viability.

Exploration. While often associated with the field of Reinforcement Learning (RL), we use the term exploration to encapsulate general RS behaviors that produce recommendations that do not exclusively consist of items with the highest predicted user-item score. In addition to the default metric of accuracy, diversity and novelty are increasingly attractive properties [22, 12]. In an RL-based RS, Chen et al. study the effect of several exploration-promoting methods in simulations and live experiments [5]. There are also many non-RL-based methods that similarly explore the dynamics and potential trade-offs that may exist between accuracy and exploration-promoting strategies [1, 26].

Filter Bubbles. Precipitating from the rise of social media, research on the relationship between RSs and filter bubbles has proliferated. As stated previously, the RS ecosystem is especially complex and dynamic. Much of the technical research on filter bubbles is focused on capturing the real-world dynamics, such as information propagation and user choice models. However, the internal dynamics of RSs are underexplored. [16] is one such study that focuses on the interaction between filter bubbles and the internal qualities of RSs.

Homogenization. A concerning issue that arises with RSs is that of homogenization, a phenomenon where, over time, content that is recommended to different users becomes more similar, and consequently as does user behavior. Chaney et al. [4] examine the effects of algorithmic confounding in RSs, which occurs when an RS is trained or evaluated on data from users that have already been exposed to recommendations from the same algorithm. Their results indicate that algorithmic confounding magnifies user behavior homogeneity without corresponding increases in utility. While homogenization is generally a negative effect, if the objective is to mitigate filter bubbles, then the distinction between homogenization within versus between filter bubbles may be important.

3 Simulation Environment

To observe the long-term effects of changing user preferences due to recommender systems, we use the agent-based T-RECS simulation environment [13]. The general architecture of the T-RECs environment can be seen in Figure 1. We specifically use the T-RECS architecture due to its ability to model dynamic user preferences [8, 11]. In T-RECS, after a user chooses an item to interact with from the recommendation slate, the user's attributes "drift" towards the chosen item's attributes. This is implemented using spherical linear interpolation [13]. This "drift" is highlighted in Figure 1 as arrow *(b)*.

The full T-RECS framework and implementation can be found in [13]. To summarize, the two main inputs of the T-RECS architecture are the true user preferences $U \in \mathbb{R}^{N_{\text{users}} \times N_{\text{attributes}}}$ and the true item attributes $I \in \mathbb{R}^{N_{\text{items}} \times N_{\text{attributes}}}$. The true attribute representations are used for measurement and interaction simulation and are not known to the model. Instead, the recommendation algorithm randomly initializes the predicted user preferences $\hat{U}$ and predicted item attributes $\hat{I}$. Using these predicted matrices, the environment recommends a set of items A_t^r at timestep t to each user u. Since I and U are initially unknown to the model, the environment is initialized in "start-up" mode, where new items are randomly recommended to users in order to maximize exploration. After start-up mode, the model enters "run" mode. In "run" mode, the default algorithm is myopic and aims to maximize the predicted score $r(u, i)$ of user u and item i at each timestep t.

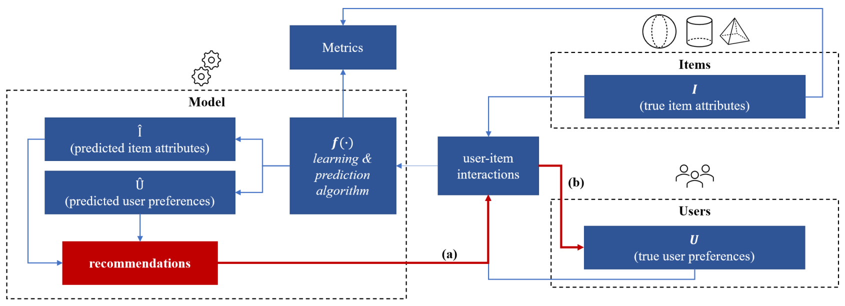


Figure 1: A conceptual diagram of the design components and system interactions of a T-RECS simulation. Lines with arrows indicate inputs fed into outputs. Dashed grey lines indicate conceptual components. The red arrows (a) and (b) are the main levers in influencing long-term user preferences.

The T-RECS framework provides default implementations for the user-choice model, the optimization model, and the user dynamics. For a detailed description, we refer readers to the official documentation[1] and Appendix A.

4 Methodology

4.1 Attributes

We first introduce three different but closely related attributes: *topic clusters*, *user communities*, and *user-topic mappings*, which we then use to define our metrics and the xQuAD recommender model. First, we compute item and user embedding vectors via non-negative matrix factorization, which is further discussed in section 5.2. Then, we perform k-means clustering ($k = 10$) on the learned item embeddings and user embeddings to compute *topic clusters* and *user communities*, respectively, such that each item in I and user in U is assigned to the nearest of their respective clusters. In our analysis, we use topic clusters to promote and measure diversity at a more abstract level, whereas user communities are used to analyze trends in user homogenization. Lastly, we introduce *user-topic mappings*, where, the user-topic mapping for each user $u \in U$, is computed as the topic cluster which minimizes the Euclidean distance between u's embedding and the corresponding topic cluster's centroid. Note that it is possible that no users are mapped to a particular topic cluster. The user-topic mappings function as a proxy for filter bubbles, where each topic cluster is considered to be a potential filter bubble. We assume that each user is associated with a filter bubble, the degree of which can vary. Using these attributes, we analyze if and how clusters, communities, and mappings are affected by various recommendation algorithms.

4.2 Recommendation Models

One objective of this study is to explore the accuracy-diversity trade-off within a dynamic RS ecosystem at the granular level. We do so via a comparative analysis of different RS models, representing a wide spectrum of algorithmic priorities and approaches with respect to this trade-off.

Myopic Recommender. The Myopic model is the standard, greedy recommendation algorithm. Recommendation slates are generated by selecting the top k items from the predicted user-item scores and are computed as:

$$S_{\text{myopic}}(\hat{u}, \hat{i}) = \hat{u} \cdot \hat{i}. \tag{1}$$

This model serves as a benchmark in our exploration of the accuracy-diversity trade-off and the effects that these two competing principles have on the RS ecosystem.

xQuAD Recommender. This model is a loose implementation of the eXplicit Query Aspect Diversification, or xQuAD, Framework presented in [18]. This re-ranking algorithm takes as its input a ranked recommendation list R generated by $S_{\text{myopic}}(\hat{u}, \hat{i})$. It then iteratively builds a new

[1]`https://elucherini.github.io/t-recs/index.html`

recommendation slate S by, first, computing new scores for each item, $S_{\text{xQuAD}}(\hat{u}, \hat{i})$, and then adding the item with the highest score, $\max_{i \in I} S_{\text{xQuAD}}(\hat{u}, \hat{i})$, to the recommendation slate. Given an item $i \in I$, a user $u \in U$, an item topic $d \in D$, and the recommendation slate at the current iteration, this new score is computed using the following criterion:

$$P(i|u) + \lambda(1 - P(i|d_i, S)).$$

The first term $P(i|u)$ represents the likelihood of user $u \in U$ being interested in item $i \in I$, independent of the current iteration's recommendation slate. This term promotes ranking accuracy and is computed via $S_{\text{myopic}}(\hat{u}, \hat{i})$. The second term $1 - P(i|d_i, S)$ incorporates the probability of item i given the item's topic $d \in D$ and the current recommendation slate S, such that if there is an item $i_k \in S$, then given an item i_l with $d_{i_l} = d_{i_k}$, and an item i_n with $d_{i_n} \neq d_{i_k}$, and $i_k \neq i_l \neq i_n$, then $P(i_l|d_{i_l}, S) < P(i_n|d_{i_n}, S)$. Consequently, this term promotes diversity in the re-ranking algorithm, such that an item that belongs to a topic that is already represented by the current recommendation slate will be penalized, the degree of which is controlled by the parameter λ. Consequently, we have:

$$S_{\text{xQuAD}}(\hat{u}, \hat{i}) = (1 - \lambda)(\hat{u} \cdot \hat{i}) + \lambda(1 - D(i, d_i, S)) \tag{2}$$

Two different approaches were used to compute $P(i|d_i, S)$.

i The *Binary xQuAD* implementation uses an indicator function where $P(i|d_i, S)$ is equal to 1 when $d_i \in S$ and 0 otherwise, such that:

$$D(i, d_i, S) = \begin{cases} 1, & \text{if } d_i = d_j \text{ for } j \in S \\ 0, & \text{otherwise} \end{cases}$$

ii The *Smooth xQuAD* implementation computes $P(i|d_i, S)$ as the ratio of items in the current recommendation slate $j \in S$ where $d_j = d_i$, such that:

$$D(i, d_i, S) = \frac{|\{j \mid j \in S \wedge d_j = d_i\}|}{|S|}$$

Probabilistic Model. This model generates randomized recommendations, where an item's recommendation probability is proportionate to its predicted score. Consequently, greater probability mass is allocated to items for which the model predicts higher user preference.

Random Interleaving Model. The Random Interleaving model randomly samples k items out of the item set I, which are then "interleaved" into the recommendation slate at each iteration of the simulation. In the implementation presented here, $k = 4$, such that a given recommendation slate consists of 4 items sampled at random and the 6 items associated with the highest predicted user score. The order in which these items are interleaved is similarly selected at random.

Random Model. All recommendations made by the Random model, as the name suggests, are randomly generated. It is a maximal implementation of the Random Interleaving model, where $k = 10$, i.e., the size of the recommendation slate. This model functions as a baseline for the converse extreme in the accuracy-diversity trade-off, as compared to the Baseline Myopic model.

Repeatable Model. In this model, repeated items are allowed in the system, such that the recommender is able to recommend items to a user which they have already interacted with in a previous timestep. Moreover, users are able to interact with an item independent of whether they have already interacted with it before. Out of the six models, this repeated items, repeat interactions model minimizes exploration throughout the simulation. This is because once a user provides high positive feedback for an item, the recommender is able to include this item in subsequent slates and the user is able to re-interact with it, rather than the model exploring different items which may in fact be associated with greater degrees of positive user feedback. It is for this reason that the Repeatedable model was included in the present study.

4.3 Measurement

Accuracy is the standard RS benchmark, and it defines how closely a recommended item aligns with a user's preferences. We use two definitions of accuracy. First, we measure the mean squared error between the predicted user preferences $\hat{U}$ and the actual user preferences U as

$$MSE = \frac{1}{N_{\text{users}}} \sum_{l=1}^{N_{\text{users}}} (u_l - \hat{u}_l)^2. \tag{3}$$

For this, it is important to remember that the actual user preferences change throughout the duration of the simulation due to *drift*. Second, we measure top-k recall on the recommended set A^r, which is defined as the ratio of items interacted with in the first k items of each recommended set such that

$$Recall = \frac{1}{N_{\text{users}}} \sum_{l=1}^{N_{\text{users}}} \sum_{j=1}^{k} 1_{A_j^r \in E_t(u)}, \tag{4}$$

where $E_t(u)$ are the interactions of u at timestep t. A lower MSE means our model knows the user well, while a high recall means that our model ranks items such that the user's choice corresponds to highly ranked items.

Although accuracy remains the driving basis behind state-of-the-art RSs, the downstream effects of accuracy-driven optimization have received growing concern and criticism [7, 14]. For this reason, there is a growing body of work examining metrics other than accuracy [15, 3, 9, 4]. We analyze several such metrics, which can be split into two buckets.

The first set of metrics relates to exploration, including properties such as diversity and novelty. These are properties that can be directly influenced by changing the recommendations (compare link **(a)** in Figure 1). Diversity measures the number of distinct topics the recommended set contains. We use the average dissimilarity of all pairs of items in the set, as defined by Chen et al. [5].

$$\text{Diversity}(A^r) = 1 - \frac{1}{|A^r|(|A^r| - 1)} \sum_{i,j \in A^r, i \neq j} sim(i,j) \tag{5}$$

where, given two items $i, j \in I$, where $i \neq j$, then $sim(i,j) = 1$ if i and j belong to the same topic cluster, and 0 otherwise. Our definition of novelty is inspired by [20], and is defined using the logarithm of user inverse frequency, or the inverse of popularity. Given an item i, the set of users U, and the subset of users that have interacted with item i, U_i, novelty is defined as:

$$\text{Novelty}(i) = -log_2 \frac{|U_i|}{|U|}. \tag{6}$$

The second set of metrics is related to the simulation dynamics themselves, including user homogenization and the distribution of items represented throughout the simulation. These metrics focus on how user-item interactions and users' preferences evolve as a result of the recommendations presented by the model (compare link **(b)** in Figure 1). This metric set includes the interaction similarity and cosine similarity of and the Euclidean distance between different subsets of the user population. Interaction similarity is computed using the Jaccard index of each user's set of interacted items. To explore homogenization in the context of filter bubbles, we measure how these metrics evolve w.r.t. the global, intra-cluster, and inter-cluster user populations. Inter- v. intra-cluster groups are defined using our operationalization of user communities, as defined above, such that intra-cluster measurements consider only pairs of users that belong to the same user cluster whereas inter-cluster measurements consider pairs that belong to different user clusters. Given a user community $C_V \in \mathcal{C}$, where V is the subset of users belonging to C_V, the intra-cluster measurements at timestep t are computed as:

$$\text{I}_{\text{Intra}}(V,t) = \frac{1}{|V|} \sum_{v_i, v_j \in V} J_{v_i, v_j}(t), \tag{7}$$

$$\text{D}_{\text{Intra}}(V,t) = \frac{1}{|V|} \sum_{v_i, v_j \in V} ||v_i - v_j||, \tag{8}$$

$$\text{C}_{\text{Intra}}(V,t) = \frac{1}{|V|} \sum_{v_i, v_j \in V} \frac{v_i \cdot v_j}{||v_i||||v_j||}, \tag{9}$$

for interaction similarity, Euclidean distance, and cosine similarity, respectively. Analogous procedures are followed for computing these metrics w.r.t. inter-cluster and global user populations.

5 Experiments

We evaluate our proposed recommendation algorithms (Sec. 4.2) through extensive experiments to measure the proposed metrics (Sec. 4.3) using the T-RECS environment. We examine the temporal development of the defined metrics and analyze if and how final user community assignments and final user-topic mappings differ between recommendation algorithms. The remainder of this report is laid out as follows: we outline the simulation ecosystem and describe our dataset and trained embeddings, and we conclude with a discussion of our findings.

5.1 Ecosystem

The T-RECS platform calculates all defined metrics for the system at each timestep t. At each timestep, the RS recommends $k = 10$ items to each user. Each user interacts with one item per timestep. They pick the item that has the highest score according to their user value function defined in equation 10, where the attention hyperparameter $\alpha = -0.8$. The actual user preferences are assumed to drift towards each interacted item by a factor of 0.10. The available items and the corresponding true item attributes are static. We run all experiments with the same environment and simulation parameters unless stated otherwise, and all simulations consist of 100 timesteps. In order to evaluate algorithmic confounding and user behavior homogenization, we follow the approach taken by [4], considering two user-item interaction scenarios. Under the *single training* scenario, for the first 50 iterations, the model runs in "start-up" mode, as described in section 3, which is then followed by a distinct run phase of 50 iterations, during which no additional training occurs. Conversely, under the *repeated training* scenario, the model undergoes 10 iterations in the "start-up" phase. For the remaining 90 iterations, the model operates in the "run" phase, but is retrained after each step using all data observed up to that current timestep. These two training modes are an additional dynamic that we consider in our following analysis.

5.2 Data

Like Mladenov et al. [15], we generate actual user preferences U and item attributes I using non-negative matrix factorization. We use the distribution containing $\sim 100,000$ ratings of $1,623$ movies by 943 users. Given the sparse ratings matrix $R \in \mathbb{R}_{\geq 0}^{N_{\text{users}} \times N_{\text{movies}}}$. Simplifying our model, we turn the ratings into a binarized interaction matrix $E \in \{0,1\}^{N_{\text{users}} \times N_{\text{movies}}}$ with $E_{i,j} = 1(R_{i,j})$. The embeddings for the true user preferences and true item attributes are formed by solving the following optimization problem:

$$\min_{U,I} ||E - UI||_{Fro}^2$$

which yields a matrix of user preferences $U \in \mathbb{R}_{\geq 0}^{N_{\text{users}} \times N_{\text{attributes}}}$ and item attributes $I \in \mathbb{R}_{\geq 0}^{N_{\text{items}} \times N_{\text{attributes}}}$. The randomly initialized factors U, I are alternatively updated via weighted alternating least squares [10] until convergence. We use embedding rank $N_{\text{attributes}} = 20$.

5.3 Results

Results Overview: As shown in 2, in general, the mean squared error (MSE) converges to ~ 0.8 and ~ 1.3 for repeated and single training modes, respectively. Noticeably, the MSEs for the Random and Random Interleaving models under the repeated training scenario, and the MSE for the Repeatable model under both training scenarios, do not converge. The former likely stems from the fact that these models were run in "train" mode for a mere 10 timesteps before undergoing the processes of randomization/random interleaving. The latter observation can be attributed to the minimized exploration taken by the Repeatable model. This is supported by table 1, which shows a statistical summary of interactions per item at the end of each simulation. For the Repeatable recommender, the maximum number of interactions received by any one item was $4,345$ and $4,770$ for single and repeated training modes, respectively, which are dramatically higher compared to other models. This is a likely indicator of bias in the recommendation slates generated by the Repeatable model. As indicated by 2, the increased number of timesteps spent in "train" mode reduced the MSE rate of increase for the Repeatable model, likely because there were more iterations devoted to exploration and to learning user preferences. Once in the "run" mode, the model focuses narrowly on the items for which it has already received high user feedback; the positive feedback received by such items may have been high relative to the user-item interactions that were observed, but that feedback may not be high relative to the entire set of items, including the items for which the model has yet to observe user-item interactions.

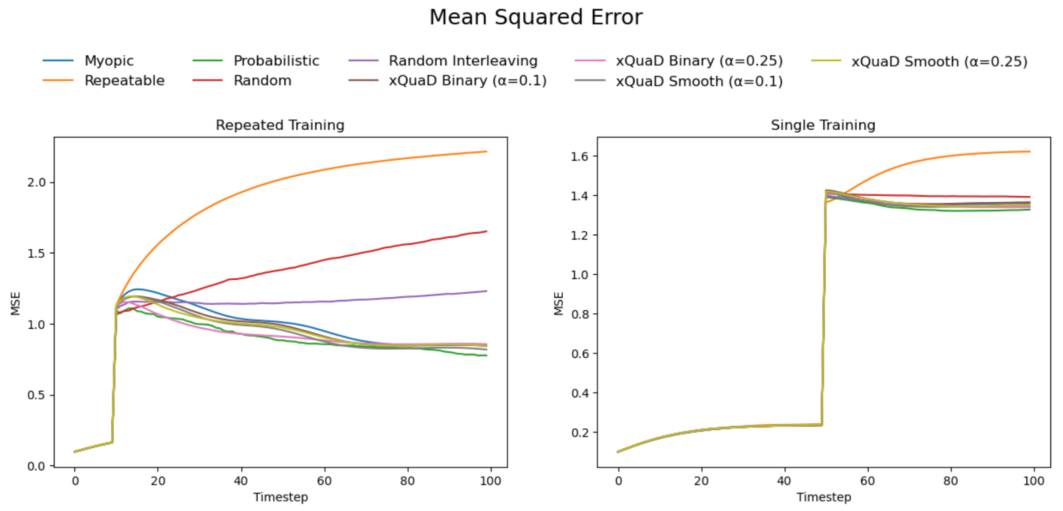


Figure 2: Mean Squared Error (MSE) by model, under repeated training mode (left) and single training mode (right).

	Single	training			Repeated	training		
	Max	Min	Median	SD	Max	Min	Median	SD
Myopic	463	0	35	61.90	527	0	37	60.78
Repeatable	4345	0	9	198.04	4770	0	4	242.66
Probabilistic	543	0	29	71.09	424	0	36	61.06
Random	373	0	18	75.41	380	0	18	75.97
Random Interleaving	303	0	30	63.57	432	0	34	61.03
Binary XquAD, $\alpha = 0.1$	571	0	33	67.76	450	0	37	59.26
Binary XquAD, $\alpha = 0.25$	603	0	25	75.28	647	0	37	60.03
Smooth XquAD, $\alpha = 0.1$	625	0	33	67.06	475	0	37	59.60
Smooth XquAD, $\alpha = 0.25$	668	0	30	71.81	521	0	39	58.77

Table 1: Statisticcal summary of item interactions at the end of each simulation, by recommender model and training mode.

Figure 3 shows the mean novelty by model under the two training modes. Due to its implementation, this metric was recorded during the "run" phase only. Because novelty was defined via user inverse frequency, a lower mean novelty correlates with a greater number of items being interacted with, as opposed to many users interacting with only a few items. Given that new items were not being created throughout the simulation, this explains the downward trend observed in mean novelty across all recommendation models. While the small rate of decline observed in the Repeatable model follows given that this model focuses narrowly on items that received positive feedback during the initial iterations of the simulation, the trends observed in mean novelty for the Random and Myopic recommenders are somewhat surprising and warrant additional analysis focusing on the distribution of items that are represented in user-item interactions and the recommendation slates generated by the particular model.

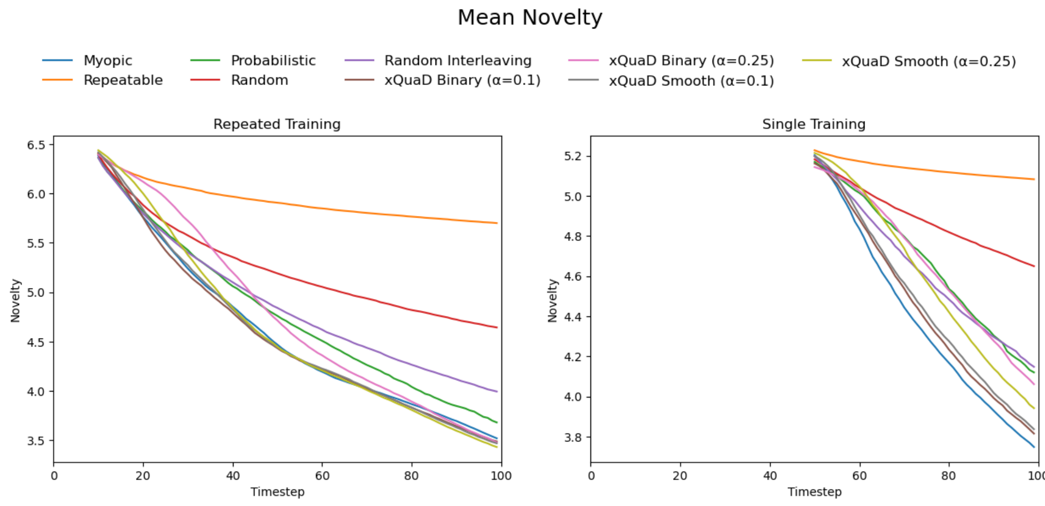


Figure 3: Mean item novelty at each iteration, under repeated training mode (left) and single training mode (right), where mean item novelty is computed as the logarithm of user inverse frequency (i.e., the log inverse of item popularity). Note that, under the present implementation, this metric was only observed during the "run" phase of the simulation.

Homogenization: One metric used to analyze user behavior homogenization was cosine similarity, which we measured with respect to intra-cluster and inter-cluster populations, as well as the average global cosine similarity. We initially hypothesized that, under a myopic recommendation policy, this metric would be greater w.r.t. intra-cluster populations, as this could indicate greater in-group homogenization alongside global trends towards user behavior homogeneity. Figure 4 presents the average cosine similarity ratio between intra-cluster and inter-cluster groups, where a value greater than 1 indicates greater intra-cluster cosine similarity. As indicated by figure 4, the Repeatable recommender is the only model that exhibits substantially greater intra-cluster cosine similarity. Notably, this ratio also converges to 1 for the Myopic recommender, under both single and repeated training modes, contrary to our initial hypothesis.

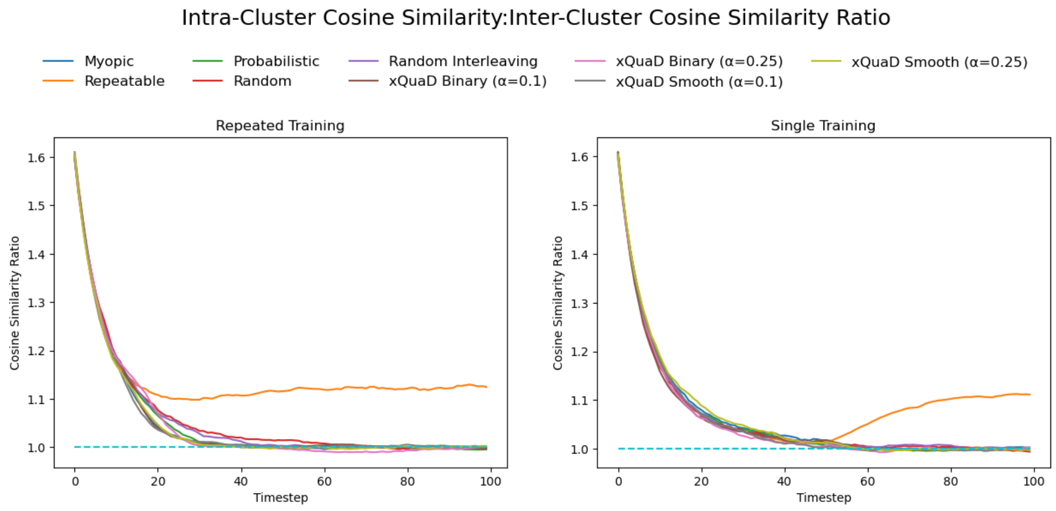
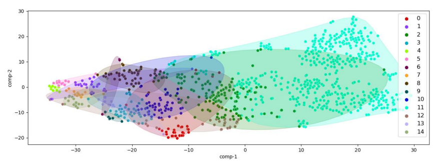
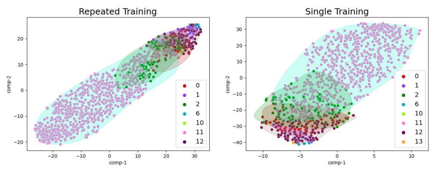


Figure 4: Ratio of mean intra-cluster cosine similarity relative to mean inter-cluster cosine similarity, under repeated training mode (left) and single training mode (right). Values greater than 1 indicate a greater degree of intra-cluster homogeneity relative to inter-cluster homogeneity.

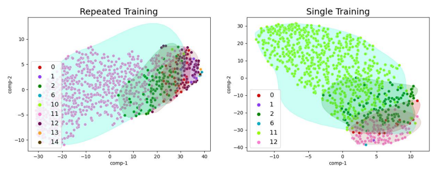
We visualize user communities using t-distributed stochastic neighbor embedding (t-SNE) [21], with $perplexity = 50$. Figure 5 shows these results: In figure 5a, we see the result of performing t-SNE on the initial user representations and the corresponding initial user community assignments. Notice that all 15 of the user communities are represented due to the fact that these community assignments were generated with the constraint $k = 15$ during the initial k-means clustering invocation. For each model, at the end of the simulation, the final user representations were extracted from the system. Users were then assigned to a final user community based on this extracted embedding. This final community was computed by mapping the final user vector to the closest user cluster centroid (*recall* that these centroids were computed prior to the start of the simulation), such that the Euclidean distance between these two vectors was minimized.



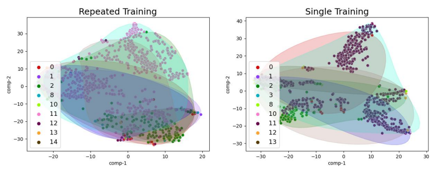
(a) Initial User Cluster Assignments



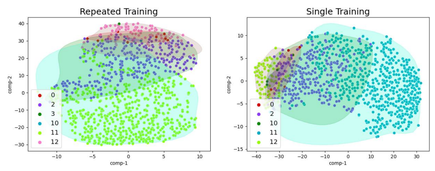
(b) Myopic Recommender



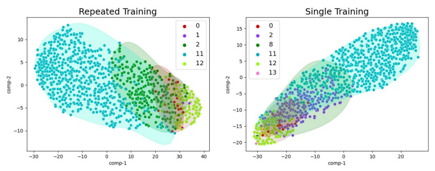
(c) Probabilistic Recommender



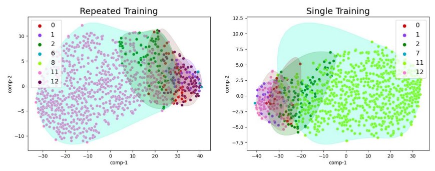
(d) Repeatable Recommender



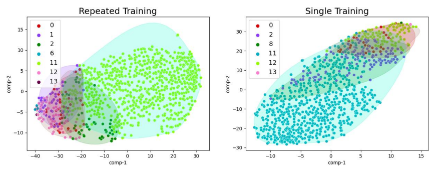
(e) Random Recommender



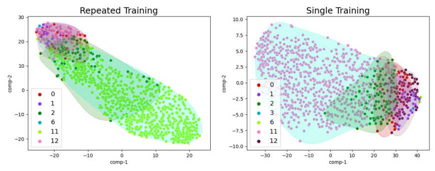
(f) Random Interleaving Recommender



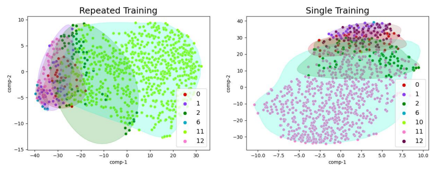
(g) Binary xQuAD, $\alpha = 0.1$



(h) Binary xQuAD, $\alpha = 0.25$



(i) Smooth xQuAD, $\alpha = 0.1$

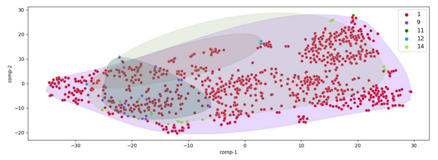


(j) Smooth xQuAD, $\alpha = 0.25$

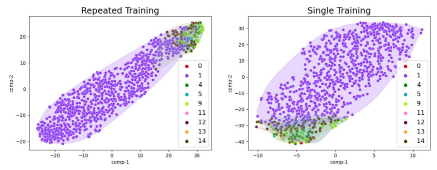
Figure 5: Visualizing user representations to user community assignments via t-distributed stochastic neighbor embedding (t-SNE), with $perplexity = 50$. (a) A t-SNE visualization that maps initial user representations to initial user communities, prior to the simulation. (b)-(j) t-SNE visualizations of final user representations mapped to final user community assignments, by model and training mode. Final user community assignments are computed by (1) extracting the final user representation from the system, (2) computing the Euclidean distance from the user representation to each of the 15 user community centroids, and (3) assigning the user to the user community that minimizes this Euclidean distance.

Figures 5b-5j show the results of performing t-SNE on these mappings from final user representations to final user communities for each of the models. Notice that in these final user community mappings, under no model nor either of the training modes are all 15 of the user communities represented.

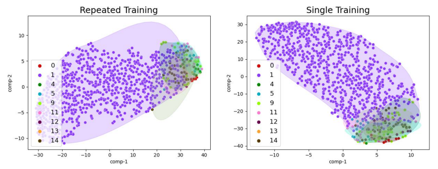
Although this could be interpreted as an indicator of user behavior homogenization, closer inspection suggests a much more nuanced narrative. Our initial hypothesis was that, in cases of increased homogenization, fewer user communities would be represented in this final mapping. The results obtained from the Repeatable model, shown in figure 5d, where a high number of final user communities are represented, are particularly confounding—we know that there was significant bias in this model's recommendation slates and user-item interactions, as indicated by table 1. Moreover, the Random recommender represents the fewest number of final user community assignments under both training modes (fig. 5e), contrary to our intuition.



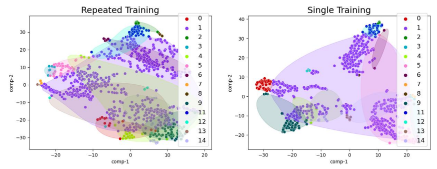
(a) Initial User Cluster Assignments



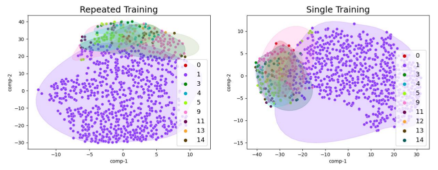
(b) Myopic Recommender



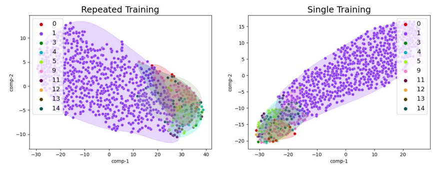
(c) Probabilistic Recommender



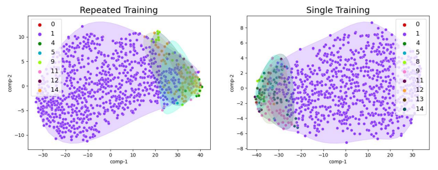
(d) Repeatable Recommender



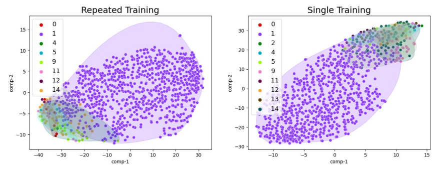
(e) Random Recommender



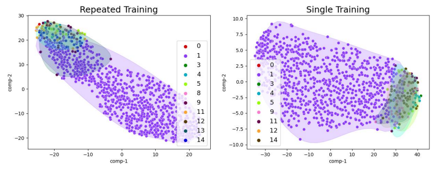
(f) Random Interleaving Recommender



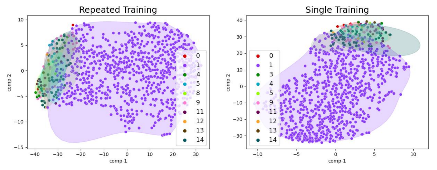
(g) Binary xQuAD, $\alpha = 0.1$



(h) Binary xQuAD, $\alpha = 0.25$



(i) Smooth xQuAD, $\alpha = 0.1$



(j) Smooth xQuAD, $\alpha = 0.25$

Figure 6: Visualizing user representations to user-topic mappings via t-distributed stochastic neighbor embedding (t-SNE), with $perplexity = 50$. (a) A t-SNE visualization that maps initial user representations to initial user-topic mappings, prior to the simulation. (b)-(j) t-SNE visualizations of final user representations mapped to final user-topic mappings, by model and training mode.

Filter Bubble Dynamics: Our analysis of filter bubble dynamics centers around the User-Topic Mappings attribute, as introduced in section 4.1. We computed final user-topic mappings using an analogous procedure to the one used to compute final user communities. Likewise, we then utilized t-SNE to visualize these final user-topic mappings, the results for which can be found in figures 6b-6j and are similarly surprising. For all models and under all training modes, a significant increase in the number of represented user-topic clusters is observed, with the Repeatable model representing *all but one* of these clusters. Moreover, for most of the models, the set of user-topic clusters represented under the single and repeated training modes are equal. Notably, this is not the case for the Random recommender nor any of the simulated variations of the xQuAD recommender (Binary and Smooth,

$\alpha = 0.1$ and $\alpha = 0.25$). Our initial hypothesis was that an increase in the number of user-topic mappings could be an indicator of bubble-building trends. However, the surprising results discussed thus far suggest a much more complex interplay between user behavior homogenization, filter bubble effects, and variations in algorithmic priorities.

User Cluster Fairness: The final dynamic that we considered is the effect of these different models on user fairness. We analyzed this by comparing the worst MSE associated with a user community versus the best MSE. The results, as shown in figure 7, indicate that, excluding the Repeatable and Random recommenders, repeated training was generally associated with lesser disparity between worst and best MSE by user community. Moreover, we see MSE values for the Myopic, Probabilistic, and xQuAD models that are in congruence with figure 2.

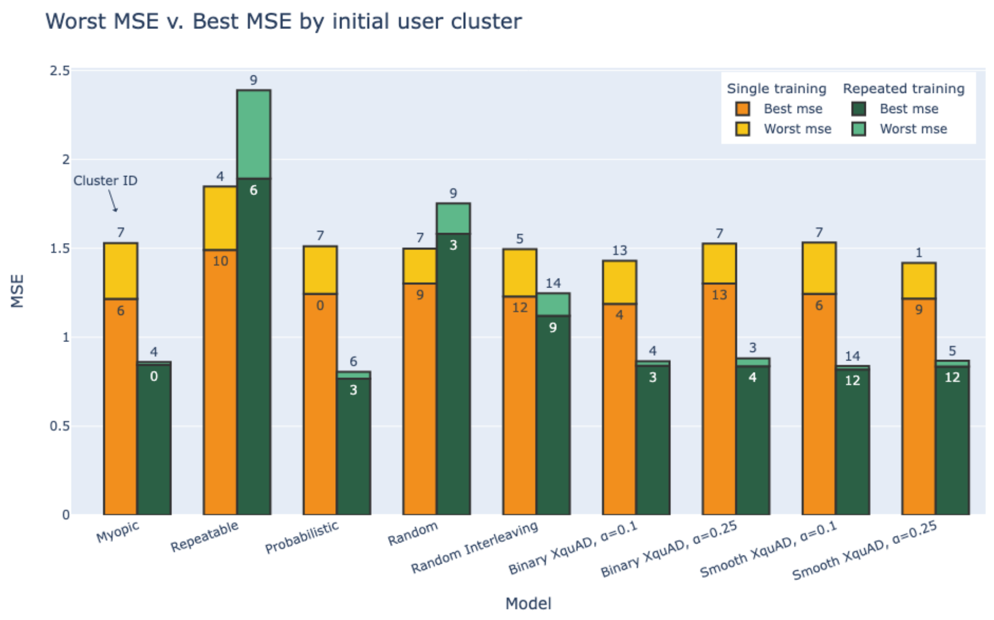


Figure 7: Comparison of the best MSE for a user cluster versus the worst MSE for a user cluster, by recommendation model and training mode.

6 Conclusion

To recap, the primary objective of this research is to further our understanding of how the dynamics of exploration, user behavior homogenization, and fairness interact with each other and the greater complexity of the RS ecosystem. While our findings suggest that quantifying these dynamics at the global scale fails to capture their evolution at the sub-population level, our results are inconclusive. Thus, we identify three key limitations of our work that also offer fruitful avenues for future research.

First, this research dealt exclusively with the Movielens 25k dataset, which is sparse and heavily skewed towards a few movies. Moreover, with only $1,682$ items, this Movielens dataset is relatively small and only allows for a limited number of item clusters and iterations of user-item interactions. As such, studying the interplay of homogenization, filter bubble effects, and user (un)fairness requires the use of sufficiently complex, built-out datasets. Moreover, we believe that examining these dynamics should be done in both synthetic and more "realistic" simulation environments in order to better understand their evolution in both the online and offline ecosystems.

Second, alternative metrics for defining and capturing user behavior homogenization and filter bubble effects are crucial. While the present study focused on cosine similarity and Euclidean distance, there is a plethora of metrics that can be used to quantify these dynamics, ones that are better equipped to deal with the high dimensionality that is typical of RS data.

Third, while our results do not support our initial hypotheses regarding intra- v. inter-cluster dynamics, they also do not refute them. While there is abundant research on the effects of both homogenization and filter bubbles in the RS ecosystem, there is limited work that explores the interplay of these properties. As such, additional work that focuses on the dynamic interactions of RSs has great value—both technically and socially.

References

[1] Sujoy Bag, Abhijeet Ghadge, and Manoj Kumar Tiwari. An integrated recommender system for improved accuracy and aggregate diversity. *Computers & Industrial Engineering*, 130:187–197, 2019.

[2] Jesús Bobadilla, Raúl Lara-Cabrera, Ángel González-Prieto, and Fernando Ortega. Deepfair: deep learning for improving fairness in recommender systems. *arXiv preprint arXiv:2006.05255*, 2020.

[3] L. Elisa Celis, Sayash Kapoor, Farnood Salehi, and Nisheeth K. Vishnoi. Controlling polarization in personalization: An algorithmic framework. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[4] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM conference on recommender systems*, pages 224–232, 2018.

[5] Minmin Chen, Yuyan Wang, Can Xu, Ya Le, Mohit Sharma, Lee Richardson, Su-Lin Wu, and Ed Chi. Values of user exploration in recommender systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 85–95, New York, NY, USA, 2021. Association for Computing Machinery.

[6] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. The echo chamber effect on social media. *Proceedings of the National Academy of Sciences*, 118(9):e2023301118, 2021.

[7] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. Echo chambers in the age of misinformation. *arXiv preprint arXiv:1509.00189*, 2015.

[8] Daniel Geschke, Jan Lorenz, and Peter Holtz. The triple-filter bubble: Using agent-based modelling to test a meta-theoretical framework for the emergence of filter bubbles and echo chambers. *The British Journal of Social Psychology*, 58:129 – 149, 2018.

[9] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, 2004.

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[11] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. Degenerate feedback loops in recommender systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, jan 2019.

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

[13] Eli Lucherini, Matthew Sun, Amy A. Winecoff, and Arvind Narayanan. T-recs: A simulation tool to study the societal impact of recommender systems. *ArXiv*, abs/2107.08959, 2021.

[14] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, page 1097–1101, New York, NY, USA, 2006. Association for Computing Machinery.

[15] Martin Mladenov, Elliot Creager, Omer Ben-Porat, Kevin Swersky, Richard Zemel, and Craig Boutilier. Optimizing long-term social welfare in recommender systems: A constrained matching approach. In *International Conference on Machine Learning*, pages 6987–6998. PMLR, 2020.

[16] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686, 2014.

[17] Antti Oulasvirta, Janne P Hukkinen, and Barry Schwartz. When more is less: the paradox of choice in search engine use. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 516–523, 2009.

[18] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 881–890, New York, NY, USA, 2010. Association for Computing Machinery.

[19] Özge Sürer, Robin Burke, and Edward C Malthouse. Multistakeholder recommendation with provider constraints. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 54–62, 2018.

[20] Panagiotis Symeonidis, Ludovik Coba, and Markus Zanker. Counteracting the filter bubble in recommender systems: Novelty-aware matrix factorization. *Intelligenza Artificiale*, 13(1):37–47, 2019.

[21] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[22] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116, 2011.

[23] Mengting Wan, Jianmo Ni, Rishabh Misra, and Julian McAuley. Addressing marketing bias in product recommendations. In *Proceedings of the 13th international conference on web search and data mining*, pages 618–626, 2020.

[24] Yao Wu, Jian Cao, Guandong Xu, and Yudong Tan. Tfrom: A two-sided fairness-aware recommendation model for both customers and providers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1013–1022, 2021.

[25] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–20, 2021.

[26] Tao Zhou, Zoltan Kuscsik, Jianguo Liu, Matú Medo, Joseph R. Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107:4511 – 4515, 2008.

A T-RECS Implementation

Optimization Algorithm: Since we use a content-filtering approach to recommending items to users, we use the *ContentRecommender* module of T-RECS. This model uses non-negative least squares (NNLS) as its optimization algorithm. NNLS seeks a vector of coefficients $\hat{\mathbf{u}}$ such that

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}\in\mathbb{R}^{N_{\text{users}}}} ||\hat{I}\mathbf{u} - \mathbf{b}||_2^2$$

where $\mathbf{b}_t$ are the interactions of a user up to a timestep t and $\hat{I}$ are the predicted item attributes.

User-choice model: For the user-choice model, we use the utility and attention dependent implementation, proposed by Chaney et al. [4]. Instead of only basing interactions on the score this model also factors in the order in which items are recommended. Each user chooses an item $i_u(t)$ such that

$$u_{(t)} = \arg\max_i (rank_{u,t}(i)^\alpha \cdot S_{u,i}(t) \tag{10}$$

where α is the attention exponent and $S_{u,i}(t) = \mathbf{u} \cdot \mathbf{i}$ is the underlying user-item score.

Dynamic User Preferences: Since we investigate the effects of different diversity-inducing RS algorithms, it is crucial to dynamically model user preferences. In T-RECS, the a user's actual preferences u "drift" towards the item's attributes $\hat{i}$ that she interacted with. This is implemented using "spherical linear interpolation, such that the user's profile vector is rotated in the direction of the item vector" [13].

Recommendation model: The recommendation algorithm is based on a score function and either deterministic or probabilistic slate picking. Scoring functions determine in which order all items are ranked. The default implementation is the dot product $\hat{\mathbf{u}} \cdot \hat{\mathbf{i}}$. The T-RECS environment gives developers an interface to develop their own scoring functions. In the deterministic recommendation model, the top k items are picked and recommended to the user. In the probabilistic recommendation model, each item is assigned probability mass based on its score and then k recommendations are sampled from all possible items.

B Additional Metric Analyses

Diversity Measurement:

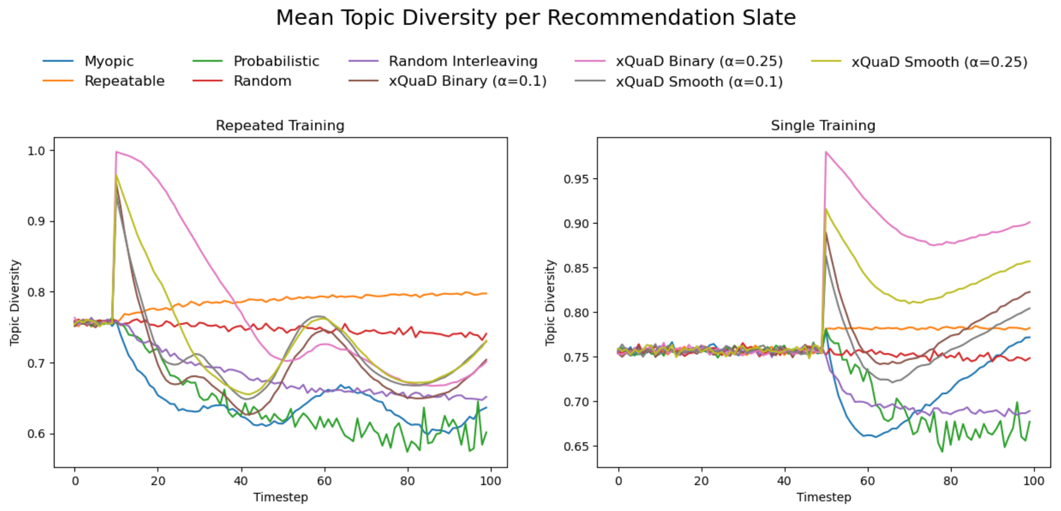


Figure 8: Mean topic diversity of all recommendation slates presented at a given timestep, by recommendation model and training mode.

Interaction Similarity Measurement:

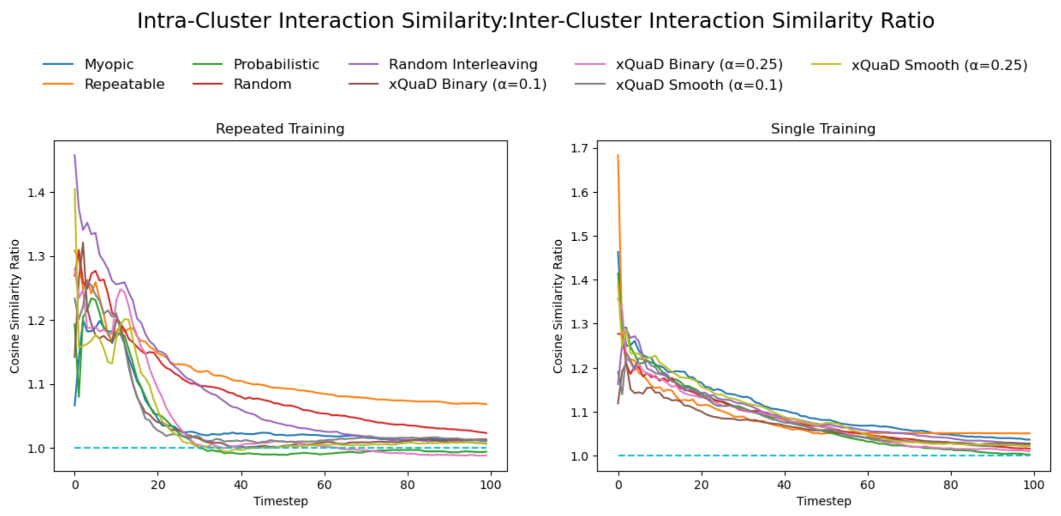
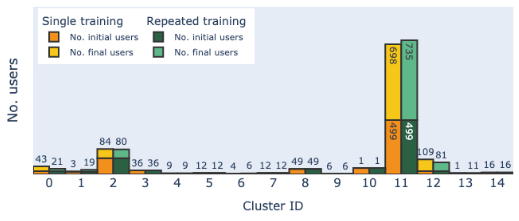
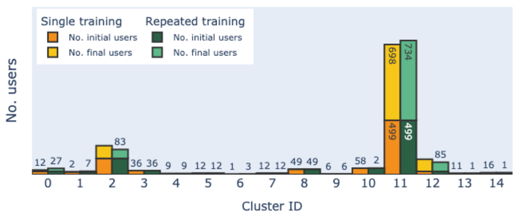
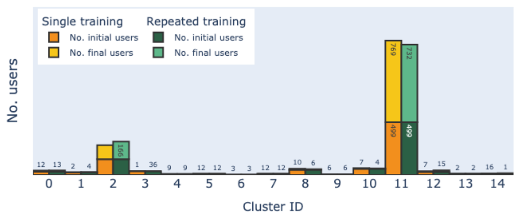


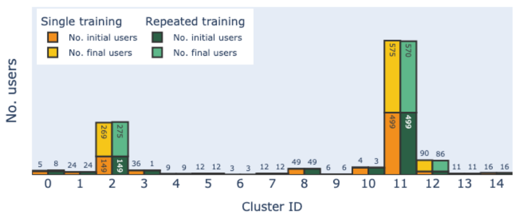
Figure 9: Ratio of mean intra-cluster interaction similarity relative to mean inter-cluster interaction similarity, under repeated training mode (left) and single training mode (right). Values greater than 1 indicate a greater degree of intra-cluster interactional similarity relative to inter-cluster interaction similarity.

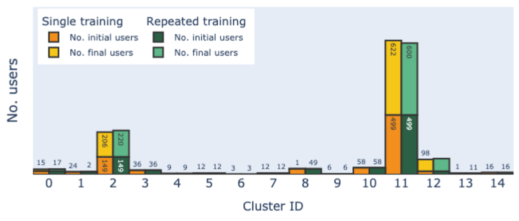
Number of Users by User Cluster:

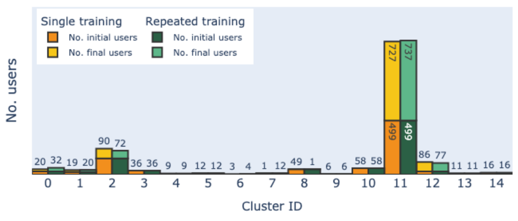

(a) Myopic Recommender

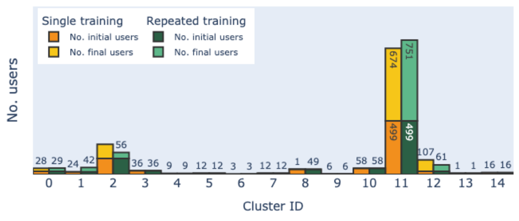

(b) Probabilistic Recommender

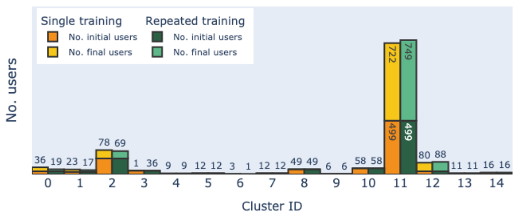

(c) Repeatable Recommender


(d) Random Recommender


(e) Random Interleaving Recommender


(f) Binary xQuAD, $\alpha = 0.1$


(g) Binary xQuAD, $\alpha = 0.25$


(h) Smooth xQuAD, $\alpha = 0.1$

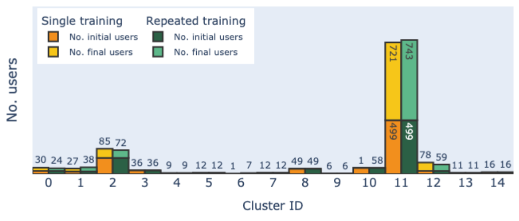
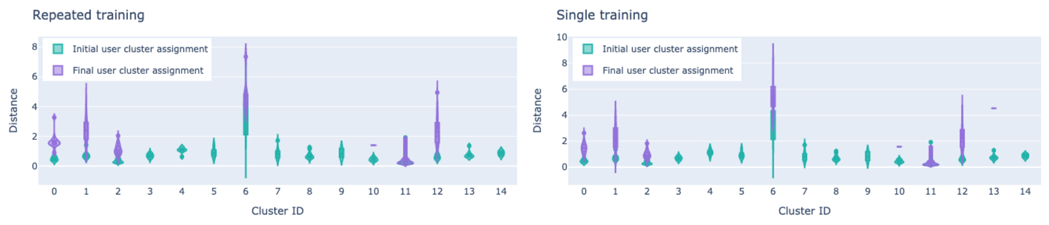
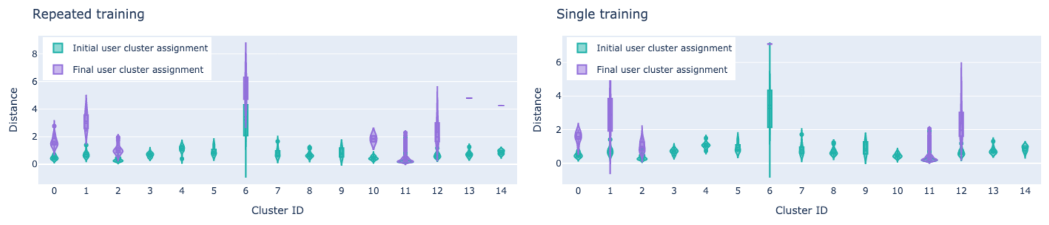

(i) Smooth xQuAD, $\alpha = 0.25$

Figure 10: Number of users per user cluster, by model and training mode.

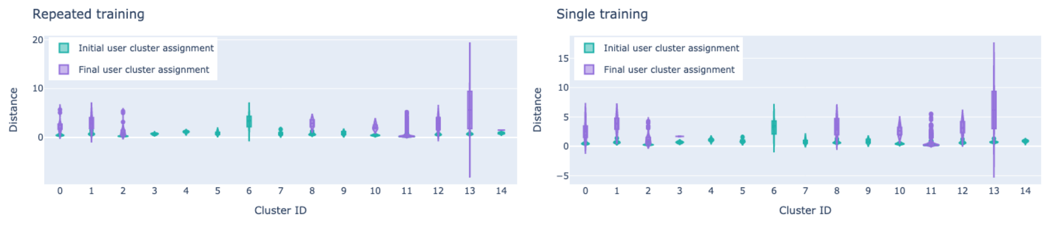
Distance From User Embeddings to User Cluster Centroids:



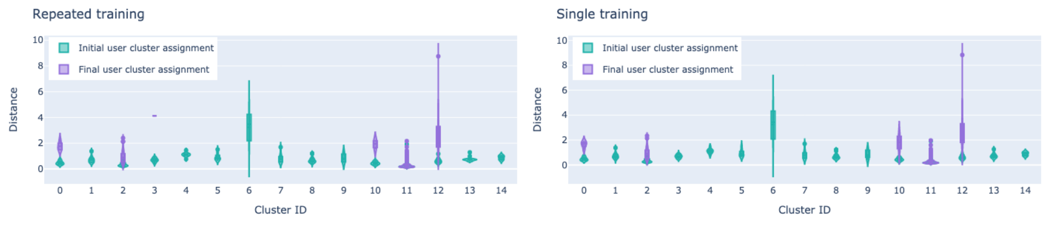
(a) Myopic Recommender



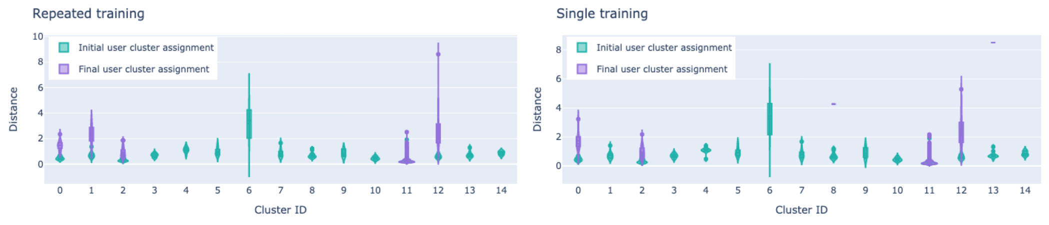
(b) Probabilistic Recommender



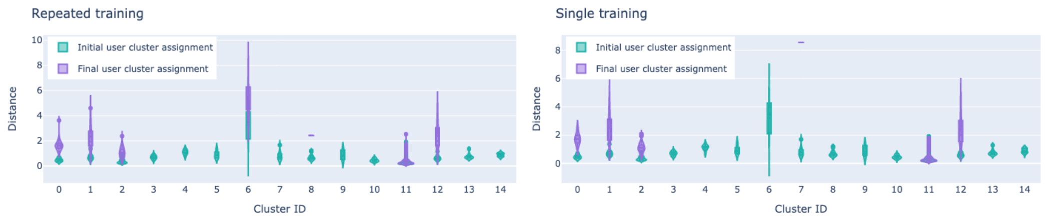
(c) Repeatable Recommender



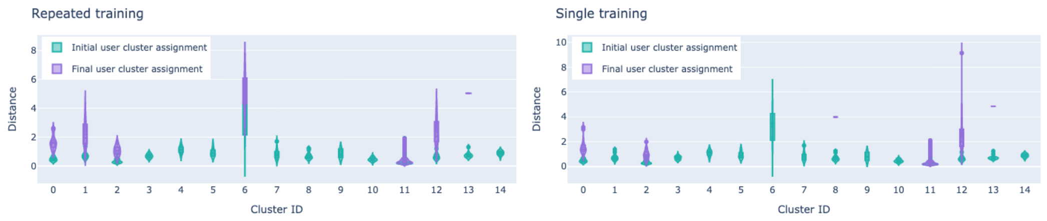
(d) Random Recommender



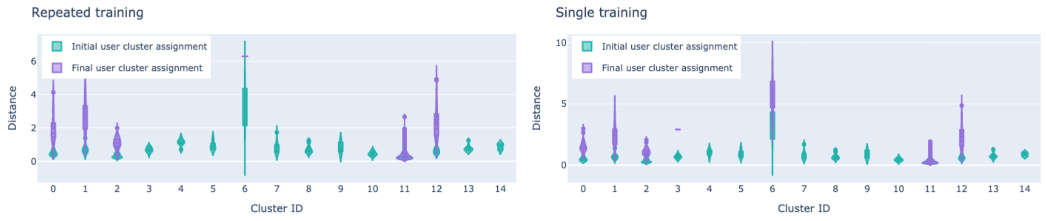
(e) Random Interleaving Recommender



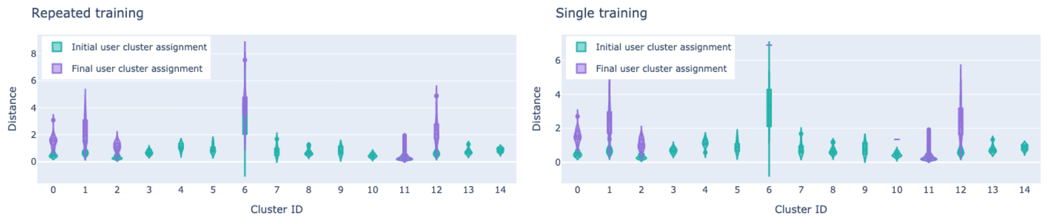
(f) Binary xQuAD Recommender, $\alpha = 0.1$



(g) Binary xQuAD Recommender, $\alpha = 0.25$



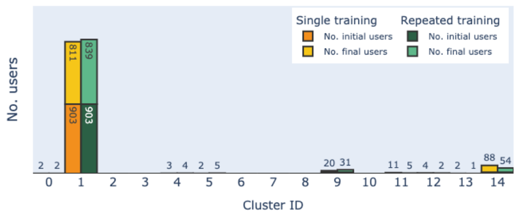
(h) Smooth xQuAD Recommender, $\alpha = 0.1$



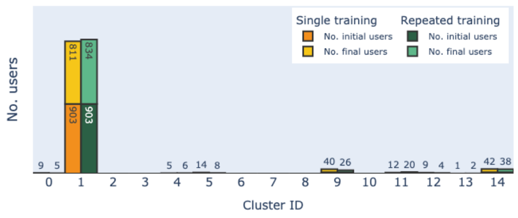
(i) Smooth xQuAD Recommender, $\alpha = 0.25$

Figure 11: Violin plots depicting the distribution of Euclidean distance between user embedding and assigned user cluster's centroid for each user cluster, by model and training mode. Distribution of Euclidean distance represented for both the initial and final user-cluster assignments.

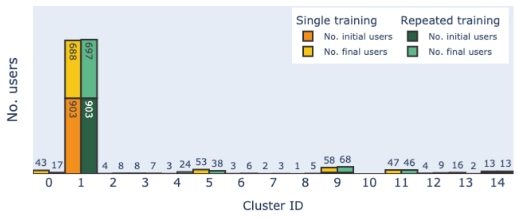
Number of Users by User-Topic Mapping:



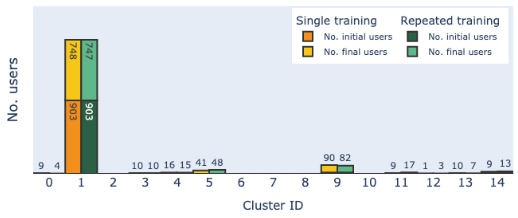
(a) Myopic Recommender



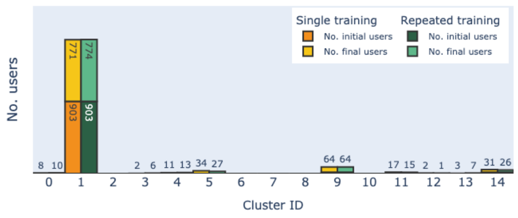
(b) Probabilistic Recommender



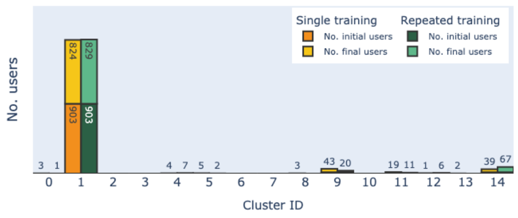
(c) Repeatable Recommender



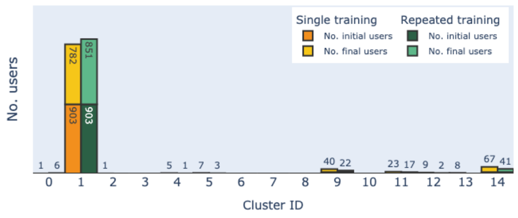
(d) Random Recommender



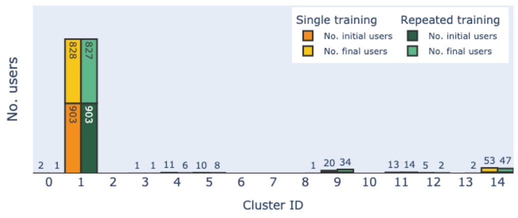
(e) Random Interleaving Recommender



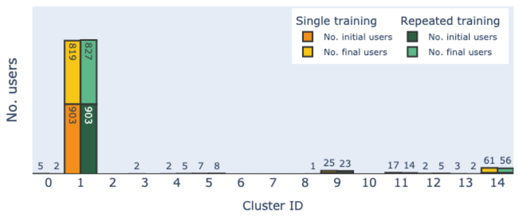
(f) Binary xQuAD, $\alpha = 0.1$



(g) Binary xQuAD, $\alpha = 0.25$



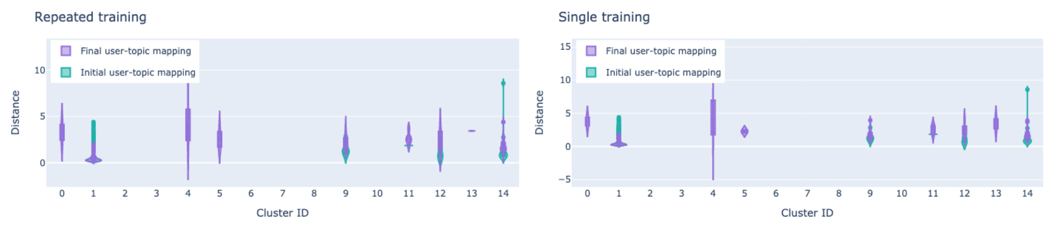
(h) Smooth xQuAD, $\alpha = 0.1$



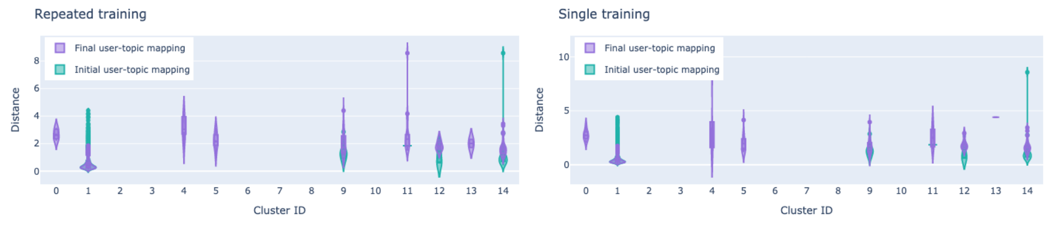
(i) Smooth xQuAD, $\alpha = 0.25$

Figure 12: Number of users per user-topic mapping, by model and training mode.

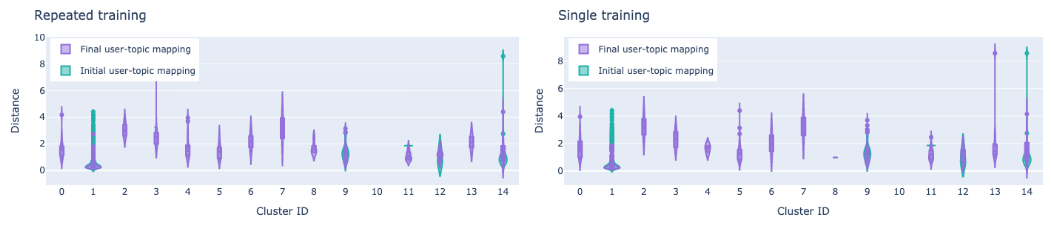
Distance From User Embeddings to Topic Centroids:



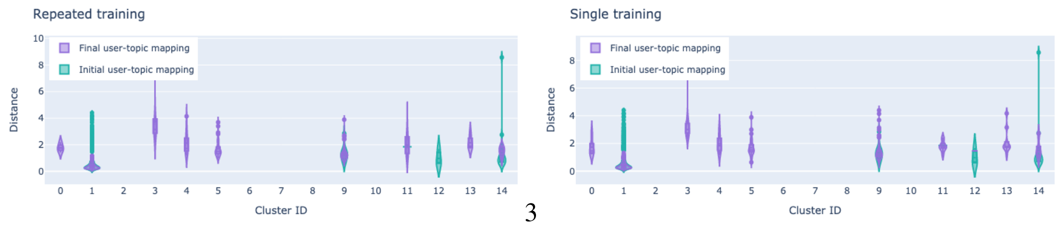
(a) Myopic Recommender



(b) Probabilistic Recommender

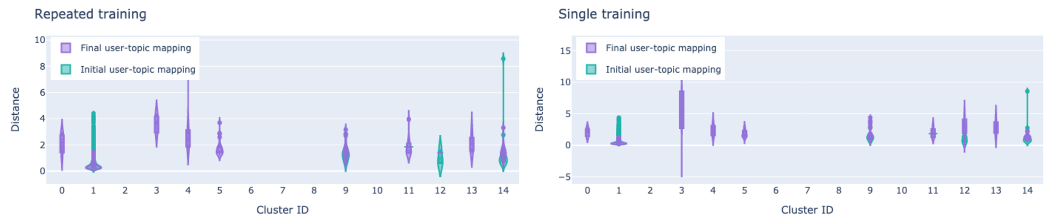


(c) Repeatable Recommender

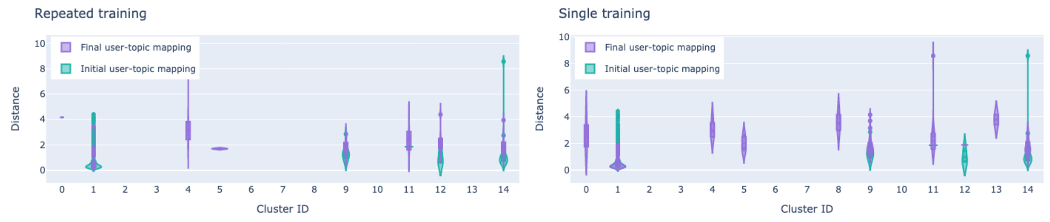


3

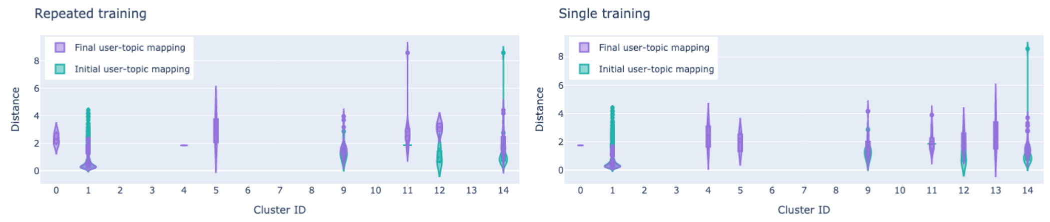
(d) Random Recommender



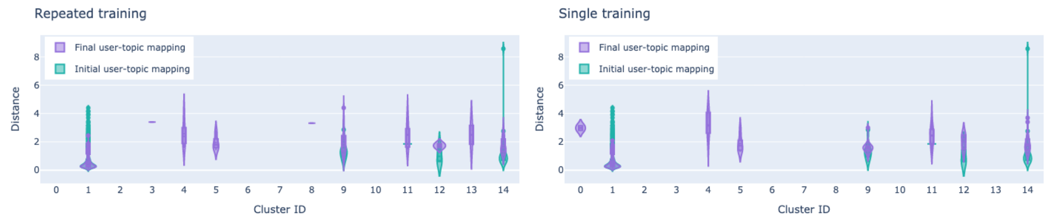
(e) Random Interleaving Recommender



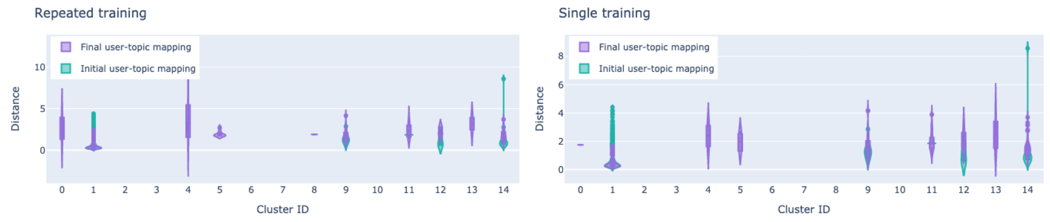
(f) Binary xQuAD Recommender, $\alpha = 0.1$



(g) Binary xQuAD Recommender, $\alpha = 0.25$



(h) Smooth xQuAD Recommender, $\alpha = 0.1$



(i) Smooth xQuAD Recommender, $\alpha = 0.25$

Figure 13: Violin plots depicting the distribution of Euclidean distance between user embedding and assigned topic cluster's centroid for each user-topic cluster, by model and training mode. Distribution of Euclidean distance represented for both the initial and final user-topic mappings.