Data Science

Capstone Report - Fall 2022

Popularity Bias and Fairness in Music Recommendations

Dennis Hu

supervised by
Hongyi Wen

Preface

The essence of any recommender system is that they never perform exactly the same on two different users. However, we are interested in the fact that as much as people have varied music tastes, currently deployed music recommenders often homogenize music consumption more than diversified them. In this essay, we dig deep into the inner workings behind popular recommenders and suggest directions in which headways can be made to a world of serendipitous music encounters.

Acknowledgements

Special thanks to Professor Hongyi Wen for his guidance and devotion throughout this semester, we couldn't do this project without the suggestions you gave us.

Abstract

Recommender systems are algorithms matching users to content based on the individual's history of interactions. As a probabilistic model, it is prone to suffer from popularity bias, which results in a self-enhancing pattern of only recommending highly popular items to all users regardless of personal preference. In this study, we use the nearest-neighbor-based hybrid continuation model to minimize the requirements of computational resources while ensuring accuracy. After applying two different reranking algorithms that allow for individualized adjustments in popularity preferences, we conclude that future recommenders can benefit from adopting our model to round out a more personalized experience for users, and to create a more equitable environment for content creators.

Keywords

Music Recommender Systems; Spotify; Popularity Bias; Item-based Re-ranking; Intra List Diversity Metrics(ILD); Hybrid Recommenders

Contents

1 Introduction

As one of the most prevailing applications of machine learning in industry, recommender systems are algorithms used to match users to content based on the individual's history of interactions. With the abundance of commodities increasing and thus becoming impossible for a user to fully experience, the role of the recommender system is becoming more prominent in affecting people's information feeds and decision-making in terms of what music to listen to, what content to watch, or what products to purchase.

Modern recommender systems have evolved rapidly, as have deep learning models that are well-optimized for overall performance. A recommendation algorithm that focuses solely on average performance, on the other hand, may reinforce the exposure bias in the training data and exacerbate the "rich-get-richer" effect, trapping users in an experience among certain subgroups. This is a common problem that many recommender systems suffer from–popularity bias. This phenomenon is especially pronounced in music streaming platforms: more and more viral hits are produced by short video content such as those in TikTok, and users often have no choice but to hear those songs in every playlist. However, recommending the ignored products in the "long tail" is critical for businesses as they are less likely to be discovered. With this in mind, a more fair and niche-sensitive recommender is needed.

Most modern music streaming platforms, including Spotify, Apple Music, and NetEase Music include some version of automatic music recommendation. Given the overloaded state of specific terms in music recommendation, we first clarify them to prevent further obfuscation. In this paper, each individual song is called a track. Each track will have a unique ID, as will artists to whom the tracks are attributed. Users and platforms can create and publish certain sets of tracks called playlists. In Spotify, our platform of focus, music recommendations come in four main forms: personal radios, artist-centric recommendation, playlist recommendation, and playlist continuation. Personal radios are virtually endless song recommendations based on each user's interaction profile. The radio does not take any specific track as input and purely generates recommendations from a user's listening history. Artist-centric recommendations are more complex with two layers: first, the system determines whether a user is likely to gain utility from listening to a certain artist; then, the system associates similar tracks and playlists to the artist's style without consideration of the user. Playlist recommendations are the most traditional and straightforward of the recommendation types: users are recommended playlists based on their listening history, and no specific track is taken as input. Lastly, playlist continuation is an emerging task of appending tracks to a given set of tracks (or one single track) based on the coherence of style, genre, mood, etc. With playlist continuation, users have the freedom to dictate which tracks should be given consideration and thus match the fleeting nature of human preference for music. Our research will mainly focus on the issue of addressing popularity-induced fairness issues in playlist continuations.

2 Related Work

2.1 Music Recommender Systems

Stemming from the general recommender systems, Music Recommender Systems(MRSs) have exploded in popularity. The basic method in a recommender system can be divided into three major categories: content-based filtering, collaborative filtering, and hybridized methods.

Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous interaction with the content. It includes different techniques such as the Bayesian approach, deep learning, decision tree, and cosine similarity [1, 2, 3]. For example, a content-based recommendation algorithm for learning resources suggested by J. Shu et al. [1] mainly employs a content-based learning resources recommendation algorithm by convolutional neural network (CBCNN). Existing language reseources are processed as text and outputs are organized by a latent factor model, which is regularized by L1-norm.

Collaborative filtering based recommender systems allow the active user to receive recommendations by using information about other users and their relation with the item[4].

In most cases, MRSs nowadays would provide songs recommendations with high accuracy using mentioned models, however, such recommender systems are far from being perfect when dealing with evaluation of MRSs that are beyond accuracy [5]. For example, instead of receiving a recommendation of a list of popular songs, a user might want to listen to a less well-known one, and thus the classic recommender systems and the related evaluation methods need to be moderated to meet that concern. Among the evaluation problems, the popularity bias, as one of the trickiest, is the problem that we will focus on in our research.

2.2 Popularity Bias

In the economics of entertainment, the term "long tail" is often used to refer to the fact that the majority of involvement with cultural products such as books, movies, music, etc. is concentrated in a few of the most popular superstars. Meanwhile, the long tail of niche content is vast but mainly overlooked. The phenomenon has been a long-standing one that precedes the existence of any recommender system. However, since the advent of recommender systems [6], the popularity bias issue has been more pronounced, especially in music recommendations [5].

The exacerbation of popularity bias has been attributed to a number of factors: collaborative filtering due to the model's preference for items with more rating [7], feedback loops with users' interaction data confirming the model's bias [8], and confounding involvement of selection bias which involves conscious human decisions [9]. This can be summed into two aspects: data-wise, the users' interaction patterns follow that of the initial long-tail distribution, resulting in the interaction data focusing on the small number of popular items. Model-wise, With more data points to train on the more popular items, recommendation systems tend to assign a lower risk to popular items when compared to the more niche items, even when they might be equally preferred by a particular user [10].

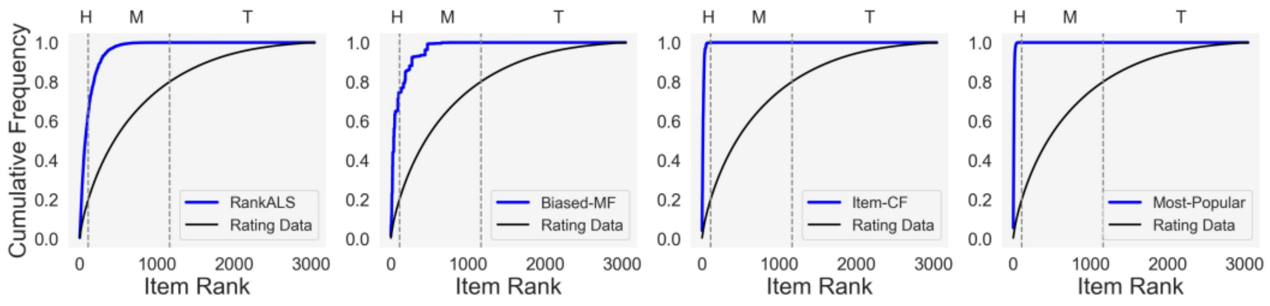

Figure 1: impact of different recommenders on item distribution

2.3 Track Popularity

Less popular items may be treated unfairly as a result of the popularity bias. Figure 1 shows different recommenders and their resulting item distribution. They compare the popularity of original items in the data to the popularity of items in recommendations generated by four well-known recommendation algorithms. (RankALS [11], Biased Matrix Factorization (Biased-MF) [12], Item-based Collaborative Filtering [13], and a naive recommendation that simply recommends the same most popular items to every user). The figure is adapted from Abdollahpouri et al. [7] The horizontal axis displays each item's rank when sorted from most popular to least popular. The black curve represents the cumulative frequency of ratings for various items at a given rank. Intuitively, if the blue curve floats high above the black with a sharp elbow point, that means a recommender is highly biased toward popular items. In many consumer taste domains where recommender systems are commonly deployed, such as video recommendation, the same highly biased distribution of user interest is also prevalent. A music catalog, meaning a whole archive of songs that are available for listening, may include big-name artists who have stream numbers at the level of billions(for example, Blinding Lights by The Weeknd, or As it Was by Harry Styles) as well as niche artists that pride themselves in a small but loyal fanbase(for example independent musician Loris S. Sarid). These mega-blockbusters are analogous to billionaires in the economy, where a fractionally small percent of the population can accumulate more than half of the wealth. In terms of volume, the middle items take up more in the interaction space than tail-end items, although still not comparable to head items. These item groups are shown at the top of each plot, and they divide the items based on popularity into the most popular items, items with medium popularity, and items with low popularity.[7] It is also worth noting that there are cases where 100% of items recommended to users are already popular items. This results in the less popular but still desirable items being completely wiped away from the vision of users. These plots show how popularity bias affects various items and creates a "rich get richer" effect.

2.4 Candidate Models

To alleviate the popularity bias problem, several attempts have been made in two directions: improving the core recommendation algorithms, or applying post-processing on the outcomes of some existing model[14, 15]. Since accuracy is still the main concern of MRSs, the latter is the most promising direction so far. Instead of discussing too much about basic RS algorithms, this section only introduces these state-of-art model-based post-processing algorithms used to decline the popularity bias caused by traditional RS models. Fairness-Aware Regularization(FARG)[16] is one strong technique that penalizes the type of recommendations containing less than k groups of items(k is a user-defined parameter i.e. the tolerance of the user.). However, FARG only attempts to decline the concentration of popular items, which leads to the emergence of the following two algorithms. Discrepancy Minimization (DM)[17] uses Gini index and Aggregate Diversity (Agg-Div) as evaluation metrics to improve the amount of individual-specified recommended items. Personalized Long-tail Promotion XQ(PLPXQ)[18], differing from DM, only concerns the balance between popular items and items in the long-tail zone in recommendation lists without considering user propensity. Though the above algorithms have been broadly used in a multitude fields, their item-based approaches have the innate defect of overlooking penalization on a micro level. We will mainly focus on how to develop a user-specified post-processing algorithm to reduce the popularity bias caused by the above methods.

3 Dataset

For the main purpose of our studies, we used Spotify's main dataset for the RecSys Challenge 2018, which contains 1 million playlists with playlist titles, track listings, and other metadata. [19]The dataset is known as the Million Playlist Dataset (MPD). It is worth mentioning that unlike most traditional recommendation datasets such as Movielens, the MPD dataset does not contain a holistic use interaction history. Instead, the main task of this dataset is playlist continuation, which will be discussed in detail in the coming sections. The main data structure of this dataset includes two aspects: Playlist data and Track data. On the playlist side, information such as: Title; Playlist Id; Track name; Description: No. of tracks; No. of artists; No. of albums; No. of followers; No. of edits; List of tracks; Playlist duration; is given. One the track side, information such as: Track Id; Artist name; Artist Id; Album name; Album Id; Track duration; Position (in list); Modification date is given. For details see Table 1.

Table 1: Meta data features for playlists and tracks

Playlist		Track	
Title	Playlist Id	Track name	Track Id
Description	No. of tracks	Artist name	Artist Id
No. of albums	List of tracks	Album name	Album Id
Playlist duration		Track duration	Position (in list)

4 Solution

The workflow of our algorithm can be divided into two stages. First, we adopted and implemented the hybrid recommender framework from one of the competitor groups of the RecSys Challenge 2018 organized by Spotify [20] and used the results as our baseline. Then, we applied the debiasing methods including user-specified factor and cost function optimization to mitigate the popularity bias problem.

4.1 Hybrid Recommender

The original framework we adopted focused on the challenge of automatic playlist continuation. By utilizing playlists that contain several tracks (seed tracks) and metadata information of the tracks including artist, album, and the position of a track in the playlist, the algorithm aims to prolong playlists by adding appropriate songs. Considering the music playlist continuation task is an ideal starting point for us to address the popularity bias problem as it can be used as the baseline for us to further investigate, we selected and implemented part of the framework of one hybrid recommendation system which has three basic approaches: Item-based Collaborative Filtering (ITEM-CF), Session-based Nearest Neighbors (S-KNN), and Alternating Least Squares Matrix Factorization (ALS-MF).(see Figure 2)

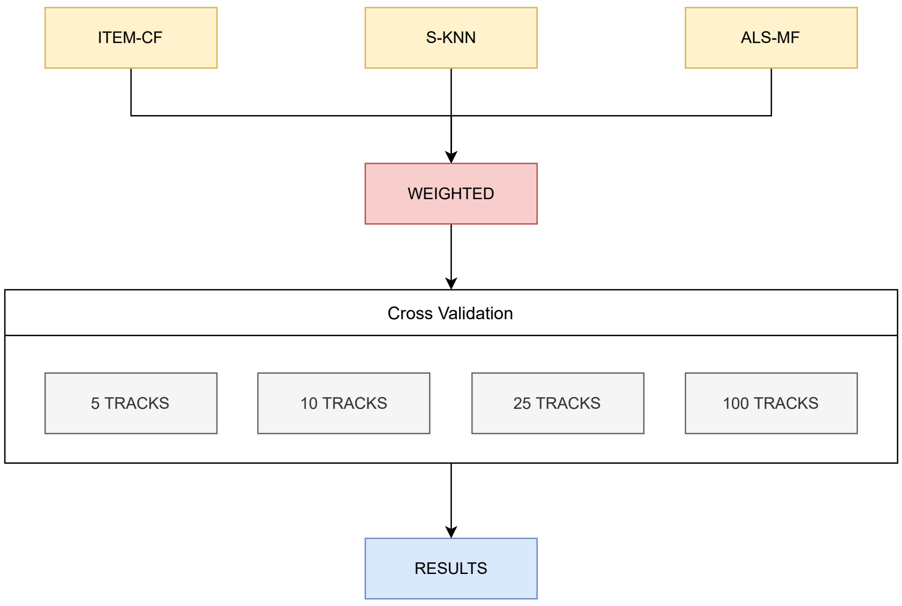

Figure 2: Architecture of Hybrid Recommender

Item-based Collaborative Filtering (ITEM-CF) The item-based collaborative filtering of our hybrid recommender supplements the current playlist by calculating the similarity between the given tracks in it and other tracks. Here, each track is represented by a dummy variable using one-hot encoding, i.e. the entry of a playlist is 1 if the playlist contains that track, and 0 otherwise. Then, the cosine-based similarity calculation was adopted to calculate the similarity between tracks i, j with the following formula:

$$sim(i,j) = \cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{\| \overrightarrow{i} \|_2 \times \| \overrightarrow{j} \|_2}.$$

To reduce the popularity bias brought by the ITEM-CF, the inverse document frequency (IDF) of the corresponding track is used as its weight factor. The IDF of a specific track t is calculated as $idf(t) = \log_{10}(\frac{|P|}{|P_t|})$, where P denotes the set of whole training playlists, and P_t denotes the set of playlists that contains the track t. Thus, the more frequent a track is, the lower its score. In this way, the less popular tracks will have more weights, and the impact of popularity bias is reduced.

Session-based Nearest Neighbors (S-KNN) Another method that is included in the framework is the session-base nearest neighbors, for its favorable results in music recommendation[21]. The S-KNN method takes a playlist as an entirety and finds the most similar playlists from the training data. Then, the recommendation playlist is generated from the top-ranked tracks from the neighbors. To determine the similarity of a given playlist, we represent a playlist as a vector and the entity of each track is 1 if it's in the target playlist and 0 otherwise. In this way, we select the k most similar playlists and denoted the set as K_q. Thus, the ranking for a track t of a given playlist p is calculated as the following formula:

$$rank_{S-KNN}(t,p) = \sum_{q \in K_q} sim(p,q) \cdot \mathbb{1}_q(t) \cdot idf(t),$$

where the $\mathbb{1}_q(t)$ is the indicator function which gives 1 if the neighboring playlist q has the track t, and 0 otherwise. Meanwhile, the inverse document frequency $idf(t)$ is also adopted as weights to strengthen the unpopular tracks.

Alternating Least Squares Matrix Factorization (ALS-MF) In the hybrid recommender, the implicit feedback Matrix Factorization algorithm is also adopted to solve the cold start problem since we need to complement the playlist given only a few seed tracks. We suppose that each playlist is represented by a different user, thus the sparse user-item rating matrix is constructed with the playlists as rows, tracks as columns, and each entry is binary.

Similarly, IDF weights are introduced to emphasize the less popular tracks. As a result, the rank

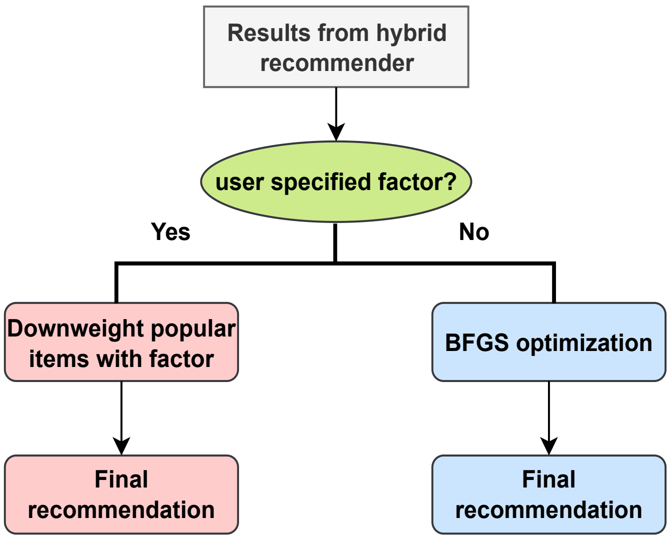

Figure 3: User-specified De-biasing Reranker

of a given track t is calculated by the following formula:

$$rank_{ALS-MF}(t,p) = TL_t \cdot (\frac{1}{\sum_{x \in p} idf(x)} \sum_{x \in p} idf(x) \cdot TL_x)$$

where TL_t is the embeddings of the track t and the latter part is the weighted arithmetic mean of *idf* of all tracks from the playlist p, with the corresponding TL as weight.

Hybridization we combined all three mentioned approaches with specific weights using cross-validation, to generate the baseline results which will be explained in the next section.

4.2 User-specified Reranker

To further mitigate the popular bias problem to the extent of user-specified factors. An adequate amount of methods have been carefully examined with different metrics and two of them survived(see Figure 3).

1. **Downweight popular items according to users' preference**: Confidence scores are calculated back in the hybrid recommender stage. Though a set of de-bias strategies are already applied in the hybrid model, some generated playlists still suffer from popularity bias. Hence, we downweight popular items using the formula: $\overrightarrow{T}_{conf} = \overrightarrow{T}_{conf} - \overrightarrow{pop} \times \overrightarrow{w}_{user}$ where $\times$ stands for element-wise multiplication and $\overrightarrow{w}_{user}$ is a weight vector defined by users.

2. **Broyden–Fletcher–Goldfarb–Shanno(BFGS) optimization based on modified cost**

function: If user instructions are missing, we also provide a BFGS optimization method equipped with cost function F(U) defined as follows:

$$F(U) = \sum_{u \in U} (\sum_{t \in u} pop(t))^2 + \sum_{u,v \in U} (pop(u) - pop(v))^2$$

where the last part is the penalty term to make sure the popularity scores of users do not differ too much. See Algorithm 1 for the pseudocode.

Algorithm 1 BFGS optimization

```
procedure BFGS OPTIMIZATION($F, U_0, tol$)
    $H_k \leftarrow I_0$
    $U_k \leftarrow U_0$
    while True do
        $\mathbf{p}_k \leftarrow -H_k \nabla F(U_k)$
        $\alpha_k \leftarrow \arg \mathbf{min} F(U_k + \alpha \mathbf{p}_k)$
        $s_k \leftarrow \alpha_k \mathbf{p}_k$
        $U_{k+1} \leftarrow U_k + s_k$
        $y_k \leftarrow \nabla F(U_{k+1}) - \nabla F(U_k)$
        $U_k \leftarrow U_{k+1}$
        $H_k \leftarrow H_k + \frac{(s_k^T y_k + y_K^T H_k y_k)(s_k s_k^T)}{(s_k^T y_k)^2} - \frac{H_k y_k s_k^T + s_k y_k^T H_k}{s_k^T y_k}$
        if $\|\nabla F(U_k)\| \leq tol$ then
            Break
        end if
    end while
    return $U_k$ , $F(U_k)$
end procedure
```

5 Results and Discussion

5.1 Evaluation metrics

1. **Precision@N**: This metric measures the precision of our model predictions. If N tracks are recommended to a user, the precision@N is calculated as the percentage of recommended tracks that are also top N tracks in other playlists.

$$Precision@N = \frac{|G \cap R_{1:|G|}|}{|G|}$$

2. **Serendipity**: Serendipity is a criterion for making appealing and useful recommendations [22]. Let $E(I, H) = \frac{1}{|I|} \sum_{i \in I} \sum_{h \in H} \cos(i, h)$ represents the unexpectedness where I is the

item matrix and H is the history matrix, then the formula is generally defined as:

$$Serendipity = \frac{1}{|U||I|}\sum_{u\in U}\overrightarrow{E}\cdot \boldsymbol{relevance}$$

3. **Intra List Diversity(ILD)**: Intra List Diversity is defined as the average pairwise cosine distance among tracks. We use this metric to evaluate how diverse different tracks within a playlist are.

$$ILD(R) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(1 - Similarity(c_i, c_j))}{n(n-1)/2}$$

4. **Popularity mean**: Mean of popularity scores. For each track, the Spotify API provides a popularity score on a scale of 0 to 100. We use the mean of this metric as an indicator of how popular the songs are in a playlist overall.

5. **Popularity standard deviation**: Standard deviation of popularity scores. If the standard deviation of the popularity is high, that means the playlist is usually diversified to the extent that mega-hits and niche songs coexist. If the converse is true, that indicates a consistent style of recommendation – if the popularity score mean is high, we have a playlist like the "Billboard top 100s"; if the popularity score is low, most of the tracks will be positioned in the long tail.

5.2 Result tables

The result from the base model–hybrid recommender is a track_id list sorted by the confidence score of each track. And we sample tracks from this new list for each user as our recommended playlists(see Table 2). Evaluations are based on the recommended playlists(see Table 3). High serendipity in Table 4 proves that our model indeed makes an appealing recommendation and other metrics like precision@N and popularity std demonstrate the stability of our base model recommendation.

Table 2: Result from hybrid recommender

track_id	741	3034	5835	1154	1571
confidence	1	0.599999	0.5	0.499986	0.499982

Table 3: Evaluations of result from base model

Precision@N	Serendipity	Diversity(ILD)	Pop mean	Pop std
0.1890	0.8200	483.7500	16.5963	17.7883

A manifest 30.44 % decrease of pop std can be directly calculated from results in Table 4. This drop means the new recommended list suffers less from popularity bias. Meanwhile, the drop in precision is minor, only 4.23%. These evaluations and comparisons demonstrate that our user-specified reranker can significantly exclude outliers of popularity scores and satisfy the user's preference.

Table 4: User-specified reweight

	Precision@N	Serendipity	Diversity(ILD)	Pop mean	Pop std
Baseline	0.1890	0.8200	483.7500	16.5963	17.7883
User-specified	0.1810	0.8270	467.2500	16.5963	12.3728

Through simple calculation of data form Table 5, we can see significant drop on pop mean(42.46%) and pop std(24.20%). Even though we get a slightly higher reduction on precision, it still lies in the acceptable scale(7.64%). These evaluations and comparisons demonstrate that our BFGS optimization algorithm can significantly exclude outliers of popularity scores, reduce overall popularity mean and still acquires a acceptable precision.

Table 5: BFGS optimization results

	Precision@N	Serendipity	Diversity(ILD)	Pop mean	Pop std
Baseline	0.1890	0.8200	483.7500	16.5963	17.7883
BFGS optimization	0.1745	0.8000	490.0355	9.5784	13.4833

6 Discussion

Table 6: Two shuffle methods

	Precision@N	Serendipity	Diversity(ILD)	Pop mean	Pop std
Baseline	0.1890	0.8200	483.7500	16.5963	17.7883
User-specified	0.1810 ↓	0.8270 ↑	67.2500 ↓	16.5963	12.3728 ↓
BFGS-optimization	0.1745 ↓	0.8000 ↓	490.0355 ↑	9.5784 ↓	13.4833 ↓

Table 6 summarizes our main findings from the two models. Relative changes in each metric compared to the baseline model has been indicated by the up/down arrows next to each metric. The approach in this paper has a major limitation in that playlist continuation tasks have notable differences from traditional user-based recommendation tasks. First of all, the magnitude of metadata available in the traditional case includes more information than a few tracks in a given set. Our task is simplified in the sense that user listening history is not directly taken into account. However, this simplification could be beneficial as it gives more flexibility to the users. Given the fleeting nature of musical preferences, the playlist continuation sheds light on one potential way that streaming platforms can give partial autonomy to the users: in fact, this approach is already seen on Spotify as the "advanced" playlist function. To transfer the playlist continuation task to traditional recommenders, future work can be done in treating the entire user listening history as a given playlist and appending on its basis. Functions to partition the history and form different combinations of sub-playlists may also be explored.

It is important to note that calibrating the recommendations based on popularity does not guarantee that the recommended items will be matched to the users' overall content preferences. For example, a documentary film and a film noir film may both be unpopular, but a user may be interested in one while disliking the other. Having said that, the popularity calibration method proposed in this paper does not aim to calibrate the recommendation lists in terms of content, but rather on the popularity of the items. This popularity calibration is primarily intended to overcome overall popularity bias, as well as to provide a more diverse list of popular recommendations to the user. To address this issue, popularity calibration can be combined with genre calibration. This, however, was not the focus of our work in this paper.

There are some problems that cannot be resolved perfectly by our model. For instance, a subset of tracks are excluded in our shuffle stage to make the calculation stale, which makes it impossible for our system to recommend some songs. Given the fact our work only focus on the post-processing stage and did not tackle the baseline of recommender models, problems like this can only be resolved by changing baseline model accordingly. If one is going to change the baseline model and expande the shuffle pool, a promising baseline model will be Reinforcement learning recommender systems(RLRSs) with special award function mentioned in this paper like novelty, ILD or serendipity on the agent side.

7 Personal Contributions

Dennis (Shiyi) Hu wrote sections 1, 2.1-2.3, 3, 6, and 7 of this paper. He also connected to the Spotify API in the first experimentation periods and ran scripts to retract metadata. In the literature review process, he focused on aggregating popularity bias research on music recommenders. In the later phases of this draft, he has written the preface and abstract, as well as did final rewording work before submission.

8 Conclusion

Throughout this project, we first successfully applied the hybrid recommender system to solve the music playlist continuation problem and meanwhile generated the baseline results to further implement the debias methods for mitigating the popularity bias problem. In the second phase of our project, we implemented two separate approaches: user-specified reranker and BFGS optimization reranker. Whereas user-specified reranker can significantly exclude outliers of popularity scores and satisfy the user's preference, BFGS optimization approach is able to exclude outliers of popularity scores, reduces overall popularity mean and still acquires a acceptable precision.

Our results serve as preliminary evidence that item-based recommendations systems have the room for improvement in terms of user specificity without significant sacrifices in overall precision. This is especially important to streaming platforms since popularity preferences are usually currently estimated from entire user listening histories. Our approach shows that fragments of a users current listening preferences may well serve better in terms of predicting the desired popularity of songs recommended. By giving more signaling power to the users, streaming platforms can potentially align recommendations much better with the fluctuating user preferences.

Furthermore, two main directions for future research stem from our results. It is important to investigate and gather empirical evidence that the general population does demonstrate an inclination toward self-tuned recommendation algorithms. One possible way to do so is to conduct experiments on selected individuals to use pilot recommendation algorithms while comparing self report a satisfaction rate with a platform to a random controlled group. One other direction is to probe into the average level of fluctuation in terms of popularity preference within a given individual. This is a key question to address before anyone can access the value provided by user-tuned recommenders. The study may also involve collecting behavioral data and regular follow-up with a group of randomly controlled individuals. If further research reveals that listeners indeed exhibit a high standard deviation in popularity preferences across time, it will serve as sound evidence that our explorations in popularity-adjusted recommenders are well justified.

References

[1] Z. Taskin and U. Al, “A content-based citation analysis study based on text categorization,” *Scientometrics*, vol. 114, no. 1, pp. 335–357, 2018. [Online]. Available: https://doi.org/10.1007/s11192-017-2560-2

[2] V. C. Tran, D. Hwang, and N. T. Nguyen, “Hashtag recommendation approach based on content and user characteristics,” *Cybern. Syst.*, vol. 49, no. 5-6, pp. 368–383, 2018. [Online]. Available: https://doi.org/10.1080/01969722.2017.1418724

[3] J. Shu, X. Shen, H. Liu, B. Yi, and Z. Zhang, “A content-based recommendation algorithm for learning resources,” *Multim. Syst.*, vol. 24, no. 2, pp. 163–173, 2018. [Online]. Available: https://doi.org/10.1007/s00530-017-0539-8

[4] B. B. Sinha and R. Dhanalakshmi, “Evolution of recommender system over the time,” *Soft Comput.*, vol. 23, no. 23, pp. 12 169–12 188, 2019. [Online]. Available: https://doi.org/10.1007/s00500-019-04143-8

[5] M. Schedl, H. Zamani, C. Chen, Y. Deldjoo, and M. Elahi, “Current challenges and visions in music recommender systems research,” *Int. J. Multim. Inf. Retr.*, vol. 7, no. 2, pp. 95–116, 2018. [Online]. Available: https://doi.org/10.1007/s13735-018-0154-2

[6] M. Naghiaei, H. A. Rahmani, and M. Dehghan, “The unfairness of popularity bias in book recommendation,” in *Advances in Bias and Fairness in Information Retrieval - Third International Workshop, BIAS 2022, Stavanger, Norway, April 10, 2022, Revised Selected Papers*, 2022, pp. 69–81. [Online]. Available: https://doi.org/10.1007/978-3-031-09316-6_7

[7] H. Abdollahpouri, R. Burke, and B. Mobasher, “Managing popularity bias in recommender systems with personalized re-ranking,” in *The thirty-second international flairs conference*, 2019.

[8] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, “Bias and debias in recommender system: A survey and future directions,” *arXiv preprint arXiv:2010.03240*, 2020.

[9] H. Steck, “Item popularity and recommendation accuracy,” in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 125–132. [Online]. Available: https://doi.org/10.1145/2043932.2043957

[10] W. Wang, F. Feng, X. He, X. Wang, and T. Chua, “Deconfounded recommendation for alleviating bias amplification,” *CoRR*, vol. abs/2105.10648, 2021. [Online]. Available: https://arxiv.org/abs/2105.10648

[11] G. Takács and D. Tikk, “Alternating least squares for personalized ranking,” in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 83–90. [Online]. Available: https://doi.org/10.1145/2365952.2365972

[12] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW ’01. New York, NY, USA: Association for Computing Machinery, 2001, p. 285–295. [Online]. Available: https://doi.org/10.1145/371920.372071

[14] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, and E. C. Malthouse, "User-centered evaluation of popularity bias in recommender systems," *CoRR*, vol. abs/2103.06364, 2021. [Online]. Available: https://arxiv.org/abs/2103.06364

[15] F. Lu and N. Tintarev, "A diversity adjusting strategy with personality for music recommendation," in *Proceedings of the 5th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2018, co-located with ACM Conference on Recommender Systems (RecSys 2018), Vancouver, Canada, October 7, 2018*, 2018, pp. 7–14. [Online]. Available: http://ceur-ws.org/Vol-2225/paper2.pdf

[16] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, P. Cremonesi, F. Ricci, S. Berkovsky, and A. Tuzhilin, Eds. ACM, 2017, pp. 42–46. [Online]. Available: https://doi.org/10.1145/3109859.3109912

[17] A. Antikacioglu and R. Ravi, "Post processing recommender systems for diversity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 707–716. [Online]. Available: https://doi.org/10.1145/3097983.3098173

[18] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," in *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, Sarasota, Florida, USA, May 19-22 2019*, R. Barták and K. W. Brawner, Eds. AAAI Press, 2019, pp. 413–418. [Online]. Available: https://aaai.org/ocs/index.php/FLAIRS/FLAIRS19/paper/view/18199

[19] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, "Recsys challenge 2018: Automatic music playlist continuation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 527–528. [Online]. Available: https://doi.org/10.1145/3240323.3240342

[20] M. Ludewig, I. Kamehkhosh, N. Landia, and D. Jannach, "Effective nearest-neighbor music recommendations," in *Proceedings of the ACM Recommender Systems Challenge 2018*, 2018, pp. 1–6.

[21] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 306–310.

[22] R. J. Ziarani and R. Ravanmehr, "Serendipity in recommender systems: a systematic literature review," *Journal of Computer Science and Technology*, vol. 36, no. 2, pp. 375–396, 2021.