Bubble Bursting Implementations and Dynamics In Recommender Systems

Fair and Robust Algorithms
Project Report

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

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

Abstract

Recommender systems (RSs), which offer personalized suggestions to users, power many of today's social media, e-commerce, and entertainment services. However, they have been known to create filter bubbles, a term used to describe the phenomenon wherein users are only shown content that aligns with their preconceived tastes and opinions. Past research has shown filter bubbles to be related to ideological extremism and systematic bias against marginalized groups. We mitigate such effects by introducing new notions of exploration into our recommender system (RS) model and measuring the trade-offs between accuracy and other metrics, such as diversity, novelty, and serendipity. We also examine the effect of filter bubble-mitigation strategies on user behavior homogenization by analyzing inter- versus intra-bubble homogeneity dynamics. You can find our implementation at `https://github.com/jannikjw/fair-recommender-systems`.

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.

Preprint. Under review.

2 Related work

Fairness. Existing research generally defines fairness with respect to either users [2, 21] or items [18, 23], 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 [22, 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 [20, 13]. 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, 24].

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 [14]. 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, 12]. 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 [14]. This "drift" is highlighted in Figure 1 as arrow *(b)*.

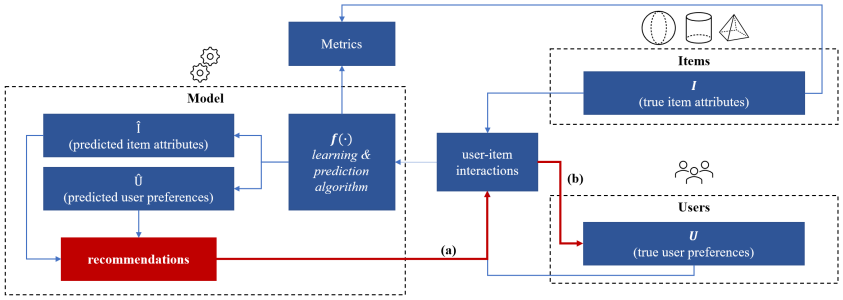


Figure 1: A conceptual diagram of the design components and system interactions of a T-RECS simulation. Lines with arrows indicate inputs fed into outputs. Dashed grey lines indicate conceptual components. The red arrows (a) and (b) are the main levers in influencing long-term user preferences.

The full T-RECS framework and implementation can be found in [14]. 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 "startup" mode, where new items are randomly recommended to users in order to maximize exploration. After startup mode, the model is ran in training mode. In training mode, the default algorithm is myopic and aims to maximize the predicted reward $r(u, i)$ of user u and item i at each timestep t.

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 Measurement

While myopic RSs might increase a metric like engagement or time spent on the platform in the short-term, this likely hurts user experience in the long run [5, 15]. For this reason, there is a growing body of work examining metrics other than accuracy [15, 3, 10, 4]. These additional measurements can be split into two buckets. The first set of metrics relate to exploration, including properties such as diversity, novelty, and serendipity. We implement [5]'s formulation of these metrics, tailoring them to our matrix-based RS simulation. These are properties that can be directly influenced by changing the recommendations (compare link **(a)** in Figure 1). The second set of metrics focuses on how users interact with the content that they are recommended. These metrics are more concerned with the actual users' preferences on which any RS can only have an indirect influence (compare link **(b)** in Figure 1). Metrics in this set include interaction similarity and diversity in preferences [4, 14]. We have detailed our measurements in Appendix B

4.2 Attributes

We first define two attributes that are important to our measurement and analysis. We then detail and define the metrics that use them.

Topic clusters. A topic cluster for each item is produced by: 1) taking the embedding vector of them item, created using NMF; 2) using k-mean to cluster the learning embeddings into 25 clusters; 3) assigning the nearest cluster to each item.

Filter bubbles. Each topic cluster is considered a potential filter bubble. We assume that each user is considered to be inside of a filter bubble, which can of course be of varying sizes and densities. Each user is initially assigned to the nearest topic cluster. This means that some topic clusters might not be assigned to any user. We want to observe whether the closest clusters are different after the simulation and the degree to which different ranking mechanisms affect stronger or weaker clustering.

4.3 Ranking Methods

In this study, we examine the effect of different ranking mechanisms on a dynamic RS. We compare these effects to a myopic RS, i.e. one that only optimizes for accurate recommendations. The myopic ranking creates recommendation sets by picking the top k elements from the predicted scores, which is computed as:

$$r(\hat{u}, \hat{i}) = \hat{u} \cdot \hat{i}. \tag{1}$$

4.3.1 Top-k Re-Ranking

The first algorithm that we implemented is a naive top-k re-ranking method, where the model's predicted scores for the top k items are reversed. The value of k is equal to the number of items presented in a slate. For our experiments, we have k equal to 10.

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

4.3.2 Cosine Similarity Regularization

Our next ranking method is cosine similarity regularization. We use the cosine similarity between user preferences and item attributes to regularize the predicted scores using the hyperparameter λ. We expect this to reduce the number of recommendations that are very aligned with the user's preferences and thus enable users to find undiscovered preferences before the RS converges to a single topic.

$$\begin{aligned} r_{\text{cosine}}(\hat{u}, \hat{i}) &= \hat{u} \cdot \hat{i} - \lambda \left[\cos(\hat{u}, \hat{i})\right] \\ &= (\hat{u} \cdot \hat{i}) \left[1 - \frac{\lambda}{||\hat{u}|| \cdot ||\hat{i}||}\right] \end{aligned} \tag{2}$$

After computing the scores according to Equation 2, we exponentiate the regularized scores as $e^{r_{\text{cosine}}(\hat{u},\hat{i})}$ to avoid negative values.

4.3.3 Entropy Regularization

Entropy regularization promotes recommending content that is less pertinent to the predicted user preferences by encouraging the reward r to have a high entropy [5]. Previous works observed that adding entropy to policies in reinforcement learning (RL) discourages premature convergence. We add entropy to our matrix factorization approach to investigate whether this generalizes to non-RL RSs such as ours. The entropy-regularized reward is defined as

$$r_{\text{entropy}}(\hat{u}, \hat{i}) = P_r(\hat{u}|\hat{i}) + \lambda H\left(P_r(\hat{i}|\hat{u})\right), \tag{3}$$

where the entropy of $P_r(\hat{i}|\hat{u})$ of is defined as $H(P_r(\hat{i}|\hat{u})) = \sum_{\hat{a}\in\hat{I}} P_r(\hat{a}|\hat{u}) \cdot \log P_r(\hat{a}|\hat{u}) - \sum_{\hat{a}\in\hat{I}\setminus\hat{i}} P_r(\hat{a}|\hat{u}) \cdot \log P_r(\hat{a}|\hat{u})$. λ controls the strength of the regularization. With increasing regularization, the ranking function pushes the score values to be closer to the uniform distribution across the set of items I.

5 Experiments

We evaluate our proposed ranking algorithms (Sec. 4.3) through extensive experiments to measure the proposed metrics (Sec. 4.1) using the T-RECS environment. We examine the temporal development of the defined metrics and analyze whether the final user-topic cluster assignments differ between ranking mechanisms. We provide a detailed evaluation of each of the different ranking mechanisms and compare their effects on the actual user preferences. We outline the simulator, describe our dataset and trained embeddings, and discuss 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 4 with $\alpha = -0.8$. The actual user preferences are assumed to drift towards each interacted item by 0.05. The available items and their actual attributes are static. We run all experiments with the same environment assumptions, first in "startup mode" for 10 timesteps and then in "repeated training mode" for 100 timesteps.

5.2 Datasets

Like Mladenov et al. [15], we train an embedding on the Movielens dataset [9] using non-negative matrix factorization on a sparse matrix of user-movie engagements. The pair of matrix factors are then used as embedding vectors for users and items for the simulator, where, for this dataset, a movie constitutes an item. User-item rewards are computed as $r(\vec{u}, \vec{i}) = \vec{u} \cdot \vec{i}$. Further details can be found in Appendix C.

5.3 Results

Movement between Clusters: Our ultimate goal is to show that, by employing different ranking mechanisms, we can nudge users to move from one topic cluster to another. We examine the cluster assignment and how the actual/ initial user preferences have changed after the simulation. Our hypothesis was that the myopic ranking would drive users to interact with many small clusters and that they would be more easily separable. We visualize users and item clusters using TSNE [19] with $perplexity = 50$. In Figure 2, we can see that the myopic ranking instead leads to an increasing number of clusters with two dominant ones (1 and 14) compared to the single dominant cluster at the beginning (14) as well as many smaller ones. This suggests that with dynamic user preferences, the myopic ranking nudges users towards more distinct preferences, indicating bubble-building behavior.

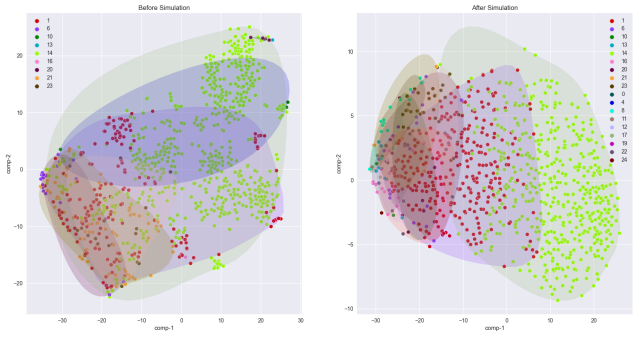
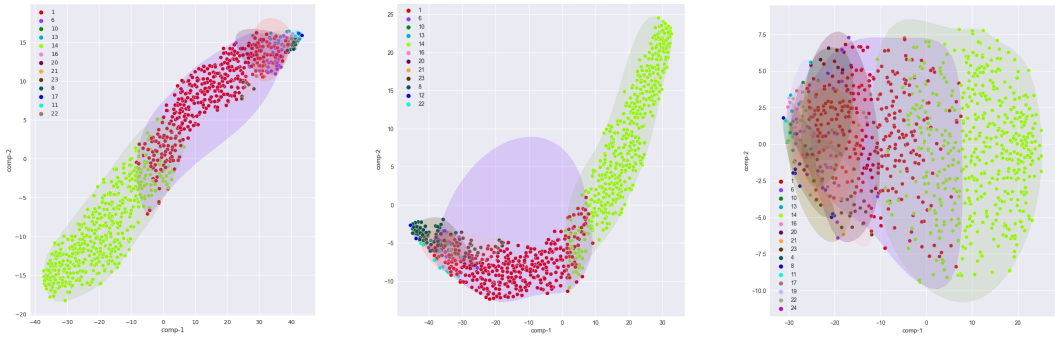


Figure 2: User and their respective clusters before the simulation (left) and after the simulation using the myopic ranking (right).

Entropy Regularization: Regularizing entropy leads to fewer clusters than the myopic policy, as seen in 3a. It creates two large clusters and a number of smaller but non-distinct ones. This indicates that entropy regularization reduces filter bubbles compared to the myopic RS. This is supported by Table 1 which shows that entropy induces more movement between clusters and reduces the average distance of users to other clusters. *Cosine Similarity Regularization*: Figure 3b shows both expected and unexpected results for the cosine similarity regularization. We expect the number of clusters to decrease as users are exposed to different interests and thus their preferences are less distinct from each other. Additionally, we also observe that more users changed clusters than before (see Table 1). However, we did not expect clusters to be seemingly more separable than before (i.e. clusters 8 and 14 are not overlapping with the other cluster compared to the beginning). As seen in Table 1, cosine regularization also reduces the average distance between users and other clusters the most. *Top-k Reranking*: Applying top-k reranking leads to nearly identical clusters as the myopic RS (see Figure 3c). 1 also shows that the results for myopic and top-k are almost identical and therefore does not help significantly in nudging users closer together across clusters.



(a) Entropy Regularization (b) Cosine Regularization (c) Top-k Reranking

Figure 3: Results from mapping users to item clusters after running the T-RECS simulation using different ranking mechanisms. From left to right: entropy regularization, cosine regularization, top-k reranking.

	Begin	Myopic	Entropy	Cosine	Top-K
Number of Cluster Changes		328	341	378	329
Average Distance to Other Clusters	2.363	2.269	2.257	2.231	2.269

Table 1: Number of users who change clusters and their average distance to other clusters.

We deduce that myopic RSs tend to separate dynamic user preferences more clearly over time and thus isolate users more from new interests. Furthermore, we can use alternative ranking mechanisms that penalize close similarity to change how users behave over time.

Homogenization and Algorithmic Confounding: For our analysis of the relationship between homogenization and filter-bubble mitigation strategies, we initially hypothesized that algorithms aimed at minimizing filter bubbles would yield increased inter-cluster homogeneity and decreased intra-cluster homogeneity. However, our results do not support this hypothesis. The first and second columns of figure 4 indicate that inter- and intra-cluster similarity change in parallel with one another, for all ranking algorithms. Moreover, column 3 shows the intra- to inter-cluster similarity ratios, where values greater than 1 correlate to greater intra-cluster (relative to inter-cluster) similarity. While the intra-cluster similarity is indeed greater than inter-cluster similarity, the degree is relatively small, with similarity ratios hovering around 1.0 for the myopic and re-ranking strategies.

To better understand the interaction between algorithmic confounding and exploration, we compared user behavior homogeneity under one-phase training and repeated training conditions. The findings of [4] showed that repeated training amplifies the effect of algorithmic confounding, and consequently homogenization. Our results suggest that:

1. For the standard myopic algorithm, algorithmic confounding introduces similar degrees of homogeneity at the global and local levels (w.r.t. filter-bubbles). This can be observed in the right column of Figure 5, where the similarity ratios under 1-phase and repeated training conditions mirror each other.
2. For exploration-oriented ranking strategies, observation (1) does not necessarily hold. This is suggested by the left column of Figure 5, where, under 1-phase training, cosine similarity regularization ($\lambda = 0.1$) results in greater intra-cluster homogeneity (relative to intra-cluster homogeneity), as compared with repeated training.

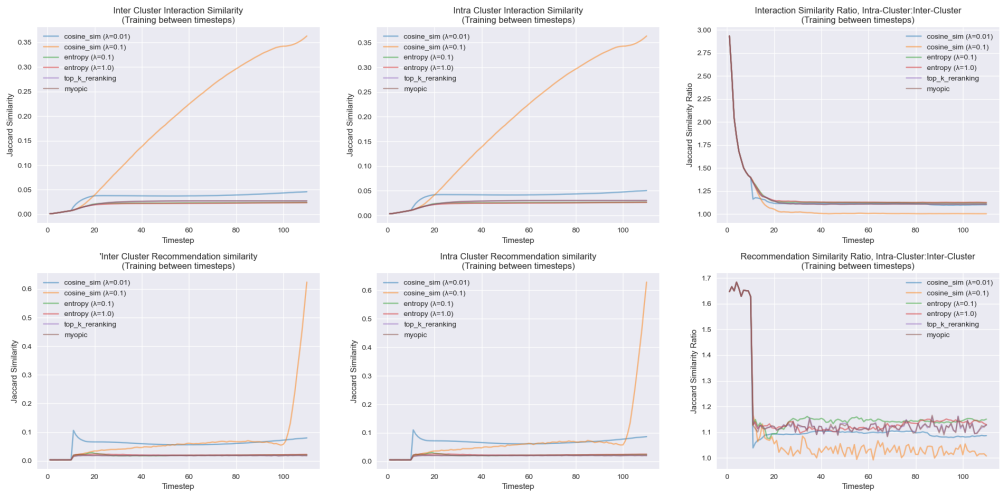


Figure 4: Analysis of intra- vs. inter-cluster homogeneity. The filter bubble-mitigating strategies are associated with greater intra- and inter-cluster homogeneity, as shown in columns 1 and 2. Increases in homogeneity affect local user clusters and the global user population at similar rates, as shown in column 3. Note that entropy regularization ($\lambda = 0.1, 1.0$), cosine similarity regularization ($\lambda = 0.01$), and top-k re-ranking were omitted due to comparatively moderate effects.

Trade-off between Accuracy and Exploration: As previous works have shown, we expect there to be a trade-off between accuracy and diversity. Figure 6 shows MSE and recall, and Figure 7 our

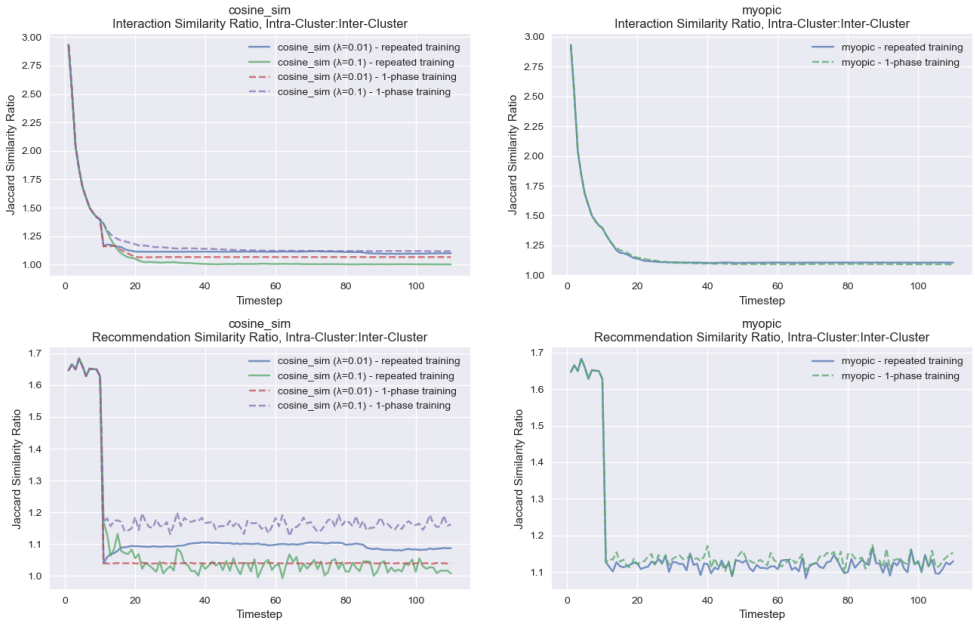


Figure 5: Ratio of similarity measurements for cosine similarity regulation and myopic ranking and under repeated versus 1-phase training conditions. These measurements include interaction similarity and recommendation similarity. Larger values indicate a larger difference in homogenization between intra- and inter-cluster user groups.

exploration metrics over time. First, this trade-off is indicated by the MSE and diversity curves. Cosine similarity ($\lambda = 0.01$) has the highest error and the highest diversity. Cosine regularization with $\lambda = 0.1$ is the opposite example, MSE is the lowest for most timesteps and diversity is continually increasing (albeit at a low absolute value). This can make it a good long-term focused algorithm. Additionally, a higher λ value indicates both higher novelty convergence and higher serendipity. It is however important to note that cosine similarity regularization displays numerical instability and thus needs further development.

Second, entropy regularization outperforms the accuracy of the myopic policy in both recall and MSE. This supports the argument that greedy optimization might not yield the highest accuracy and that exploration is valuable also from an accuracy perspective.

Third, we note that top-k re-ranking does not increase any of the exploration metrics and thus does not achieve our desired outcome.

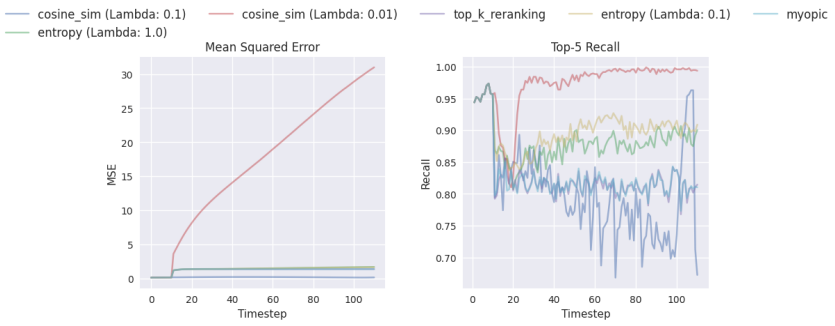


Figure 6: Accuracy Metrics. Mean Squared Error (MSE) and Top-5 Recall for the simulation by the different implemented ranking mechanisms.

Finally, we discover that the cosine similarity regularization with $\lambda = 0.01$, entropy regularization with $\lambda = 0.1$, and $\lambda = 1.0$, all outperform the myopic policy in novelty and recall while providing less slate diversity and similar serendipity. We conclude, that RSs can be optimized using different ranking functions without necessarily sacrificing accuracy.

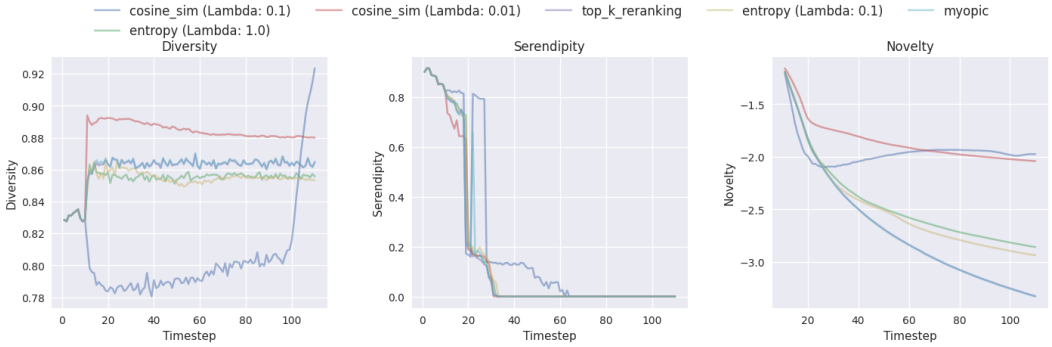


Figure 7: Exploration Metrics. Diversity, Serendipity, and Novelty for the simulation by the different implemented ranking mechanisms.

Worst-Group Impact: The final dynamic of interest to us is the interaction between exploration and user fairness. Figure 8 shows the difference between average MSE and worst-user cluster MSE for each of the implemented ranking algorithms. MSE for the top-k reranking algorithm mirrors that of the myopic policy. Moreover, these 2 algorithms show the greatest disparity in average MSE versus worst-group MSE. Generally, ΔMSE is highest during the beginning training phase for all ranking algorithms except for cosine similarity regularization.

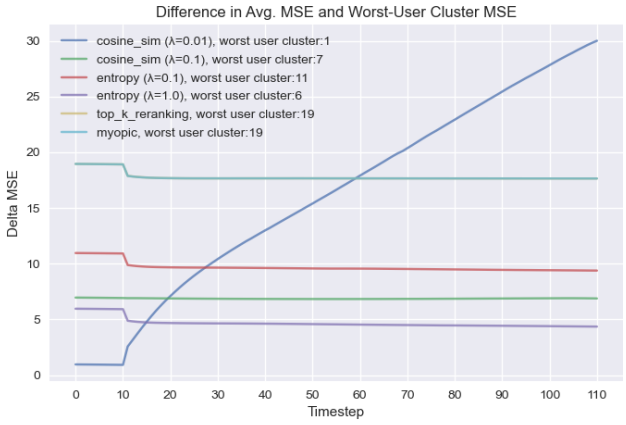


Figure 8: Difference in average MSE and worst user cluster MSE for the different ranking algorithms. All models were trained for 10 timesteps and then ran for 100 timesteps.

6 Conclusion

To recap, the primary objectives of this project are to identify filter bubble-mitigation strategies and understand how these strategies interact with the extensive dynamics of an RS ecosystem. While we are able to show different changes in user preferences, our results are not conclusive. 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 dataset, which is heavily skewed towards a few movies, which consequently results in a skewed number of items. Moreover, with only 1682 items, the Movielens dataset is small and only allows for a limited number of item clusters. On the

other hand, large RSs, like those deployed by YouTube or Netflix, contain millions of items and can support more clusters and more accurate item representations.

Second, the ranking functions presented in this report are made to work seamlessly with the T-RECS environment. Building off of this work, the development of additional, complex ranking algorithms would be very valuable—for example, algorithms that optimize for different definitions of fairness, such as the content fairness approach presented by Celis et al. [3].

Third, while our results do not support our initial hypothesis regarding intra- vs. inter-cluster homogeneity, they also do not refute this hypothesis. While there is abundant research on the effects of both algorithmic confounding 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] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5:19:1–19:19, 2016.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[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] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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

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

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

$$u_{(t)} = \arg\max_i (rank_{u,t}(i)^\alpha \cdot S_{u,i}(t) \tag{4}$$

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" [14].

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 Evaluation Metrics

B.1 Accuracy

Likely the most important metric for any RS, accuracy 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

$$MSE = \frac{1}{N_{\text{users}}} \sum_{l=1}^{N_{\text{users}}} (u_l - \hat{u}_l)^2. \tag{5}$$

For this, it is important to remember that the actual user preferences are changed during 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

$$Recall = \frac{1}{N_{\text{users}}} \sum_{l=1}^{N_{\text{users}}} \sum_{j=1}^{k} 1_{A_j^r \in E_t(u)} \tag{6}$$

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 rank items such that the user's choice corresponds to highly ranked items.

B.2 Diversity

Diversity measures the number of distinct topics the recommended set contains. We use the average dissimilarity of all pairs of items in the set by Chen et al. [5].

$$Diversity(A^r) = 1 - \frac{1}{|A^r|(|A^r| - 1)} \sum_{i,j \in A^r, ij} sim(i, j) \tag{7}$$

where the $sim(i, j) = 1$ between two items i and j if i and j belong to the same topic cluster, and 0 otherwise.

B.3 Novelty

We define novelty similarly to Chen et al. [5]. That is, novelty concerns the RS capacity to suggest something the user is unlikely to know about or discover by themselves. Novelty is based on the self-information from Zhou et al. [24] which measures the unexpectedness of a recommended item relative to its global popularity.

$$\begin{aligned}\mathcal{I}(i) &= -\log p(i) = -\log \frac{N_{\text{users consumed item } i}}{N_{\text{users}}} \\ &= -\log N_{\text{users consumed item } i} + \textit{const}\end{aligned}$$

Here $p(i)$ measures the chance a random user would have consumer item i. Following this, globally under-explored items will have higher self-information. Novelty under a scoring function r is then defined as

$$Novelty(r) = \sum_{\hat{i}\in\hat{I}} \sum_{\hat{u}\in\hat{U}} P(\hat{i}|r(\hat{u},\hat{i}))\mathcal{I}(\hat{i}). \tag{8}$$

A reward function $r(\cdot)$ that casts more probability mass on items with higher self-information, is therefore deemed more novel.

B.4 Serendipity

Serendipity captures the degree of unexpectedness associated with recommending a content item to a particular user [5]. The calculation is based on two important components: relevance and unexpectedness—the content should be interesting but surprising to the user. As such, serendipity measures the ability of the RS to discover new interests of the user which were previously unknown to the model. The serendipity value of a particular recommendation at a given timestep i_t is defined with respect to the topic interaction history $\mathcal{I}$ of the user, such that:

$$Serendipity^{topic}(i_t|u_t,\mathcal{I}_t) = \begin{cases} 1, & \text{if } r(\hat{u}_t,\hat{i}_t) > 0 \text{ and the topic cluster to} \\ & \text{which } i_t \text{ belongs is not present in } \mathcal{I}_t \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

B.5 Inter-Cluster and Intra-Cluster Similarity

One of our objectives is to expand upon the work conducted by Chaney et al. [4], exploring the relationship between homogenization and filter-bubble mitigation strategies in an RS ecosystem. Following [4]'s approach, we measure user behavior homogenization via the average Jaccard index for pairs of users. Given a pair of users $< v, w >\in U$, user interaction similarity is computed as the Jaccard index at timestep t, such that:

$$\textit{JI}_{v,w}(t) = \frac{|\mathcal{I}_v \cup \mathcal{I}_w|}{|\mathcal{I}_v \cap \mathcal{I}_w|}$$

At each timestep, we measure inter-cluster and intra-cluster interaction similarity and recommendation similarity. Interaction similarity measures the average Jaccard index of interacted items for pairs of users at every timestep. Recommendation similarity measures the average Jaccard index of recommended items for pairs of users at every timestep. Inter- vs. intra-cluster groups are defined using our operationalization of filter bubbles, as defined above, such that intra-cluster similarity is measured as the average Jaccard index (w.r.t. either interactions or recommendations) between pairs of users that belong to the same user cluster. Given a user cluster $C_V \in \mathcal{C}$, where V is the subset of users belonging to C_V, the intra-cluster interaction similarity is then computed as:

$$Sim_{Intra} = \frac{1}{|V|} \sum_{v_i,v_j\in V} \textit{JI}_{v_i,v_j}(t),$$

and the inter-cluster interaction similarity is computed using pairs of users that belong to different user clusters, such that:

$$Sim_{Inter} = \frac{1}{|V||W|} \sum_{v\in V, w\in W} \textit{JI}_{v,w}(t), \text{ where } V \neq W$$

The same approach is used to compute inter- and intra-cluster recommendation similarity, but instead using each user's history of recommended items, $\mathcal{R}_v$ and $\mathcal{R}_w$.

C Datasets

From the MovieLens dataset, we generate actual user preferences U and item attributes I using non-negative matrix factorization. We use the distribution containing about 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 $E_{i,j} = \mathbb{1}(R_{i,j})$. The embeddings of our actual preferences and attributes are formed by solving the following optimization problem

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

which yield 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 [11] until convergence. We use embedding rank $N_{\text{attributes}} = 20$.