Addressing Popularity Bias in Citizen Science

Amit Sultan
Avi Segal
Guy Shani
Ben-Gurion University of the Negev
Beer-Sheva, Israel

Kobi Gal
Ben-Gurion University
Beer-Sheva, Israel
University of Edinburgh
Edinburgh, U.K.

ABSTRACT

Citizen science projects rely on volunteers to contribute time and effort to solve scientific problems, but the majority of citizen science users typically contribute to only one or two projects. Recommender systems have been recently used in citizen science to motivate people to contribute to additional projects. However, these systems often recommend the more popular projects at the expense of less popular projects which need people's contributions more critically. In this work we develop a post processing approach for enhancing "long-tail" item recommendation that can be applied to any recommendation system. We propose a novel re-ranking model, based on the lift measure used in machine learning, which considers item co-occurrence as well as popularity for enhancing long tail recommendations. We demonstrate the efficacy of our approach in the citizen science domain on two data sets and three state of the art recommendation algorithms, comparing hit rate before and after applying lift boosting. Additionally, we compare our approach to two predetermined re-ranking baselines. Results show that our proposed approach significantly improves performance for tail item recommendation without a substantial loss in head item and overall item recommendation performance. Our approach is general and can be naturally applied to existing recommendation systems in citizen science that personalize project suggestions to users, potentially leading to an increase in efficiency and performance.

CCS CONCEPTS

• **Applied computing** → **Computers in other domains**.

KEYWORDS

citizen science, recommendation systems, popularity bias

ACM Reference Format:
Amit Sultan, Avi Segal, Guy Shani, and Kobi Gal. 2022. Addressing Popularity Bias in Citizen Science. In *Conference on Information Technology for Social Good (GoodIT'22), September 7–9, 2022, Limassol, Cyprus.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3524458.3547229

GoodIT'22, September 7–9, 2022, Limassol, Cyprus

ACM ISBN 978-1-4503-9284-6/22/09…$15.00
https://doi.org/10.1145/3524458.3547229

1 INTRODUCTION

Citizen science engages people in scientific research by completing various tasks such as categorizing, transcribing and classifying scientific data [5–7]. For example, Zooniverse (Zooniverse.org), which is part of the empirical focus of this paper, is the world's largest and most popular platform for citizen science, with more than a million users around the world and hundreds of unique projects published by professional researchers [23]. In all Zooniverse projects the volunteers gather, identify, classify, mark, and label data, which is subsequently aggregated and analyzed in order to reach scientific conclusions.

The success of citizen science platforms critically depends on users' sustained contributions to the different projects on the platform [23]. However, the number of citizen science projects people contribute to typically follows a power law distribution, by which the majority of people contribute only to one or two projects. This prevents many viable projects from collecting sufficient contributions and fulfilling their scientific potential.

Past work has demonstrated the benefit of employing machine learning to personalize project recommendations to users in several citizen science portals [4]. However, recommendation systems commonly prefer to recommend popular projects, which are better known and receive more contributions, at the expense of niche projects that are popular only among smaller groups of volunteers. This effect, known as "popularity bias" [2] reduces the scientific value of the citizen science portal and prevents volunteers from discovering new projects that may lead them to engage and contribute more on the portal. While popularity bias has been documented in e-commerce and movie domains, it is more acute in volunteer-based crowd-sourcing settings such as as citizen science. To illustrate, consider an exiting recommendation solution implemented for the SciStarter citizen science platform (scistarter.org). We found that over 90% of the projects recommended to SciStarter users during a 6 months period consisted of 20% of the most popular projects.

This paper addresses popularity bias by re-ranking the output of existing recommendation algorithms to favour long tailed projects in a controlled way, using the "lift" measure from machine learning [16]. For a given project and user, the method considers three factors: First, the popularity of the project. Second, the potential of the project for the user, which is measured by the number of users who participated in the project and exhibited similar participation patters to the user. Third, the relevance score that is outputted by the existing recommendation algorithm. It is designed to "lift" the ranking of non-popular projects with high potential to the user.

We evaluate the method by comparing its performance when using state-of-the-art recommendation algorithms in the Zooniverse and SciStarter citizen science platforms. We compared the hit rate

performance of the lift based approach to the original recommendation algorithms, as well as a baseline approach for re-ranking projects that solely considers the relevance score. In both of these platforms, we show that the lift based re-ranking approach significantly improved the performance of long-tail projects without hindering the overall performance for all projects.

The contribution of this work is in suggesting a domain-independent method for addressing popularity bias in a socially relevant domain. It can be applied to any recommendation algorithm that ranks projects by score and has great potential for improving the scientific value of citizen science.

2 RELATED WORK

Our work builds on two separate research areas: recommendation systems for crowd sourcing and citizen science, and algorithms for addressing popularity bias in recommendation systems.

Recommendation systems have been used in the past to recommend tasks to workers in crowd sourcing platforms. Lin et al. [14] developed an implicit feedback based approach for task recommendation in the UHRS system [1], Microsoft's crowd sourcing platform. They showed that reasoning about both positive and negative implicit feedback improved the quality of task recommendations in offline experiments. Li et al. [12] proposed a social influence-based recommendation engine to recommend suitable tasks for users in crowdsourcing. They demonstrated the efficacy of their approach using offline testing on a dataset from JointForce [2], a popular software crowdsourcing platform in China. Finally, Safran and Che [19] proposed new representation schemes to improve task recommendations to crowd sourcing workers. They used explicit features and categories unique to the domain, and compared their algorithms to known baselines on a public dataset. In the citizen science domain, Ben Zaken et al. [4] evaluated the performance of different recommendation algorithms in offline settings for the Scistarter platform [3]. They then conducted a large scale online experiment and demonstrated the efficacy of recommendations in increasing users engagement for this citizen science platform. We note that these past approaches did not consider popularity bias in their work.

It is well-known that recommender systems are susceptible to popularity bias and tend to favor more popular items [1, 10]. Past studies have developed multiple bias-controlling algorithms to provide long-tail items a greater probability of success. We name main approaches in this field. Regularization-based approaches have been utilized in recommendation systems to alleviate popularity bias. For example, Abdollahpouri et al. [2] focused on adding a regularization component to matrix factorization techniques and demonstrated improvement in overall performance and an increase in the number of suggestions for long tail items. Although this method improves long tail performance, it is not generalizable and must be manually developed and tuned for different algorithms. Our solution provides a generalized approach that can be added to any given recommendation system.

[1]https://prod.uhrs.playmsn.com/uhrs/
[2]https://www.chinasofti.com/en/JointForce/index.htm
[3]https://scistarter.org/

Meta-learners based approaches use transfer learning to enhance long tail item recommendations. in Zhang et al. [28], transfer learning is used to transfer rich information from head items to tail items which receive less user feedback. The method is shown to improve the suggestions for tail items in terms of hit rate and NDCG. This approach requires item level features, which is not required by our approach. Additionally, it uses a multilayer perceptron (MLP) as a base learner, while our approach enables flexibility across base learners.

Finally, several re-ranking based approaches have been developed for alleviating popularity bias in recommendation systems. Abdollahpouri et al. [3] adapted a probabilistic web search results diversification approach called xQuAD [20] for the recommendation domain. They have tested their approach on the Epinion and Movielens datasets. Miyamoto et al. [17] defined a new "Appearance Frequency Metric" and used it to balance between long tail items and popular items in a re-ranking process on the Movielens and Epinion datasets. Liu et al. [15] developed a multi-objective re-ranking approach which considers both performance and diversity. They used offline analysis to demonstrate the efficacy of their approach in improving long tail item performance for the Movielens dataset. Our approach is the first re-ranking based approach that utilizes the lift metric, and the first popularity bias handling algorithm implemented for the citizen science domain.

3 METHODOLOGY

We now present our approach for enhancing recommendations of the less popular tail projects, using statistical measures. We first define some required standard notation. Let U be the set of users, and P the set of projects. Let P_u be the set of projects that user u has participated at. Let U_p be the set of users that participated at project p.

In data mining and associative rule learning, the lift measure [16] is the rise in probability of observing event j when it occurs together with event i, over the probability of observing j and event i independently:

$$Lift(j,i) = \frac{p(j \mid i)}{p(j)} = \frac{p(j \cap i)}{p(j) \cdot p(i)} \tag{1}$$

With respect to the citizen science domain, the probability $p(i)$ measures the popularity of project i, that is, the amount of user that participated at i. Hence, in our case:

$$Lift(p_i,p_j) = \frac{\frac{|U_{p_i} \cap U_{p_j}|}{|U|}}{\frac{|U_{p_i}|}{|U|} \cdot \frac{|U_{p_j}|}{|U|}} = \frac{|U_{p_i} \cap U_{p_j}| \cdot |U|}{|U_{p_i}| \cdot |U_{p_j}|} \propto \frac{|U_{p_i} \cap U_{p_j}|}{|U_{p_i}| \cdot |U_{p_j}|} \tag{2}$$

$Lift(p_j,p_i)$ measures the potential that a user will participate at a new project p_j given the user has already participated at project p_i in the past.

We further define the lift value of a recommended project p_j to a target user u, as the median lift between p_j and all projects that u has participated at:

$$Lift(p_j,u) = median_{p_i \in P_u} Lift(p_j,p_i) \tag{3}$$

The lift of a recommended project p_j for target user u will be higher if more users participated at both p_j and other projects which user

u has participated at in the past, and if the popularity of project p_j in the overall population is low.

Our working hypothesis is that projects which have higher lift values with past projects that the user has already participated at present stronger potential for the user, while controlling for the project popularity. Our approach modifies the rankings of recommended projects according to their lift measures in order to improve performance of long tail projects while minimizing the decrease in performance on head projects.

Clearly, one can choose other aggregation metrics than the median, such as the mean, or the maximal value. Nonetheless, our empirical evaluation showed that in the citizen science domain, the median produced the best tradeoff between accuracy and popularity.

Our approach uses the combination of lift and relevance to rank the recommendation list. We take as input a base recommendation algorithm a that, for a user u and a project p_j computes $\hat{r}_a(u, p_j)$, the relevance of the project p_j to the target user u. To compute a list of recommended projects for u, we order the list of projects that u has not yet participated in by decreasing order of $Lift(u, p_j) \times \hat{r}_a(u, p_j)$. This method promotes projects that are both relevant for user u, and have a high lift value for them. We term this re-ranking method "lift boosting".

3.1 Base Recommendation Algorithms

We use several algorithms as base recommenders in our experiments. All of the approaches are based on collaborative filtering, where individuals with similar past behavior are also predicted to behave similarly in the future[21, 25].

For example, in the context of the Zooniverse citizen science platform, suppose both user i and user j participated in astronomy projects Galaxy Zoo, Planet Hunters and Dark Energy Explorers. Additionally, suppose that user i also participated at the project Aurora Zoo, another astronomy related project. Therefore, Aurora Zoo may be a suitable new project recommendation for user j both by context and by users' shared history.

The baseline recommendation algorithms are as follows:

User-based Collaborative Filtering (UBCF). A strategy for recommending projects to a certain user can thus be accomplished by selecting the K nearest neighbors (users) for that user. To determine which users are most comparable to a target user, a similarity metric is needed. A naïve method for computing user similarity is the cosine similarity metric for binary item consumption in which a user interacted with or did not interact with a project [24].

$$SIM(u_1, u_2) = \frac{|P_{u_1} \cap P_{u_2}|}{|P_{u_1}| \cdot |P_{u_2}|} \tag{4}$$

Similar users will share many projects, therefore the intersection and union of project sets will be high, resulting in a greater similarity value compared to users who share fewer projects. We note that the similarity definition is proportional to the lift between two users.

Based on the similarity metric, the User-based Collaborative Filtering algorithm [25] determines the rating for each new project p_j for user u. Typically, for each user u, we use a relatively small set of users $N(u)$ with relatively high similarity to u. $N(u)$ is called the set of neighbors for user u. We can now compute a relevance score:

$$\hat{r}_{UBCF}(u, p_j) = \frac{\sum_{u_k \in N(u) \cap U_{p_j}} SIM(u, u_k)}{\sum_{u_k \in N(u)} SIM(u, u_k)} \tag{5}$$

That is, the predicted relevance score for project p_j of user u is obtained by summing the similarities of users in the neighborhood of u that participated in p_j, normalized by the total sum of neighbor similarity scores.

Matrix factorization. An alternative method to user similarity based approaches is to represent user-project relevance in a latent space that characterizes both users and projects. The latent space approach enables a condensed representation of the usually sparse participation matrix while uncovering latent commonalities in the user and project spaces.

The matrix factorization approach seeks to construct such a latent space that characterizes users and projects through vectors of a predefined dimension k [8, 9]. The original participation matrix $M_{|U|x|P|}$, is a matrix representing user and project participation where each row represents a user and each column represents a project. Then, cell value $r_{u,i}$ is 1 if user u participated in project p_i and 0 otherwise.

Matrix factorization algorithms work by decomposing this user-item participation matrix into the product of two lower dimensionality rectangular matrices, one for the users and one for the projects. Given k, the size of the latent space for both users and projects, the two decomposed matrices $P_{U,k}$ and $Q_{I,k}$ are obtained by minimizing the difference between matrix $\hat{M} = PQ^T$ and the original M matrix.

The result is a set of vectors $\vec{u}$ for a user u and $\vec{p}$ for a project p, and the relevance score is computed using:

$$\hat{r}_{MF} = \vec{u} \cdot \vec{p} = \sum_{i=1}^{k} \vec{u}_i \cdot \vec{p}_i \tag{6}$$

Specifically, we use the BPR algorithm [18] which augments the matrix factorization approach with a Bayesian methodology for personalizing a ranking over projects for each user. The method incorporates a normal prior distribution over model parameters and derives the posterior distribution using Stochastic gradient decent.

Variational AutoEncoder. AutoEncoders (AE) are a neural network-based technique for dimensionality reduction [26]. These networks consist of an encoder architecture that encodes the original data to a predifined smaller dimension, the latent space, and a decoder architecture that decodes the encoded values back to the original dimension. By training the network we learn a latent representation of the data that can best be used to reconstruct the input data. Variational AutoEncoders [13] share the same concept of an Autoencoder but instead of encoding and decoding to a certain latent space the input is encoded as a distribution over the latent space.

When training the autoencoder for our task, we use as input a vector of a given user u participation's — a row in the $M_{|U|x|P|}$ matrix described above — containing 1 for each project that the user participated in, and 0 otherwise. The input X is then encoded as a probability distribution over the latent space which is then sampled for the decoding and the re-construction phase of $\hat{X}$.

To provide a relevance score for a user u and a project p_j, we take the latent vector associated with u, and run it through the encoder. Then, we take the value at the output entry associated with p_j and use this as $\hat{r}_{VAE}(u, p_j)$.

4 EXPERIMENTS

We now report on the evaluation of our lift boosting approach. Specifically, we compare the performance of the 3 base recommendation algorithms defined earlier when applied to one citizen science dataset, and then proceed to test the best performing algorithm (as measured by performance on tail projects) on a second citizen science dataset.

4.1 Data Sets

Our evaluation uses data collected from two different citizen science platforms. The first dataset includes click-stream data collected from the Zooniverse platform [29]. Each instance is a participation of a user in a project which includes a project ID and a timestamp. The data was collected between January 1,2021 to December 8, 2021. The second dataset includes click-stream data collected from the Scistarter [22] platform. The interaction matrices are sparse as the majority of users often participate in small numbers of available projects [4]. Table 1 describes the amount of users, projects, and user participation in both datasets.

Figure 1 illustrates the split between the 20% most popular projects (head) and the 80% projects left (tail) in the Zooniverse dataset. Here, the X axis is the popularity ranking and the Y axis is the number of user participations in each project. The red line represents the split between head and tail projects. Consistent with the Pareto principle, we can see in the figure that 20% percent of the most popular projects account for 78.33% percent of the platform's total activity.

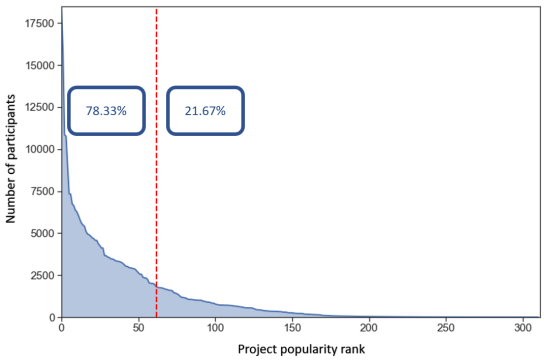

Figure 1: Zooniverse head and tail projects split

4.2 Splitting the data

Our algorithms require a train-test split that uses some project participations of every user as the test set. We hence select 20% of the project participations in the dataset and use them as our test set. The rest of the project participations are used for training. We used the stratified train-test split supported by the LightFM package [11] which randomly divides participations between training and testing.

4.3 The Hit Rate Metric

We use the hit rate metric for evaluating algorithms performance before and after applying our lift based boosting. The hit rate for a particular user $u \in U$ is either 1 if a relevant project in the test set was discovered in the recommended list of projects, or 0 otherwise. It is commonly used in evaluating recommendation systems[4, 27]. The top-K hit rate metric measures the hit rate for the projects in the recommended list that are ranked in the top K positions, where K is a parameter. The hit-rate at K for the population is computed as the mean value over all users' hit rate at K values.

$$HR@K = \frac{|U^K_{hit}|}{|U|} \tag{7}$$

We denote $|U^K_{hit}|$ as the number of test set users with at least one correct recommendation in their top K recommendation list, and compute the average hit rate for all test users.

We prefer hit-rate to precision here because most users participate in only a few projects, and hence, almost all users have only a single project in their test set. Thus, the precision for a list of K recommendations is bounded by $1/K$.

4.4 Baselines

We implemented a simple re-ranking baseline method for long tail project recommendations which only consider the relevancy score of the base recommendation algorithm. This approach uses a fixed portion of long tail projects in each recommendation issued by the baseline. Formally, given a portion $q \in [0, 1]$, and K, the amount of recommendations, we take the $\lfloor q \cdot K \rfloor$ projects with the highest relevance score in the head, and $K - \lfloor q \cdot K \rfloor$ projects with the highest relevance score from the tail. We chose two different q portion values: (1) a "fair" model with $q = 0.5$ which ensures each recommendation includes an equal number of head and tail projects and (2) a biased model, with $q = 0.2$ where 80% of the projects in each recommendation list are long tail projects, giving long tail projects very high importance.

5 RESULTS

We implemented the post processing re-ranking method using the combined lift and relevance scores. For BPR, we used the algorithm by [18]. For the Variational Autoencoder, we used the algorithm of Li and She [13]. All experiments were executed on a 3.00GHz machine with 32GB of RAM.

We use the hit rate metric and compare overall performance of the different algorithms, as well as performance for head and tail projects.

We first report our results for $K = 10$ which is a common recommendation threshold in many previous publications [2, 3, 28]. Table 2 presents the hit rate performance on the Zooniverse dataset for top 10 recommendations - before and after applying our lift re-ranking approach. Bold numbers represent statistically significant results when compared to the other approaches for each algorithm (sign test, $p < 0.0001$). For BPR, we add two alternative head-tail

Table 1: Zooniverse and Scistarter dataset properties

	Zooniverse	Scistarter
Users	170,411	13,858
Projects	311	84
Records	366,413	16,964

splits, namely "q=0.5" which ensures that the amount of recommendations from the head and the tail are equal, and "q=0.2", which is biased in favor of less popular projects.

As can be seen from the table, the UCBF, BPR and MVAE lifted versions demonstrate comparable or improved hit rate performance on all item recommendation. All boosted algorithms outperform the non boosted algorithms on tail items hit rate. As can be expected, the results for head items are opposite, in that the lift boosting approach decreases performance for the most popular projects.

Table 2: Recommendation performance of lift-boosted vs. non-boosted approaches (Zooniverse dataset)

Measure	Overall	Head	Tail
	HR@10	HR@10	HR@10
UBCF	0.110	0.090	0.03
UBCF Lift	0.130	0.082	**0.05**
BPR	0.249	0.208	0.062
BPR q=0.2	0.185	0.094	0.115
BPR q=0.5	0.233	0.176	0.088
BPR Lift	0.249	0.182	**0.125**
MVAE	0.501	0.473	0.045
MVAE Lift	0.504	0.47	**0.055**

When comparing the simple baselines to our lift-boosted strategy, we observe two key differences. The first is that our approach achieves better results for tail projects and suffers less for head projects than the baseline which is biased towards unpopular projects ("q=0.2"). The second is that our strategy yields superior outcomes compared to the fair baseline ("q=0.5") for both overall projects and tail projects.

To evaluate the robustness of our approach, we apply it to the SciStarter dataset, comparing it to the UBCF and BPR algorithms (Table 3). On the Scistarter data set, our method achieves the highest hit rate for overall, head and tail projects. We speculate that for this smaller dataset, lift-boosting enabled better diversity on head projects, hence the improved results.

To demonstrate the behavior of our re-ranking method, we present an example of recommended projects computed for one of the users in the Zooniverse dataset, which participated in a relatively large number of projects. Table 4 displays the top-10 recommendations of the BPR algorithm before and after the re-ranking procedure in order to showcase the efficacy of our lift-boosted method. The table presents two sets of recommendations, one before and one after re-ranking. Each column displays the recommendations in descending order and indicates the project's popularity with a H or T for Head and Tail, respectively. Recommendations with a bold font were consumed by the specific user in the test set.

Table 3: Recommendation performance of lift-boosted vs. non-boosted approaches (Scistarter dataset)

Measure	Overall	Head	Tail
	HR@10	HR@10	HR@10
UBCF	0.257	0.150	0.107
UBCF Lift	**0.278**	**0.162**	**0.120**
BPR	0.407	0.352	0.065
BPR q=0.2	0.291	0.176	0.132
BPR q=0.5	0.407	0.356	0.066
BPR Lift	**0.556**	**0.447**	**0.126**

The recommendations prior to the re-ranking procedure contain 90% popular (head) projects, with all correct recommendations being head projects. Comparing this to our lift-boosting strategy, we notice a greater diversity of head and tail projects at the top 10 list, including correct tail projects which were not recommended to the user in the non-boosted approach.

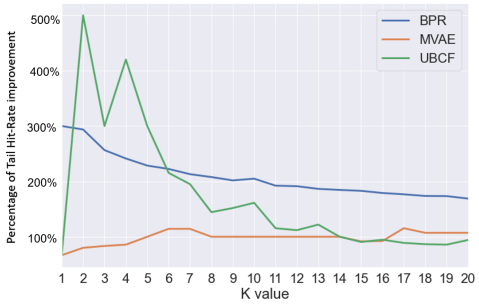

Figure 2: Percentage of Tail Hit-Rate@K improvement over the baseline.

Finally, we inspect the algorithms behaviour over changing k values. Figure 2 shows the percentage of tail hit-rate improvement of using lift at each k value for the 3 tested algorithms. For example, the BPR algorithm shows a 100% increase for tail hit-rate at K=10 when combined with Lift-boosted approach. As seen in the figure, the extent of improvement achieved by lift depends on the base algorithm, especially for smaller k values. Specifically, UBCF is optimal for k smaller than 6, while for larger k values BPR is optimal. These results are statistically significant for all k values (sign test, $p<0.0001$). A possible explanation for the reduction in improvement for larger k values, is that for these values, base algorithms already include more long-tailed items even without considering lift.

Table 4: Example of Top Ranked Recommendations with and without Lift-boosting

Popularity	BPR Recommendations	Popularity	Lift-Boosted BPR Recommendations
H	**Disk Detective**	T	Protect Our Planet From Solar Storms
H	**Radio Galaxy Zoo: LOFAR**	T	**Milkey Way Project**
H	Citizen ASAS-SN	T	**Exoplanet Explorers**
H	Bursts from space	T	Catalina Outer Solar System Survey
H	SuperWASP Variable Stars	H	**Disk Detective**
H	Radio Meteor Zoo	H	Radio Meteor Zoo
H	Galaxy Zoo Mobile	H	**Radio Galaxy Zoo: LOFAR**
H	Active Asteroids	T	Astronomy Rewind
H	Planet Four: Terrains	H	Bursts from space
T	Zwicky Chemical Factory	T	Zwicky's Quirky Transients

Taken together, our results demonstrate the efficacy of our approach in boosting hit rate performance for tail project recommendations with minimal impact on the recommendation performance of the entire project set.

6 CONCLUSION & FUTURE WORK

In this work we have proposed a new approach for addressing popularity bias in project recommendations for citizen science. Our approach augments existing recommendation algorithms by reranking projects in a way that considers the popularity of the project as well as co-occurence with other projects. It "lifts" the ranking of long tailed projects with high potential for a given user. The approach was implemented in three state of the art recommendation algorithms and applied to data collected from two leading citizen science platforms. In empirical experiments, the lift-based reranking approach significantly improved the hit-rate performance of the algorithms for long-tail projects without hindering the overall performance for all projects. In future work, we intend to deploy the approach in the wild and improve the scientific output of long-tailed citizen science projects.

7 ACKNOWLEDGEMENTS

This work was supported in part by the European Union Horizon 2020 WeNet research and innovation program under grant agreement No 823783.

REFERENCES

[1] Himan Abdollahpouri. 2019. Popularity Bias in Ranking and Recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (Honolulu, HI, USA) *(AIES '19)*. Association for Computing Machinery, New York, NY, USA, 529–530. https://doi.org/10.1145/3306618.3314309

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 42–46. https://doi.org/10.1145/3109859.3109912

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. arXiv:1901.07555 [cs.IR]

[4] Daniel Ben Zaken, Kobi Gal, Guy Shani, Avi Segal, and Darlene Cavalier. 2021. Intelligent Recommendations for Citizen Science. 35 (May 2021), 14693–14701. https://ojs.aaai.org/index.php/AAAI/article/view/17726

[5] Rick Bonney, Caren B Cooper, Janis Dickinson, Steve Kelling, Tina Phillips, Kenneth V Rosenberg, and Jennifer Shirk. 2009. Citizen science: a developing tool for expanding science knowledge and scientific literacy. *BioScience* 59, 11 (2009), 977–984.

[6] Dominique Brossard, Bruce Lewenstein, and Rick Bonney. 2005. Scientific knowledge and attitude change: The impact of a citizen science project. *International Journal of Science Education* 27, 9 (2005), 1099–1121.

[7] Cary Funk, Jeffrey Gottfried, and Amy Mitchell. 2017. Science news and information today. *Pew Research Center* (2017).

[8] Yehuda Koren and Robert Bell. 2015. *Advances in Collaborative Filtering*. Springer US, Boston, MA, 77–118. https://doi.org/10.1007/978-1-4899-7637-6_3

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[10] Dominik Kowald and Emanuel Lacic. 2022. Popularity Bias in Collaborative Filtering-Based Multimedia Recommender Systems. *arXiv preprint arXiv:2203.00376* (2022).

[11] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015. (CEUR Workshop Proceedings, Vol. 1448)*, Toine Bogers and Marijn Koolen (Eds.). CEUR-WS.org, 14–21. http://ceur-ws.org/Vol-1448/paper4.pdf

[12] Ning Li, Wenkai Mo, and Beijun Shen. 2016. Task recommendation with developer social network in software crowdsourcing. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 9–16.

[13] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 305–314.

[14] Christopher Lin, Ece Kamar, and Eric Horvitz. 2014. Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[15] Xiangyong Liu, Guojun Wang, and Md Zakirul Alam Bhuiyan. 2022. Re-ranking with multiple objective optimization in recommender system. *Transactions on Emerging Telecommunications Technologies* (2022), e4398.

[16] P.D. McNicholas, T.B. Murphy, and M. O'Regan. 2008. Standardising the lift of an association rule. *Computational Statistics & Data Analysis* 52, 10 (2008), 4712–4721. https://doi.org/10.1016/j.csda.2008.03.013

[17] Seiki Miyamoto, Takumi Zamami, and Hayato Yamana. 2019. Appearance frequency-based ranking method for improving recommendation diversity. In *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*. IEEE, 420–425.

[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[19] Mejdl Safran and Dunren Che. 2018. Efficient learning-based recommendation algorithms for top-n tasks and top-n workers in large-scale crowdsourcing systems. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–46.

[20] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting Query Reformulations for Web Search Result Diversification. In *Proceedings of the 19th International Conference on World Wide Web* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 881–890. https://doi.org/10.1145/1772690.1772780

[21] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. *Collaborative Filtering Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–324. https://doi.org/10.1007/978-3-540-72079-9_9

[22] Scistarter. 2022. *Scistarter platform*. https://www.scistarter.org/

[23] Avi Segal, Ya'akov Gal, Robert J Simpson, Victoria Victoria Homsy, Mark Hartswood, Kevin R Page, and Marina Jirotka. 2015. Improving productivity in citizen science through controlled intervention. In *Proceedings of the 24th International Conference on World Wide Web*. 331–337.

[24] Ramni Harbir Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, and Gaurav Srivastav. 2020. Movie recommendation system using cosine similarity and KNN. *International Journal of Engineering and Advanced Technology* 9, 5 (2020), 556–559.

[25] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 501–508. https://doi.org/10.1145/1148170.1148257

[26] Yasi Wang, Hongxun Yao, and Sicheng Zhao. 2016. Auto-encoder based dimensionality reduction. *Neurocomputing* 184 (2016), 232–242. https://doi.org/10.1016/j.neucom.2015.08.104 RoLoD: Robust Local Descriptors for Computer Vision 2014.

[27] Xiwang Yang, Chao Liang, Miao Zhao, Hongwei Wang, Hao Ding, Yong Liu, Yang Li, and Junlin Zhang. 2017. Collaborative Filtering-Based Recommendation of Online Social Voting. *IEEE Transactions on Computational Social Systems* 4, 1 (2017), 1–13. https://doi.org/10.1109/TCSS.2017.2665122

[28] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H. Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. arXiv:2010.15982 [cs.IR]

[29] Zooniverse. 2022. *Zooniverse platform.* https://www.zooniverse.org/