Defusing Popularity Bias in Recommender Systems for Software Engineering

Dr. Phuong Nguyen, Tenure-track Assistant Professor
DISIM, University of L'Aquila, Italy
Email: phuong.nguyen@univaq.it; Website: https://www.disim.univaq.it/ThanhPhuong

A research proposal submitted to the University of L'Aquila on February 5th, 2024

Abstract. This proposal represents an opportunity to contribute to the advancement of Machine Learning in Software Engineering, and aligns with the commitments to innovative research in the field of ethics and fairness for Artificial Intelligence (AI) systems. The core objective of this proposal is to develop effective techniques to address a key challenge, i.e., improving fairness in recommender systems for software engineering. Should this proposal receive funding, would it pave the way for the formation of a dedicated research team at the University of L'Aquila. Being Principal Investigator (PI), I will lead the team to work diligently to advance ongoing research efforts, and propose practical solutions that address unresolved issues in the field of Machine Learning for Software Engineering.

1 General Information

- Title: Defusing Popularity Bias in Recommender Systems for Software Engineering.
- Principal Investigator: Phuong Thanh Nguyen.
- Position: Tenure-track Assistant Professor (RTD/b), University of L'Aquila.
- Type of project: Progetti di ricerca.

2 Introduction

When building new software, developers usually have to deal with an overload of information from heterogeneous and rapidly evolving open sources. Recommender systems for software engineering (RSSEs) [19] serve as an effective means to provide developers with instant support consisting of different items, e.g., code examples [7, 13], possible third-party components [12, 17], documentation, to name a few. Nevertheless, while the main effort has been spent to make RSSEs more effective and efficient, there are several issues that have not attracted enough attention from the research community.

Moreover, while RSSEs are becoming more and more effective in suggesting handy recommendations, they tend to suffer from popularity bias, i.e., favoring items that are relevant mainly because several developers are using them. While this rewards artifacts that are likely more reliable and well-documented, it would also mean that missing artifacts are rarely used because they are very specific or more recent.

Through systematic literature reviews, we carefully investigated state-of-the-art studies published in several software engineering venues. We found out that while recommender systems have become more effective at suggesting relevant items, they are prone to popularity bias [16] and adversarial attacks [14, 15]. In particular, we realized that dealing with popularity bias in TPL recommender systems has not gained traction from the software engineering community. Among the considered studies, only one approach tries to improve diversity in the recommendation results, nevertheless it suffers from a low prediction accuracy. Moreover, while AML has been well studied in other domains, e.g., online shopping systems [6, 8], or computer vision [11], there exists no work discussing adversarial attempts to RSSEs. In fact, the majority of existing studies attempt to improve the prediction accuracy, and no effort has been spent to tackle popularity bias [16], as well as to deal with the abuse of intentionally manipulated data to compromise recommender systems [15, 14]. So far, there are no concrete countermeasures that can be instantly deployed to defend RSSEs against attacks. In

this respect, we see an urgent need to thoroughly study the related issues, with the ultimate aim of *devising effective mechanisms to improve the fairness in RSSEs*.

3 State-of-the-art research

Being heavily reliant on data and algorithms, recommender systems can be susceptible to biases that may lead to unfair outcomes, potentially jeoparadizing trust in them. For recommender systems, the *long tail effect* indicates that a handful of items are extremely popular, whilst most of the remaining ones, so-called the long tail, are not seen by users [3]. *Popularity bias* is a common phenomenon of general purpose recommender systems [1, 2, 4], i.e., providing to users only frequently seen items. Likewise, RSSEs are no exception, while they become more effective in suggesting handy recommendations, RSSEs also suffer from popularity bias by presenting artifacts used by several developers [16]. While this favors artifacts that are possibly more reliable and well-maintained, it would essentially mean that systems fail to recommend some relevant goals, architecture- or solution-specific artifacts.

In our recent work [16], by means of mixed methods research, i.e., performing both a qualitative and quantitative evaluation, we studied popularity bias recommender systems for mining third-party libraries (TPLs). First, following existing guidelines for such type of study software engineering [9], we investigated whether state-of-the-art research has already tackled the issue of popularity bias. Interestingly, the literature review on major software engineering venues reveals that the issue of dealing with popularity bias has not received enough attention from the community. All of the surveyed studies tackled different issues in library recommendation, with the main aim of improving the relevance of the final ranked list, only one work attempts to tackle popularity, unfortunately, it fails to maintain a trade-off between fairness and accuracy.

Then, we performed a quantitative evaluation on four existing TPL RSSEs, exploring their capability to deal with popular artifacts. The experiments showed that three among the considered systems provide to developers highly popular TPLs. The remaining system, while being able to lessen the effect of frequent TPLs, suffers from a low accuracy. Altogether, we see that cutting-edge research in software engineering neglects the issue of popularity bias in TPL recommender systems, leaving a research gap that has to be bridged.

Our work was positively received by the reviewers, and it was accepted for publication in a CORE Rank A conference[1] with the following details:

- Phuong T. Nguyen, Riccardo Rubei, Juri Di Rocco, Claudio Di Sipio, Davide Di Ruscio, Massimiliano Di Penta "*Dealing with Popularity Bias in Recommender Systems for Third-party Libraries: How far Are We?*", in Proceedings of the IEEE/ACM 20th Int. Conf. on Mining Software Repositories (MSR 2023), DOI: 10.1109/MSR59073.2023.00016.

4 Methodology

Figure 1 depicts the architecture with the proposed module printed in the cyan color, which can be independently assembled to any existing recomender systems, being padded right before the interface to developers to defuse popularity bias. This section presents our proposed approach to fill the research gap introduced in Section 2.

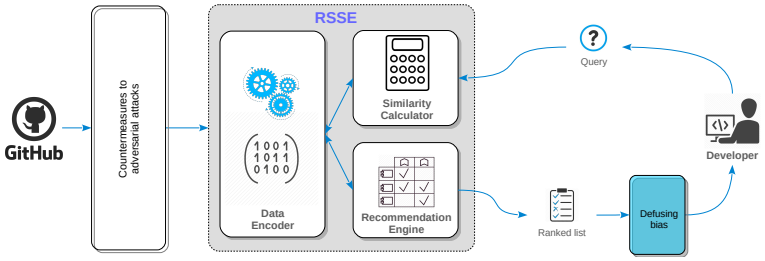

Figure 1: The proposed architecture.

4.1 Research objectives

Within the funded project, we are going to answer the following research questions.

[1] http://portal.core.edu.au/conf-ranks/?search=MSR&by=all&source=CORE2023&sort=atitle&page=1

– **RQ**: "*How can we make RSSEs less biased towards popular items, while still preserving accuracy?*" We plan to develop novel methods to defuse popularity bias. Existing mechanisms conceived for other domains might not work well for recommender systems for software engineering, as we showed in our recent work [16]. We anticipate that Reinforcement Learning [21] can be applied to improve re-ranking techniques, taking into consideration the similarity between projects when a rare library needs to be moved up in the ranked list.

To answer the research question, correspondingly we divide the activities into three tasks, namely **T1**, **T2**, and **T3** (see Figure 1), explained in the succeeding subsections.

4.2 Defusing popularity bias

The application of various Machine Learning algorithms to tackle bias in RSSEs.

Looking to other domains, we see that there are three major methods to defuse bias in general, i.e., the pre-processing, in-processing, and post-processing paradigms [5]. Our proposed approach is based on three main techniques, i.e.,.

We suppose that these can also be adopted in software engineering for the same purpose. We are going to perform the following tasks:

- **T1: Re-implementation of existing re-ranking mechanisms.** In our recent work [16], we applied xQuAD [20] to improve diversity in the recommendations. We plan to investigate and re-implement other techniques, with the aim of finding suitable mechanisms to be adopted for RSSEs. Another possible candidate is PFAR [10], a practical approach conceived to allow items to have a fair chance of being recommended. We will considered also personalized ranking [1], which promotes unpopular items in the ranked list, while maintaining a trade off between fairness and accuracy. Once re-implemented, these techniques can be used as baselines for comparison with our proposed approach presented in **T2**.

- **T2: Reinforcement learning for reducing bias.** Existing re-ranking algorithms such as xQuAD [20] attempt to reduce the number of popular items, as well as to increase the number of unpopular (but useful) ones in the results. However, our empirical evaluation on two recommender systems showed that while introducing diversity in the recommendation results, it also introduces a setback in the prediction accuracy. Our findings suggest that further research should be conducted to propose effective counter-measures. We suppose that it is crucial to consider additional factors, e.g., the degree of specificity (to certain solutions) of a library, when it comes to providing recommendation. Thus, we will deploy Reinforcement Learning to implement effective post-processing techniques to defuse popularity bias, aiming to improve diversity in the recommendations, while keeping a reasonable prediction accuracy.

- **T3: Self-improvement learning for improving ranking.** Self-improvement learning allows a model to incorporate its own predictions into the training process [22]. This helps it to learn from its errors, thereby lessening the effects of exposure bias. Such an algorithm can be applied to improve ranking of items provided by a recommender system. In particular, we train the model with collected training data, and once the model has been trained, it is used to generate for each sample in the training dataset a set of predictions. For each training sample, we compute the diversity scores between the ground-truth list and all the predictions. The predicted libraries with the highest diversity is then chosen to replace the ground-truth one. We get a new training pair composed of the input sample, and the new ground-truth library. By replicating the steps with all the training samples, we get the so-called augmented training dataset that can then be used to re-train the model.

5 Plan

The Gantt chart in Table 1 depicts a tentative plan for the entire project. Within the one-year funded research, we will conduct activities to fulfill the objectives. Besides the activities related to the implementation and evaluation pertaining to the three defined tasks (T1, T2, and T3), we will also devote ourselves to writing deliverables, reports, and papers. In particular, we plan to submit one deliverable every six months, and at the end of the project, the final report. In parallel, we will review related work, and write papers to submit to conferences and journals.

Table 1: Task schedule.

		Month											
Task	**Description**	1	2	3	4	5	6	7	8	9	10	11	12
Reading	Reviewing SOTA												
T1	Re-implementation of re-ranking techniques												
T2	Reinforcement learning for reducing bias												
T3	Self-improvement learning for ranking												
Writing	Deliverable D1												
Writing	Deliverable D2												
Writing	Final report												
Writing	Papers												

A tentative budget plan is shown in Table 2. The award will be completely used for the research activities pertinent to the project. The largest part of the money will be paid two postdoctoral researchers for the duration of one year. We also plan to spend money for the purchase of equipment, including servers and laptops for running the experiments. Moreover, a certain amount of the budget is reserved for registration fees and travel expense for conferences and meetings.

Table 2: Budget distribution.

No.	**Item**	**Amount (EURO)**
1	Paying salary for two master students to work on the project	10,000
2	Acquisition of devices, including GPU computers, laptops, monitors, keyboards, and mouses	5,000
3	Registration fee and travel expenses for conferences/meetings	5,000

6 Results

The proposed research on fairness in recommender systems for software engineering has the potential to yield significant outcomes, positively impacting the software engineering field, and artificial intelligence. We aim at innovation of the proposal and impact in terms of relevance of the progress in basic research for the relevant scientific community. The findings and defense mechanisms developed in this research will be disseminated through academic publications and presentations, contributing to the body of knowledge in software engineering.

6.1 Novelty

We aim to empower RSSEs with the capability to effectively deal with popularity bias, while maintaining or even enhancing their accuracy. Our work distinguishes itself from state-of-the-art research in the following aspects.

- **Pioneering exploration.** We take the initiative to bring attention to the previously overlooked issues of popularity bias affecting RSSEs. This represents a pioneering effort in the field.
- **Holistic approach.** We propose a holistic and integrated approach to fortify RSSEs, leveraging tailored reinforcement learning. This comprehensive strategy is designed to enhance their fairness.
- **Applicability.** The techniques we develop are not limited to RSSEs only, they can be readily applied to fine-tune pre-trained deep learning models in the broader context of software engineering, contributing to the advancement of AI-driven solutions in the field. The potential of Large Language Models (LLMs) in software engineering has been recently acknowledged [18]. However, these models are trained on vast and diverse datasets, which can introduce biases. Additionally, LLMs often require prompt tuning to tailor them to specific purposes, necessitating fine-tuning with relevant data. In this context, the techniques developed in this research project offer practical solutions by mitigating biases within training data for LLMs.

6.2 Implications

- **Enhancing user trust and satisfaction.** By mitigating bias and promoting fairness in recommender systems, our approach aims to enhance user trust and satisfaction with software engineering platforms.

As users rely on recommendations to make critical decisions in software engineering, our research helps to preserve user trust by ensuring that recommendations remain trustworthy.

- **Improving system performance.** Fairness-aware recommender systems are likely to generate recommendations that are not only unbiased but also more accurate and relevant to users. This improvement in recommendation quality can positively impact the overall performance of software engineering platforms.
- **Ethical and responsible AI development.** Addressing fairness issues in recommender systems aligns with the principles of ethical and responsible AI development. Our research contributes to the responsible use of AI in software engineering by reducing potential biases and discrimination.
- **Educational impact.** The ethical implications of AI, including bias mitigation and adversarial counteraction, have gained significant attention in recent years. Our research offers an opportunity to educate future AI professionals about the importance of ethical considerations when developing AI-based systems. It can be integrated into AI ethics courses to promote responsible AI development.

6.3 Prospective publications

The findings and methodologies conceived in this funded project will contribute to state-of-the-art research in software engineering and AI ethics. We will target both the software engineering, and the machine learning communities, aiming to have *at least two articles* submitted and accepted for publication to a Rank A or A* conference,[2] or Scimago Q1 journals.[3] The following venues are considered: Int. Conf. on Automated Software Engineering (ASE, Rank A*); Int. Conf. on Software Engineering (ICSE, Rank A*); Int. Conf. on Evaluation and Assessment in Software Engineering (EASE, Rank A); Int. Conf. on Mining Software Repositories (MSR, Rank A); The ACM Conf. on Recommender Systems (RecSys, Rank A); Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER, Rank A); The ACM Int. Conf. on Information and Knowledge Management (CIKM, Rank A); Elsevier Information and Software Technology Journal (IST, Q1); Elsevier Journal of Systems and Software (JSS, Q1); Elsevier Expert Systems with Applications (ESWA, Q1); IEEE Transactions on Software Engineering (TSE, Q1).

References

[1] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," in *Proceedings of AAAI 2019, 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

[2] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation," in *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with RecSys 2019, Copenhagen, Denmark, September 20, 2019*, ser. CEUR Workshop Proceedings, R. Burke, H. Abdollahpouri, E. C. Malthouse, K. P. Thai, and Y. Zhang, Eds., vol. 2440. CEUR-WS.org, 2019. [Online]. Available: http://ceur-ws.org/Vol-2440/paper4.pdf

[3] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

[4] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He†, "Bias and debias in recommender system: A survey and future directions," *ACM Trans. Inf. Syst.*, oct 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3564284

[5] B. d'Alessandro, C. O'Neil, and T. LaGatta, "Conscientious classification: A data scientist's guide to discrimination-aware classification," *Big Data*, vol. 5, no. 2, pp. 120–134, 2017, pMID: 28632437. [Online]. Available: https://doi.org/10.1089/big.2016.0048

[6] Y. Deldjoo, T. Di Noia, and F. A. Merra, "Adversarial machine learning in recommender systems (aml-recsys)," in *Proceedings of the 13th Int. Conf. on Web Search and Data Mining*, ser. WSDM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 869–872. [Online]. Available: https://doi.org/10.1145/3336191.3371877

[2] CORE Rankings Portal `http://portal.core.edu.au/conf-ranks/`

[3] Scimago Journal & Country Rank `https://www.scimagojr.com/`

[7] J. Fowkes and C. Sutton, "Parameter-free Probabilistic API Mining Across GitHub," in *24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York: ACM, 2016, pp. 254–265.

[8] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey," *Artif. Intell. Rev.*, vol. 42, no. 4, p. 767–799, Dec. 2014. [Online]. Available: https://doi.org/10.1007/s10462-012-9364-9

[9] B. A. Kitchenham, P. Brereton, Z. Li, D. Budgen, and A. J. Burn, "Repeatability of systematic literature reviews," in *15th Int. Conf. on Evaluation & Assessment in Software Engineering, EASE 2011, Durham, UK, 11-12 April 2011, Proceedings*, 2011, pp. 46–55. [Online]. Available: https://doi.org/10.1049/ic.2011.0006

[10] W. Liu and R. Burke, "Personalizing fairness-aware re-ranking," *CoRR*, vol. abs/1809.02921, 2018. [Online]. Available: http://arxiv.org/abs/1809.02921

[11] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." in *CVPR*. IEEE Computer Society, 2015, pp. 427–436. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#NguyenYC15

[12] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "CrossRec: Supporting Software Developers by Recommending Third-party Libraries," *Journal of Systems and Software*, p. 110460, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121219302341

[13] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, L. Ochoa, T. Degueule, and M. Di Penta, "FOCUS: A Recommender System for Mining API Function Calls and Usage Patterns," in *Proceedings of the 41st Int. Conf. on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 1050–1060. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00109

[14] P. T. Nguyen, D. Di Ruscio, J. Di Rocco, C. Di Sipio, and M. Di Penta, "Adversarial machine learning: On the resilience of third-party library recommender systems," in *Proceedings of the 25th Int. Conf. on Evaluation and Assessment in Software Engineering*, ser. EASE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 247–253. [Online]. Available: https://doi.org/10.1145/3463274.3463809

[15] P. T. Nguyen, C. Di Sipio, J. Di Rocco, M. Di Penta, and D. Di Ruscio, "Adversarial attacks to api recommender systems: Time to wake up and smell the coffee?" in *2021 36th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*, 2021, pp. 253–265. [Online]. Available: https://doi.org/10.1109/ASE51524.2021.9678946

[16] P. T. Nguyen, R. Rubei, J. Di Rocco, C. Di Sipio, D. Di Ruscio, and M. Di Penta, "Dealing with popularity bias in recommender systems for third-party libraries: How far are we?" in *2023 IEEE/ACM 20th Int. Conf. on Mining Software Repositories (MSR)*, 2023, pp. 12–24. [Online]. Available: https://doi.org/10.1109/MSR59073.2023.00016

[17] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, and K. Inoue, "Search-based software library recommendation using multi-objective optimization," *Inf. Softw. Technol.*, vol. 83, no. C, pp. 55–75, Mar. 2017. [Online]. Available: https://doi.org/10.1016/j.infsof.2016.11.007

[18] I. Ozkaya, "Application of large language models to software engineering tasks: Opportunities, risks, and implications," *IEEE Software*, vol. 40, no. 3, pp. 4–8, 2023.

[19] M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, *Recommendation Systems in Software Engineering*. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-642-45135-5

[20] R. L. Santos, C. Macdonald, and I. Ounis, "Exploiting query reformulations for web search result diversification," in *Proceedings of the 19th Int. Conf. on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 881–890. [Online]. Available: https://doi.org/10.1145/1772690.1772780

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998. [Online]. Available: https://www.worldcat.org/oclc/37293240

[22] C. Wang and R. Sennrich, "On exposure bias, hallucination and domain shift in neural machine translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3544–3552.