Fairer Recommendation Systems through Graph Neural Networks

John Hannebery

The University of Melbourne

May 2023

Overview

- In this thesis, we address fairness in recommendation systems from the user perspective
- We hypothesise that a certain class of recommendation algorithms (graph neural networks) can lead to fairer recommendations than traditional approaches (matrix factorisation)
- We will
  - Introduce recommendation systems
  - Review fairness in recommendation systems
  - Understand matrix factorisation approaches
  - Understand graph neural network approaches and why they can be more fair
  - Conduct experiments to address our hypothesis

Recommendation Systems

Recommendation Systems

A brief overview

- One of the most prolific applications of machine learning (ML) deployed today
- Powers many top internet sites' product offerings including YouTube [1], Spotify [2], Netflix [3] and TikTok [4].
- Retrieve and rank relevant items out of a large corpus of content

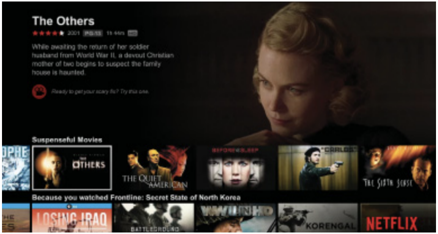


Figure: Netflix recommendations from [5]

Recommendation Systems

Data for Recommendation

- User-item interactions (explicit or implicit feedback)
- Explicit feedback
  - User gives true preference to an item
  - Rating data (out of 5)
  - Sparse
- Implicit feedback
  - Purchases, clicks, searches
  - More readily available
  - Assumed as a positive interaction
- Encoded into $\mathbf{A} \in \mathbb{R}^{M \times N}$ for M users and N items.
  $A_{ij} = 1$ if user i has interacted with item j, else 0

Recommendation Systems

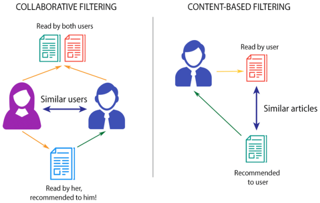
Mathematical Problem

- Sets
  - M users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$
  - N items $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$
- Data
  - Encoded into $\mathbf{A} \in \mathbb{R}^{M \times N}$
  - Split into train and test set
- Objective
  - Predict preference scores for users to items $\{S_{ij} | i \in \mathcal{U}, j \in \mathcal{V}\}$.
  - Encoded into $\mathbf{S} \in \mathbb{R}^{M \times N}$
- Output
  - Produces a ranked top-K item list of recommendations for each user i
  - Achieved by sorting $\mathbf{s}_i \in \mathbb{R}^N$, the preference vector for user i

Recommendation Systems

Approaches

- Content based
  - Recommend similar items to the users preference
  - Considers each user in isolation
- Collaborative filtering (CF)
  - Assumes similar users will like similar items
  - Learn user embeddings $\mathbf{u}_i \in \mathbb{R}^{M\times d}$ and item embeddings $\mathbf{v}_j \in \mathbb{R}^{N\times d}$, in an inner product space to model preference
  - Modelled through shallow methods (matrix factorisation), deep neural networks or graph neural networks



Recommendation Systems

Evaluation Methods

- Traditionally analysed via accuracy metrics
  - Attempts to measure recommendation quality
  - Different metrics usually analysed together
  - May fail to capture essential recommendation system aspects
- Beyond accuracy metrics proposed to measure other qualities of a recommendation system (novelty, diversity, filter bubbles)
- Fairness is another way of evaluating a recommendation system to understand bias and recommendation quality compared across groups

- NDCG [6] emphasises more relevant items higher on a recommendation list:

$$NDCG@K(i) = \frac{DCG@K(i)}{IDCG@K(i)} \quad (1)$$

where

$$DCG@K(i) = \sum_{l \in rec_K(i)} \frac{2^{rating(i,l)} - 1}{log_2\left(rank\left(i, l\right) + 1\right)} \quad (2)$$

- Hit-rate [6] measures whether at least one item in the ranked list is relevant, defined as

$$HR@K(i) = \mathbb{1}\left[rel(i) \cap rec_K(i) > 0\right] \quad (3)$$

Recommendation Systems

Beyond Accuracy Metrics

- Diversity [7] measures the range of recommendations:

$$Diversity@K(i) = \frac{\sum_{l \in rec_K(i)} \sum_{j \in rec_K(i) \setminus l} v_l \cdot v_j}{K(K-1)} \tag{4}$$

- Coverage [7] measures the percentage of items recommended:

$$Coverage@K = \frac{|\cup_{i \in \mathcal{U}} rec_K(i)|}{|\mathcal{V}|} \tag{5}$$

- Novelty [7] measures the uniqueness of a set of recommendations:

$$Novelty@K(i) = \frac{\sum_{j \in rec_K(i)} - \log_2 p(j)}{K} \tag{6}$$

where $p(j)$ is the popularity of item j:

$$p(j) = \frac{|\{i \in \mathcal{U}, rating(i,j) = 1\}|}{|\mathcal{U}|} \tag{7}$$

Fairness in Recommendation Systems

Fairness in Recommendation Systems

Fairness in Machine Learning

- "A fair and ethical machine learning system should be inclusive and accessible, and should not involve or result in unfair discrimination against individuals, communities or groups" [8].
- Fairness issues arise from bias in the data (unrepresentative data, minority groups) and can be propagated via ML algorithms
- General ML fairness solutions involve:
  - pre-processing (data augmentation to reduce bias)
  - in-processing (regularisation or constrained optimisation)
  - post-processing (re-ranking of results)

Fairness in Recommendation Systems

Bias in Recommendation Systems

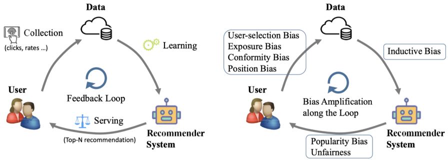


Figure: Recommendation system feedback loop from [9]

- Like ML, fairness arises from biases, although propagated through an explicit feedback loop

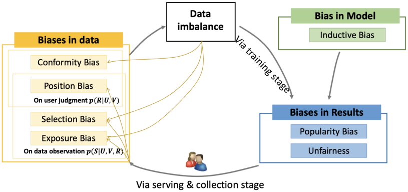


Figure: Bias feedback loop in RS from [9]

Fairness in Recommendation Systems

Types of Fairness

- ML fairness methods don't easily extend to RS due to multi-sided nature (users and items)
- Fairness in recommendation system can generally be split into user, item and multi-sided
- Item fairness aims for fair exposure, lots of research on popularity bias [10–12].
- Multi-sided fairness trades off user and item fairness together [13–15].

User Fairness in Recommendation Systems I

- Aims for minority groups to receive similar recommendations to other non-protected groups.
- Measured via a comparison of a metric between groups:

$$Fairness(G_1, G_2, M) \tag{8}$$

where G_1 is one group of users, G_2 is the other group of users such that $G_1 \cap G_2 = \emptyset$ and M is any recommendation metric

User Fairness in Recommendation Systems II

- [16] splits users into high activity and low activity. Low activity users receive poorer recommendations than the active users via NDCG. Propose a heuristic re-ranking technique to improve fairness.
- They show fairness improves post-heuristic according to

$$Fairness(G_1, G_2, M) = \left| \frac{1}{G_1} \sum_{i \in G_1} M(i) - \frac{1}{G_2} \sum_{i \in G_2} M(i) \right| \quad (9)$$

User Fairness in Recommendation Systems III

- [17] similarly groups users into advantaged and disadvantaged based on interactions, price and consumption. Advantaged users receive better recommendations, apply a similar re-ranking approach to improve fairness, using the same metric.
- [18] generalises [17], repeating the approach for multiple algorithms and multiple datasets.
- [19] split users into groups based on personality traits and compare recommendation performance
- [20] split users into groups based on age and gender, comparing recommendation performance via NDCG
- One paper [21] considered user fairness from a beyond-accuracy perspective, although metrics were drawn from survey questions

Gaps in the research

- No explicit research on out of the box algorithms
  - Graph neural networks can learn higher order structure in the data $\rightarrow$ could lead to fairer outcomes for disadvantaged users
- Beyond accuracy and accuracy: a lack of research together in fairness to get a holistic overview
  - Just because a RS is fair for accuracy, doesn't mean it's the case for beyond-accuracy (and vice versa)

Matrix Factorisation for Recommendation Systems

Matrix Factorisation for Recommendation Systems

Overview

- Matrix factorisation [22] is a CF method that decomposes $\mathbf{A}$ into embedding matrices $\mathbf{U} \in \mathbb{R}^{M \times d}$ and $\mathbf{V} \in \mathbb{R}^{N \times d}$:

$$\mathbf{A} \approx \mathbf{S} = \mathbf{U}\mathbf{V}^T \tag{10}$$

- The preference score for user i to item j, $\mathbf{S}_{ij}$ is expressed via the inner product of their embeddings:

$$\mathbf{S}_{ij} = \mathbf{u}_i \cdot \mathbf{v}_j^T \tag{11}$$

Matrix Factorisation for Recommendation Systems

Example

- Consider a toy example with 3 users and 3 items (movies).

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- The embedding matrices $\mathbf{U}, \mathbf{V}$ and also $\mathbf{S}$ are learned as

$$\mathbf{U} = \begin{bmatrix} 1 & 0.1 \\ -1 & 0 \\ 0.2 & -1 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} 0.9 & -0.2 \\ -1 & -0.8 \\ 1 & -1 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0.88 & -1.08 & 0.9 \\ -0.9 & 1 & -1 \\ 0.38 & 0.6 & 1.2 \end{bmatrix}$$

$$S_{11} = u_1 \cdot v_1^T = \begin{bmatrix} 1 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0.9 \\ -0.2 \end{bmatrix} = 0.88$$

- User 1's ordered recommendations are items $3, 1, 2$.

- Singular value decomposition [23] decomposes $\mathbf{A}$ into

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{12}$$

  $\mathbf{U} \in \mathbb{R}^{M\times M}$, $\mathbf{V} \in \mathbb{R}^{N\times N}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{M\times N}$ is diagonal.

- While the above is computationally heavy, a low-rank approximation can be found:

$$\mathbf{A} = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T \tag{13}$$

  with $\mathbf{U}_k\Sigma_k^{1/2}$ user embeddings and $\mathbf{V}_k\Sigma_k^{1/2}$ item embeddings, and preference score $\mathbf{S}_{ij}$:

$$\mathbf{S}_{ij} = \left(\mathbf{U}_k\Sigma_k^{1/2}\right)_i \left(\mathbf{V}_k\Sigma_k^{1/2}\right)_j \tag{14}$$

Matrix Factorisation for Recommendation Systems

Classical Matrix Factorisation

- Classical matrix factorisation rose to prominence during the Netflix Prize [24], with a winning solution proposed in [22].
- Embeddings are learned that minimise the sum of squared errors across all positive (the set P) user, item pairs i, j:

$$\min \sum_{i,j \in P} \left(\mathbf{A}_{ij} - \mathbf{u}_i \mathbf{v}_j^T \right)^2 + \lambda \left|\left|\mathbf{u}_i\right|\right|^2 + \left|\left|\mathbf{v}_j\right|\right|^2 \tag{15}$$

- Stochastic gradient descent used to update embeddings:

$$\mathbf{u}_i = \mathbf{u}_i + \alpha \left(e_{ij} \mathbf{v}_j - \lambda \mathbf{u}_i \right) \tag{16}$$

$$\mathbf{v}_j = \mathbf{v}_j + \alpha \left(e_{ij} \mathbf{u}_i - \lambda \mathbf{v}_j \right) \tag{17}$$

with e_{ij} as the error from prediction:

$$e_{ij} = \mathbf{A}_{ij} - \mathbf{u}_i \mathbf{v}_j^T \tag{18}$$

Matrix Factorisation for Recommendation Systems

Alternating Least Squares

- Alternating least squares [22]: fix $\mathbf{U}$ or $\mathbf{V}$ and a quadratic form can be obtained.
- Take v_j constant, solving the loss function yields for $\mathbf{u}_i$:

$$\mathbf{u}_i = \mathbf{A}_i \mathbf{V} \left(\mathbf{V}^T \mathbf{V} + \lambda \mathbf{I} \right)^{-1} \tag{19}$$

  In a similar fashion, taking u_i constant, solving for $\mathbf{v}_j$ yields

$$\mathbf{v}_j = \mathbf{A}_j \mathbf{U} \left(\mathbf{U}^T \mathbf{U} + \lambda \mathbf{I} \right)^{-1} \tag{20}$$

- Repeat in a two-step process until convergence.

Graph Neural Networks for Recommendation Systems

- Graph neural networks (GNN's) are a general framework for defining deep learning algorithms over graph-structured data
- Graph structured data consists of nodes and edges
- At each layer in a GNN, the embedding of a node is refined based on the neighbors, through three steps:
  - Messaging: Neighbours pass their embeddings as messages
  - Aggregation: Messages are aggregated for the layer
  - Updating: Combine previous layer embeddings with current layer neighbour embeddings to update

Graph Neural Networks for Recommendation Systems

Applying to Recommendation Systems

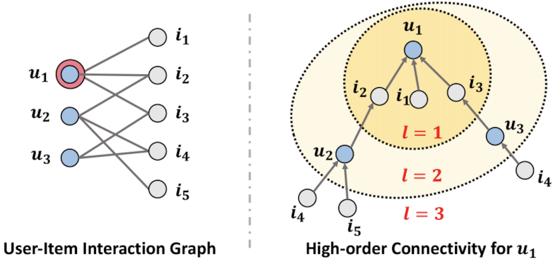


Figure: User-item interactions as a graph from [25]

- To apply graph neural networks to recommendation system, represent user-item interactions as a graph.
- Embeddings can be propagated and stacked to access higher order information in the interaction data

Graph Neural Networks for Recommendation Systems

Neural Graph Collaborative Filtering [25]

- Define a message embedding for a user-item pair (i, j) as

$$\mathbf{m}_{i \leftarrow j} = \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}} \left(\mathbf{W}_1 \mathbf{v}_j + \mathbf{W}_2 \left(\mathbf{v}_j \odot \mathbf{u}_i\right)\right) \tag{21}$$

  where $\mathbf{W}_1, \mathbf{W}_2 \in R^{d' \times d}$ are learnable weight matrices and $\mathcal{N}_i$ and $\mathcal{N}_j$ represent the first-hop neighbours of user i and item j.
- Messages can be aggregated from neighbours to refine the embedding after one layer:

$$\mathbf{u}_i^{(1)} = f\left(\mathbf{m}_{i \leftarrow i} + \sum_{k \in \mathcal{N}_i} \mathbf{m}_{i \leftarrow k}\right) \tag{22}$$

  including self-connection $(\mathbf{m}_{i \leftarrow i} = \mathbf{W}_1 \mathbf{u}_i)$

Graph Neural Networks for Recommendation Systems

Neural Graph Collaborative Filtering [25]

- To include higher order structure and improve CF, propagate embeddings across multiple layers to refine further:

$$\mathbf{u}_i^{(l)} = f\left(\mathbf{m}_{i\leftarrow i}^{(l)} + \sum_{k\in\mathcal{N}_i} \mathbf{m}_{i\leftarrow k}^{(l)}\right) \tag{23}$$

with the message propagation functions defined as

$$\mathbf{m}_{i\leftarrow j}^{(l)} = p_{ij}\left(\mathbf{W}_1^{(l)}\mathbf{v}_j^{(l-1)} + \mathbf{W}_2^{(l)}\left(\mathbf{v}_j^{(l-1)} \odot \mathbf{u}_i^{(l-1)}\right)\right) \tag{24}$$

$$\mathbf{m}_{i\leftarrow i}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{u}_i^{(l-1)} \tag{25}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in R^{d_l\times d_{l-1}}$ are different weight matrices for that layer, and $\mathbf{u}_i^{(l-1)}$ and $\mathbf{v}_j^{(l-1)}$ are aggregated embedding representations from previous layers.

Graph Neural Networks for Recommendation Systems

Neural Graph Collaborative Filtering [25]

- The l layers encode different levels of information, for users as $\{\mathbf{u}_i^{(1)}, \ldots, \mathbf{u}_i^{(L)}\}$ and items as $\{\mathbf{v}_j^{(1)} || \ldots || \mathbf{v}_j^{(L)}\}$.
- Embeddings are combined by concatenation to yield a final user and item embedding:

$$\mathbf{u}_i^* = \mathbf{u}_i^{(0)} || \ldots || \mathbf{u}_i^{(L)} \tag{26}$$

$$\mathbf{v}_j^* = \mathbf{v}_j^{(0)} || \ldots || \mathbf{v}_j^{(L)} \tag{27}$$

The inner product is used to calculate similarity between a user i and item j as in (11),

$$\mathbf{S}_{ij} = \mathbf{u}_i^* \cdot \mathbf{v}_j^{*T} \tag{28}$$

Graph Neural Networks for Recommendation Systems

LightGCN [26]

- LightGCN [26] was proposed, eliminating the weight matrices and nonlinear activation function, surprisingly leading to faster and more accurate recommendation.
- The same message passing and aggregation principle applies.
- The embedding propagation step becomes (for user embedding i and item embedding j at layer l):

$$\mathbf{u}_i^{(l)} = \sum_{p \in N_i} \frac{1}{\sqrt{|N_i||N_p|}} \mathbf{u}_i^{(l-1)} \tag{29}$$

$$\mathbf{v}_j^{(l)} = \sum_{p \in N_j} \frac{1}{\sqrt{|N_p||N_j|}} \mathbf{v}_j^{(l-1)} \tag{30}$$

Graph Neural Networks for Recommendation Systems

LightGCN [26]

- The embeddings are then aggregated across the l layers via a sum,

$$\mathbf{u}_i^* = \sum_{m=0}^{l} \alpha_m \mathbf{u}_i^{(m)} \tag{31}$$

$$\mathbf{v}_j^* = \sum_{m=0}^{l} \alpha_m \mathbf{v}_j^{(m)} \tag{32}$$

  where α_m is the importance of the weights at each layer.

- The predicted preference is also the inner product,

$$\mathbf{S}_{ij} = \mathbf{u}_i^* \cdot {\mathbf{v}_j^*}^T \tag{33}$$

Evaluation and Experiments

Evaluation and Experiments

Motivation

- Gap 1: lack of investigation into which off-the-shelf algorithms are most effective at promoting fairness between users
- Gap 2: lack of research considering both accuracy and beyond-accuracy metrics in assessing fairness.
- This leads us to answer two research questions via experiments:
  RQ1 (Primary Research Question): Can graph neural networks lead to fairer outcomes in recommendation systems compared to matrix factorisation algorithms?
  RQ2 (Secondary Research Question): Does a different picture of fairness in recommendation systems emerge when we go beyond traditional accuracy metrics?

Evaluation and Experiments

Experimental Setup

- **User grouping**: Lowest 80% quantile of activity as the low activity (G_1) group, and the rest as high activity users (G_2), such that $G_1 \cap G_2 = \emptyset$.
- **Datasets**
  - **Last-FM** [27]: Music listening dataset
  - **MovieLens 100K** [28]: Movie review dataset
- **Algorithms**
  - **Alternating Least Squares (ALS)** Implemented using Apache Spark MLlib library in Python.
  - **LightGCN** Implemented using TensorFlow in Python.

Evaluation and Experiments

Metrics

- **Fairness**: define the fairness score under the generic framework (8)

$$Fairness(G_1, G_2, M) = \frac{1}{|G_2|}\sum_{i \in G_2} M(i) - \frac{1}{|G_1|}\sum_{i \in G_1} M(i) \quad (34)$$

- **Accuracy metrics**: NDCG@K (1) and HR@K (3)
- **Beyond-accuracy metrics**: Novelty@K (6) and Diversity@K (4)
- For all metrics, set $K = 10$
- Metrics are computed on a hold-out test set.
  - For the low activity users, compute metrics on the test set with only their interactions
  - Repeat for the high activity users

Results

MovieLens		NDCG@10	Hitrate@10	Diversity@10	Novelty@10
ALS	**All**	0.18	0.38	1.25	2.99
	Low	0.16	0.33	1.23	2.83
	High	0.26	0.56	1.33	3.58
	Fairness	0.10	0.23	**0.10**	0.75
LightGCN	**All**	0.18	0.42	2.19	2.27
	Low	0.18	0.40	2.23	2.32
	High	0.21	0.51	2.03	2.11
	Fairness	**0.03**	**0.11**	-0.20	**-0.21**
Last-FM		**NDCG@10**	**Hitrate@10**	**Diversity@10**	**Novelty@10**
ALS	**All**	0.11	0.21	0.39	6.66
	Low	0.09	0.18	0.39	6.64
	High	0.18	0.36	0.36	6.73
	Fairness	0.09	0.18	**-0.03**	**-0.09**
LightGCN	**All**	0.20	0.42	3.51	7.54
	Low	0.20	0.40	3.50	7.58
	High	0.22	0.50	3.55	7.38
	Fairness	**0.02**	**0.10**	0.05	0.20

- Results show LightGCN fairer on both datasets for accuracy metrics
- ALS fairer for both beyond-accuracy metrics on Last-FM, although un-normalised. LightGCN fairer by larger margin
- Speculate LightGCN fairer for accuracy due to high order information gained using a graph neural network
  - Through this, low activity users can have better recommendations, learning more from the data
- High activity users have more accurate recommendations for both datasets

References I

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

[2] Rishabh Mehrotra and Benjamin Carterette. Recommendations in a marketplace. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 580–581, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi: 10.1145/3298689.3346952. URL https://doi.org/10.1145/3298689.3346952.

References II

[3] Mohammad Saberian and Justin Basilico. Recsysops: Best practices for operating a large-scale recommender system. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 590–591, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3474620. URL https://doi.org/10.1145/3460231.3474620.

[4] Zhuoran Liu, Leqi Zou, Xuan Zou, Caihua Wang, Biao Zhang, Da Tang, Bolin Zhu, Yijie Zhu, Peng Wu, Ke Wang, and Youlong Cheng. Monolith: Real time recommendation system with collisionless embedding table, 2022. URL https://arxiv.org/abs/2209.07663.

References III

[5] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), dec 2016. ISSN 2158-656X. doi: 10.1145/2843948. URL https://doi.org/10.1145/2843948.

[6] Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev. Quality metrics in recommender systems: Do we calculate metrics consistently? In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 708–713, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3478848. URL https://doi.org/10.1145/3460231.3478848.

References IV

[7] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Trans. Interact. Intell. Syst.*, 7(1), dec 2016. ISSN 2160-6455. doi: 10.1145/2926720. URL https://doi.org/10.1145/2926720.

[8] Department of Industry, Science, Energy and Resources. *Australia's Artificial Intelligence Ethics Framework*. Australian Government, 2020. URL https://www.industry.gov.au/sites/default/files/2020-11/australias-artificial-intelligence-ethics-framework.pdf.

References V

[9] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *CoRR*, abs/2010.03240, 2020. URL `https://arxiv.org/abs/2010.03240`.

[10] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation ranking through pairwise comparisons. *CoRR*, abs/1903.00780, 2019. URL `http://arxiv.org/abs/1903.00780`.

[11] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. *CoRR*, abs/1907.13286, 2019. URL `http://arxiv.org/abs/1907.13286`.

References VI

[12] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *CoRR*, abs/1901.07555, 2019. URL http://arxiv.org/abs/1901.07555.

[13] Robin Burke. Multisided fairness for recommendation. *CoRR*, abs/1707.00093, 2017. URL http://arxiv.org/abs/1707.00093.

[14] Himan Abdollahpouri and Robin Burke. Multi-stakeholder recommendation and its connection to multi-sided fairness. *CoRR*, abs/1907.13158, 2019. URL http://arxiv.org/abs/1907.13158.

[15] Gourab K. Patro, Arpita Biswas, Niloy Ganguly, Krishna P. Gummadi, and Abhijnan Chakraborty. Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms. *CoRR*, abs/2002.10764, 2020. URL https://arxiv.org/abs/2002.10764.

References VII

[16] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, and Gerard de Melo. Fairness-aware explainable recommendation over knowledge graphs. *CoRR*, abs/2006.02046, 2020. URL https://arxiv.org/abs/2006.02046.

[17] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. User-oriented fairness in recommendation. *CoRR*, abs/2104.10671, 2021. URL https://arxiv.org/abs/2104.10671.

[18] Hossein A. Rahmani, Mohammadmehdi Naghiaei, Mahdi Dehghan, and Mohammad Aliannejadi. Experiments on generalizability of user-oriented fairness in recommender systems, 2022. URL https://arxiv.org/abs/2205.08289.

References VIII

[19] Alessandro B. Melchiorre, Eva Zangerle, and Markus Schedl. Personality bias of music recommendation algorithms. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 533–538, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. doi: 10.1145/3383313.3412223. URL https://doi.org/10.1145/3383313.3412223.

[20] Michael D. Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D. Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 172–186. PMLR, 23–24 Feb 2018. URL https://proceedings.mlr.press/v81/ekstrand18b.html.

References IX

[21] Ningxia Wang and Li Chen. User bias in beyond-accuracy measurement of recommendation algorithms. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 133–142, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3474244. URL https://doi.org/10.1145/3460231.3474244.

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, aug 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

References X

[23] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

[24] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of the KDD Cup Workshop 2007*, pages 3–6, New York, August 2007. ACM. URL `http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf`.

[25] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. *CoRR*, abs/1905.08108, 2019. URL `http://arxiv.org/abs/1905.08108`.

References XI

[26] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *CoRR*, abs/2002.02126, 2020. URL https://arxiv.org/abs/2002.02126.

[27] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. *CoRR*, abs/1905.07854, 2019. URL http://arxiv.org/abs/1905.07854.

[28] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.