5-2022

New debiasing strategies in collaborative filtering recommender systems: modeling user conformity, multiple biases, and causality.

Mariem Boujelbene
University of Louisville

NEW DEBIASING STRATEGIES IN COLLABORATIVE FILTERING RECOMMENDER SYSTEMS: MODELING USER CONFORMITY, MULTIPLE BIASES, AND CAUSALITY

By

Mariem Boujelbene
M.Sc., Computer Engineering and Computer Science,
University of Louisville, Louisville, KY

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Computer Science and Engineering

Department of Computer Science and Engineering
University of Louisville
Louisville, Kentucky

May 2022

NEW DEBIASING STRATEGIES IN COLLABORATIVE FILTERING RECOMMENDER SYSTEMS: MODELING USER CONFORMITY, MULTIPLE BIASES, AND CAUSALITY

By

Mariem Boujelbene
M.Sc., Computer Engineering and Computer Science,
University of Louisville, Louisville, KY

A Dissertation Approved On

25 April 2022

by the following Dissertation Committee:

Dr. Olfa Nasraoui, Dissertation Director

Dr. Hichem Frigui

Dr. Nihat Altiparmak

Dr. Cara Cashon

Dr. Juwon Park

ACKNOWLEDGEMENTS

I would like to thank my adviser Dr. Olfa Nasraoui for her guidance and support during the past five years. She supported us every step of the way and inspired us to aim for exellence. Her mentorship and feedback have guided us toward the right direction and enabled us to further improve our research. I thank her for being not only an incredible advisor but also an amazing friend.

I want to thank my committee members, Drs. Cara Cashon, Hichem Frigui, Juw Won Park, and Nihat Altiparmak, for accepting to be part of this journey, for reviewing my work, and for their insightful feedback.

Last but not least, I thank my husband Sami for his patience and his support; and for the fruitful research discussions that inspired and contributed to this work. I also thank my family for supporting me during this journey, with in particular, a special thanks to my mother for all the sacrifices she has made. Without her, none of this work would be possible.

I would also like to acknowledge the National Science Foundation for partially supporting my research through grants IIS-1549981, CNS-1828521, and DRL-2026584.

I would finally like to thank my puppy Simba for his company during the late work nights.

ABSTRACT

NEW DEBIASING STRATEGIES IN COLLABORATIVE FILTERING RECOMMENDER SYSTEMS: MODELING USER CONFORMITY, MULTIPLE BIASES, AND CAUSALITY

Mariem Boujelbene

April 25 2022

Recommender Systems are widely used to personalize the user experience in a diverse set of online applications ranging from e-commerce and education to social media and online entertainment. These State of the Art AI systems can suffer from several biases that may occur at different stages of the recommendation life-cycle. For instance, using biased data to train recommendation models may lead to several issues, such as the discrepancy between online and offline evaluation, decreasing the recommendation performance, and hurting the user experience. Bias can occur during the data collection stage where the data inherits the user-item interaction biases, such as selection and exposure bias. Bias can also occur in the training stage, where popular items tend to be recommended much more frequently given that they received more interactions to start with. The closed feedback loop nature of online recommender systems will further amplify the latter biases as well. In this dissertation, we study the bias in the context of Collaborative Filtering recommender system, and propose a new Popularity Correction Matrix Factorization (PCMF) that aims to improve the recommender system performance as well as decrease popularity bias and increase the diversity of items in the recommendation lists. PCMF mitigates popularity bias by disentangling relevance and conformity and by learning a user-personalized bias

vector to capture the users' individual conformity levels along a full spectrum of conformity bias. One shortcoming of the proposed PCMF debiasing approach, is its assumption that the recommender system is affected by only popularity bias. However in the real word, different types of bias do occur simultaneously and interact with one another. We therefore relax the latter assumption and propose a multi-pronged approach that can account for two biases simultaneously, namely popularity and exposure bias. our experimental results show that accounting for multiple biases does improve the results in terms of providing more accurate and less biased results. Finally, we propose a novel two-stage debiasing approach, inspired from the proximal causal inference framework. Unlike the existing causal IPS approach that corrects for observed confounders, our proposed approach corrects for both observed and potential unobserved confounders. The approach relies on a pair of negative control variables to adjust for the bias in the potential ratings. Our proposed approach outperforms state of the art causal approaches, proving that accounting for unobserved confounders can improve the recommendation system's performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Artificial Intelligence (AI) applications have witnessed a rapid growth in recent years, thanks to the abundance of available data, models, hardware, and advances in Machine Learning (ML) algorithms, which nowadays power multiple sectors, ranging from health and education to e-commerce, entertainment, and social media. As a result, AI has started affecting people's lives in their daily lives, whether they are applying for a loan, credit card, or jobs; or exploring and interacting with their social media feeds or music and video streaming platforms. Yet, the performance of Machine learning models depends on the quality of the collected data. In fact, in recent years, concerns have been raised about the potential effects of bias emerging from either the algorithms or the data, on the ML models' performance.

Recommendation systems (RS) are one particular application where AI algorithms has made significant improvements in recent years. Recommendation systems power the majority of e-commerce websites, music streaming servers, social media platforms, news websites, etc. Recommendation systems help the users by by providing recommendations that match their preferences, as they navigate online environments and services that are filled with a staggering amount of available options that would otherwise lead to a paralyzing information overload. For this reason, recommender systems can lead to a better user experience, and in turn benefit both the user and the service/business providers. Recommender systems can be divided into three categories. Collaborative Filtering recommender systems [1] operate by leveraging the users' past interactions with the items in order to detect the similarity between users and items without explicitly encoding the user features and item content features. Collaborative Filtering approaches are easy to implement given

that the only information we need to encode is user-item interactions. The second category is content-based filtering [2] which relies solely on the user and item features to build predictive recommendation models. The third category is hybrid approaches [3], which as the name suggests, may rely on user and item features as well as collaborative filtering.

Collaborative filtering recommender systems are currently considered the state of the art given the ease of usage and interaction data collection. They rely on the assumption that users with similar interests have similar interaction behavior and leverage past user-item interaction data to learn patterns or models, for instance based on latent factors, that capture the similarity between users and items. Then these learnt latent factors are used to predict the users' preferences and to recommend items to them. Given that collaborative recommender systems rely solely on historic user-item interaction data, they are more sensitive to the biases that are present in data or in the algorithms, as well as biases emerging from the closed feedback loop between the algorithms' output predictions and their inputs.

Bias can be manifested in different stages of the life cycle of a recommendation system [4]. It can be introduced in the data collection phase, where the data may inherit the underlying biases of the interaction data's collection process [4]. For instance, the user's clicks might be affected by the recommendation list's presentation order [5] [6], or by the popularity of the items [7], the crowd's opinion [8], or past exposure of the user to the items [9] (such as a friend recommendation, ads, etc). Bias might also be introduced during the model training phase [4]. In fact, popular items tend to receive a higher number of interactions which in turn leads to a highly unbalanced training data where the majority categories of items get to control the training loss. The recommender system, by nature, operates in a closed feedback loop setting where feedback from the users is used to provide recommendations for the next iteration. This feedback loop further amplifies the biases discussed above [10, 11].

As recommender systems (RS) continue to permeate through an expanding set of online applications and domains, they are starting to have an increasing impact on human decisions and behaviour. Hence, depending on the application domain and on their scale of

use, biases that emerge within these recommender systems can have significant social and economical impacts [12]. For these reasons, research on understanding bias and designing debiasing strategies has been attracting a growing interest from the recommender systems community [4]. Bias in recommender systems may result in multiple consequences. From the user's perspective, bias might decrease the performance of the recommendation algorithms by pushing popular and less diverse items to the top of the recommendation list, leading to less personalization which may hurt the user experience. From the supplier's perspective, bias may lead to pushing popular items to the top of the recommendation lists more than long tail items, thus leading to unfairness of exposure among the different providers' items. This phenomenon is known as the Mathew effect [13] or the "rich get richer" effect. Bias may also lead to polarization and echo chambers [14, 15].

1.1 Problem Statement

Recommendation systems are known to suffer from extreme popularity bias [16] [17], where popular items are recommended the most at the top of the recommendation list, resulting in an imbalanced training data. The problem stems from the fact that popular items receive more clicks compared to long tail items. This leads to highly unbalanced training data and to a training process that is biased towards popular items. For instance, [16] showed that the recommendation lists generated by Matrix Factorization consist mainly of popular items, regardless of the user's preference for popular items or long tail items. This shows that there is a disparity between the user's conformity level and what the recommender is delivering, which may hurt the user experience. The Inverse Propensity Scoring (IPS) framework [18, 19] is widely used to correct for popularity bias by down-weighing the contribution of popular items using inverse propensity. One limitation of this approach is the non-personalized representation of conformity bias, where the propensities are treated as user-independent and the goal is to remove popularity bias regardless of the user preference towards popular items. Yet the users' preferences for popular or long tail items is expected to vary from one user to the other and this preference should be

captured by the recommender system when debiasing recommendations. A more faithful representation of preference can be decomposed into a general user preference vector and a bias vector that represents the degree that each given user likes popular items. Learning only the general preference of the user, or counteracting the popularity bias without taking into account the relevance, will neglect an important component of the popularity bias and might degrade the performance of the model. On another hand, ignoring the popularity bias of the system will underfit the general preference of the user and will learn a representation that is biased toward the popularity bias component. In addition, conformity bias does not vary only between users, it may vary for the same user depending on the items' categories. Another shortcoming of existing debiasing efforts has been their lack of studying the effect of multiple biases simultaneously. Yet, the logged data may be affected by multiple biases at the same time, such as selection bias, exposure bias, presentation bias, etc.

1.1.1 Dissertation Scope and Impact

In this work, we investigate bias from three main perspectives that advance the state of the art in debiasing research: First, we correct for the popularity bias, while taking into account different users' conformity bias toward popular items, but assuming that the recommender system is affected only by popularity bias. Then, we relax the latter assumption and propose a bias correction mechanism for mitigating two biases simultaneously. Finally, we study bias from a causal perspective in order to create a more robust system to the dynamic shift in the data distribution.

We first propose a popularity correction approach that learns embeddings that can incorporate the user's conformity bias. Our approach aims to mitigate popularity bias by disentangling conformity and relevance. In the second part of this work, we propose an approach that simultaneously accounts for both exposure bias and popularity bias. For instance, users tend to interact more with items presented at the top of the recommendation list, as well as with items that are more popular and have higher ratings, thus potentially leading to both conformity bias and popularity bias.

1.1.1.1 Correcting for Popularity Bias

We propose a novel popularity bias correction matrix factorization, that aims at reducing popularity while providing more accurate recommendations. Personalized correction of popularity bias benefits both users and suppliers. Since, it increases the accuracy of the RS by learning the preference of the user for popular and long tail items while promoting more diverse recommendations. Our method relies at learning richer embedding that accounts for the preference of the user to the popular items. We show that by learning better embeddings, the accuracy of the results increases while keeping a good diversity in the recommendation

1.1.1.2 Correcting for Multiple Biases

We studied the effect of correcting for two biases simultaneously on the recommendation accuracy and diversity. This is crucial, since biases do occur simultaneously and current work usually isolate the system and correct for one bias at the time assuming that biases do not interfere. Correcting for the bias interference may help further improve the accuracy of the system and push more diverse and long tail items.

We combine IPS strategy with previously designed popularity-enhancing approach in order to mitigate the selection bias effect on the training of the model. We show that combining both approaches yields a superior performance in terms of accuracy and bias.

1.1.1.3 Causal Inference Based Debiasing Strategy

One important drawback of recommender systems is ignoring confounders while learning the parameters of the model [4]. Confounders such as popularity, position of the item or the exposure of the user affects the data collection process and therefore the observational data is heavily biased. Although previous methods such as IPS [18] aim at reducing the bias effect, certain unmeasured confounders effect remain present and exacerbate the results. We propose a novel two step approach, inspired from the proximal causal inference framework [20, 21]. We aim at mitigating the effect of unmeasured confounders in

order to get more unbiased estimation of the performance of the model. Our method relies on designing a bridge function that estimates the effect of the unmeasured confounders on the observational data and then use the newly generated data to train a final model. our approach is model agnostic and can be applied with several Collaborative Filtering algorithms.

1.1.1.4 Debiasing using a Causal Framework

we present a new approach to adjust for confounders in recommendation systems. In order to do this, we reframe the recommendation system within a causal framework and then propose to use a proximal causal inference framework [22] to adjust for confounders. Confounders are variables that may affect the predicted ratings leading to biased estimates that can be misleading.

1.1.1.5 Scope of the Research and Assumptions

This research focuses on matrix factorization-based (MF) recommender systems, and two types of bias, namely, popularity bias and exposure bias. The focus on MF in no way minimizes the potential impact of this work since MF and latent factor methods are considered state of the art on their own, and further form a major building block of more complex models, including Deep Learning, where they are one foundation of representation learning.

1.2 Research Contributions

1. We propose a popularity bias correction matrix factorization algorithm (PCMF) that learns a popularity-aware embedding for users and items to capture the user bias towards different group of items, as well as a user-personalized bias vector to capture the level of conformity bias among different users. We integrate the popularity component in the learning process in order to learn more accurate and diverse embeddings. PCMF learns tailored or personalized recommendations that capture the user interest as well as her preference for popular items across different group of items. Our

approach is modular and can be adapted to recommendation algorithms other than MF.

2. We perform extensive experimentation to evaluate our proposed approach and find that PCMF outperforms other state of the art models in terms of recommendation accuracy, correcting for the popularity bias and increasing the diversity of the recommendations. Our model shows a better generalization ability when tested on intervened test data (where the test popularity distribution is different form the training data).

3. We propose a debiasing approach that corrects for two biases simultaneously. Specifically, we proposed an approach that mitigates exposure bias and popularity bias by combining the IPS debiasing framework to account for biases in the data collection step and our popularity correction matrix factorization (PCMF) to account for the model's popularity bias during the training. It is worth noting that to the best of our knowledge, our work is the first work that studies the effect of popularity bias and exposure bias simultaneously in the collaborative filtering recommendation system setting.

4. We perform extensive experiments to evaluate the performance of our algorithm and to test the effectiveness of accounting for two biases. Our experiments show that accounting for two biases further improves the recommendation accuracy and diversity, while decreasing the average popularity.

5. We propose a novel two-stage causal debiasing methodology, that is inspired from the proximal causal inference framework [20, 21]. Unlike a previous causal IPS approach [18] that corrects for *observed* confounders, our proposed approach corrects for *both* observed and potential *unobserved* confounders. First, we learn a bridge function to adjust the rating data for observed an unobserved confounders using a pair of negative control variables. Second, we fit an MF model on the adjusted ratings that are predicted by the learnt bridge function.

6. We evaluate our proposed two-stage causal debiasing methodology empirically using two unbiased real-world datasets and find that our proposed approach outperforms state of the art causal approaches, proving that accounting for unobserved confounders improves the recommendation system's performance.

1.3 Document Organization

The remainder of the dissertation is organized as follows. Chapter 2 presents a literature review about recommendation systems in general, followed by a review of previous research on bias, including efforts for studying bias and bias correction. In Chapter 3, we propose a new popularity correction matrix factorization (PCMF) approach as well as its experimental evaluation. In Chapter 4, we relax the main assumption made in Chapter 3 which is that the recommender system is affected by only popularity bias. We thus propose a new approach to correct for both exposure bias and popularity bias. Then we present an empirical study to test the importance of accounting for different biases simultaneously. Chapter 5 describes a new two-stage debiasing approach that corrects for observed and unobserved confounders inspired by proximal causal inference. Then we present experiments to evaluate the effectiveness of our causal debiasing approach. Finally, we conclude the dissertation in Chapter 6.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

As recommender systems (RS) continue to permeate through an expanding set of online applications and domains, they are starting to have an increasing effect on human decisions and behaviour. Hence, depending on the application domain and on their scale of use, biases that emerge within these recommender systems can have significant social and economical impacts [12]. For these reasons, research on understanding bias and designing debiasing strategies has been attracting a growing interest from the RS community [4].

We start by discussing RS by detailing the different methods used. We describe different families of models and learning algorithms used to recommend items to users. We then describe the different biases that can be present in a Collaborative Filtering recommender system. Finally, We highlight the shortcomings of previous research that has not dealt with multiple biases at the same time.

2.1 Recommendation Systems

In this section we describe the different techniques used in recommender systems. In our work, we mainly focus on Collaborative Filtering (CF) techniques, and so we start by describing the models used in this category of RS. Then we enumerate the models that are used in Content-based Filtering (CBF). Finally we briefly mention other types of recommender systems such as hybrid systems.

2.1.1 Collaborative filtering Methods

Collaborative Filtering is a family of recommender system models that predict future interactions by detecting similarity patterns between users and items. For instance, CF

models assume that similar users will have a similar rating behavior. On the other hand, similar items will be rated similarly by a given user. Therefore we notice that there are two families of CF models, user-based CF and item-based CF [23].

Collaborative Filtering only needs user-items interactions. It can predict the future interactions by only using past interactions of these users with the items. We explore a few classical models that fall into this category and we describe the learning algorithms used to build these models. We also study the correctness of the assumptions that make these methods work and how these relate to our work.

2.1.1.1 Matrix Factorization

Matrix Factorization [24] is a well known state of the art CF model. It became especially popular when it was part of the winning solution in the Netflix 1M$ prize challenge [25]. The aim of Matrix Factorization is to project users and items in to a shared latent space. Items that fall close to users in such a latent space will be then given a higher recommendation score. The distance between users and items is based on the cosine similarity since we use the dot product to map their latent variables.

To describe the formal learning objective function of Matrix Factorization, we adopt the following notation:

- U: Set of users
- I: Set of items
- u: A given user
- i: A given item
- P: Latent variable for the users
- Q: Latent variables for the items
- α: Learning rate
- β: Regularization hyperparameter

- K: Dimension of the latent space
- R: Rating matrix

The main goal of Matrix Factorization (MF) is to learn the latent variables P and Q where $P.Q^T$ reconstructs the original rating matrix R. In order to learn these representations, we minimize the standard loss function for MF, defined as follows:

$$J(P,Q) = \sum_{u \in U, i \in I} (R_{ui} - P_u Q_i^T)^2 \tag{2.1}$$

Equation 2.1 describes the mean squared error loss for Matrix Factorization. The idea is to learn a representation for each user and item by approximating the original rating matrix with the product of the representations/factors, through minimizing the mean squared error (MSE) of the reconstruction. In order to prevent the model from overfitting and failing to generalize to unseen data, we typically add a regularization term, as follows:

$$J(P,Q) = \sum_{u \in U, i \in I} (R_{ui} - P_u Q_i^T)^2 + \beta(||P||_2 + ||Q||_2) \tag{2.2}$$

By adding the regularization term, we make sure that our model does not reproduce the input data and thus end up failing to predict the missing ratings. β is a hyperparameter that is used to control the importance of the regularization.

To learn the model parameters P and Q, Alternate Least square is used by employing Stochastic Gradient Descent (SGD) [26] in an alternating fashion to update P and Q:

$$P_{t+1} = P_t - \alpha \frac{\partial J}{\partial P} \tag{2.3}$$

$$Q_{t+1} = Q_t - \alpha \frac{\partial J}{\partial Q} \tag{2.4}$$

Using Stochastic Gradient Descent (SGD) allows for parallelization [26] in the training phase which allows us to train scalable models. One of the drawbacks of Matrix Factorization is that it is not guaranteed to converge [27]. In fact, the function is non-convex. Therefore, the optimization algorithm may find a local minimum which does not reflect the global minimum of the problem.

Another drawback of Matrix Factorization is the cold start problem [28]. In fact, if a new user or a new item comes to the system, it will be hard to model it without existing interactions. Methods such as augmenting the embedding with external features have been used to mitigate this problem.

Furthermore, we notice that the main objective for MF is to reconstruct the rating matrix. It learns the preference of the user toward a given item but does not account for the ranking of items. This is solved using methods such as Bayesian Personalized Ranking [29].

2.1.1.2 Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) [29] is a method of learning that employs pairwise ranking along with a Bayesian framework. In fact, the loss employed learns user's preference by using pairs of items (i,j) where i is a positive item (the user likes the item and j is a negative item where the user is not interested in the item. The goal of the optimization problem is to learn the following relation $f_u(i) > f_u(j)$ where f_u is a preference function for user u that maps the items to a preference score.

The loss function used to learn such a property is the following:

$$J(u, i, j) = \sum_{u \in U, i, j \in I} \ln(\sigma(f_u(i) - f_u(j))) - \beta ||\Theta||_2 \tag{2.5}$$

σ is the sigmoid function and serves to map the user preference to a probability function. Θ represents the parameters of the model.

The next step is to model f_u. One way to do that is to use Matrix Factorization where $f_u(i) = P_u Q_i^T$. Other preference models can be used like Neural network architectures and other factorization models.

The advantage of BPR is that it directly optimizes ranking evaluation metrics while learning the user item preference. It has shown great improvement compared to other Collaborative Filtering methods.

One of the main drawbacks of BPR is that it needs to define positive items and negative items in the training phase. This might not be very trivial especially in the implicit feedback setting where the user item interactions are clicks. Therefore, negative

sampling is usually needed to sample negative items from the set of non interacted item. This may introduce bias in the system [30]. This can be mitigated by adopting different negative sampling strategies [31].

2.1.1.3 Nearest-Neighbors Methods

Nearest-Neighbors methods recommend items to users that are the closest to the previously interacted items based on a defined distance measure [32]. For instance, we can use the learned items' embedding of Matrix Factorization coupled with cosine similarity [33] in order to recommend items that are the closest (in terms of cosine distance) to the previously rated items by user u.

The Nearest-Neighbor method adopts an item-based Collaborative Filtering as it only uses the item embeddings during the prediction. However, the users are also used during the training of the embedding if we use a method such as Matrix Factorization.

Other models can be used to project the items in a space where we can calculate distances. These include Neural Networks where we learn an embedding space through multiple hidden layers. A naive approach can also be used by using the original rating matrix as a space for the items. One drawback of this method is that the space can have a very high dimensionality, therefore the distances between items will always be high (due to the curse of dimensionality [34])

2.1.1.4 Neural Network Based methods

Recommender systems have benefited from the advancement of the deep learning field where various architectures have been adapted in order to improve the recommendation quality [30].

One famous Neural Network architecture for Collaborative Filtering is Neural Matrix Factorization [35]. The idea is to learn two different representations for users and items using a Multi-Layer Perceptron layer and a Matrix Factorization layer. The two embeddings are then concatenated and fed into a sigmoid head to predict the score. The idea of Neural

Matrix Factorization NeuMF is to be able to detect the non-linear relation between the interaction of the user and item. Later work [36] has argued that using a dot product head in order to learn the similarity of the final concatenated embeddings yields better results.

Another family of methods for using a deep learning architecture is using sequential based architectures [37]. For instance, the recommendation problem can be seen as: Given a sequence of rated items $\{i\}_n$, we want to predict the next item to be rated/clicked. To this goal, models such as Recurrent Neural Nets (RNNs) have been used [38]. RNNs are a family of deep learning models that capture the sequential behavior of the interaction pattern of the user in order to provide next item recommendations. Later works have exploited Transformer architecture in order to leverage the self-attention framework for modelling sequential data [39]. Bert4Rec, for instance, models the interacted items as a sequence and uses a BERT [40] architecture to model the session of the user and provide recommendations.

An important family for Collaborative Filtering modelling, using Deep Learning networks, is Autoencoders (AE) [41]. AE tries to reconstruct the original input through an encoder-decoder architecture. The idea is similar to Matrix Factorization where the end goal is to represent the users and items through a shared latent space and then perform recommendations using various similarity measures [41]. Many variations of Autoencoders have been used such as Variational Autoencoders which adopt a Bayesian framework for the problem [42]. Autoencoder based architectures are being used in industry to model the items and users and is a growing field of research. A major drawback of this method is difficulty of convergence [43].

Graph Neural Networks can also be used in a Recommendation context [44]. The recommender system problem can be seen as a bipartite graph problem. Therefore, Graph Neural Networks (GNN) can be used in order to model the user profile and provide recommendations [45]. GNN can be used to model the embeddings of users and items by leveraging this graph relation. We put an emphasis in this case on the social influence for the user and the item similarity for the items. GNN can be extended to use more advanced graph based

networks such as Convolutional Graph Networks [46] or Graph Attention Networks [47].

A major issue for Collaborative Filtering is the need for interactions in order to work. Hence, users and items with no recorded interactions face the cold start problem. Models will not be able to learn correct embeddings for these entities. Therefore, one solution is to use Content-based Filtering methods where the use of external information is required.

2.1.2 Content-based Filtering Methods

Content-based Filtering methods rely on using external information about users and items in order to provide recommendations [48]. There are different ways of incorporating external features in a recommendation model. One way is to formulate the problem in a supervised learning setting as a classification problem. In this setting, the goal is to classify an interaction as a positive or a negative interaction (or no interaction). Therefore, users and items features can be used to achieve this result using typical classification algorithm. A popular class of algorithms are Decision Trees [49,50] through ensembles like Random Forest and Gradient Boosting Decision Trees (GBDT) [51]. These methods make use of sparse features for both items and users in order to provide recommendations. These methods can also be used beyond classification and a ranking model where pairwise and listwise loss functions are used in order to directly optimize ranking metrics. An example of such loss function is LambdaRank that uses a listwise strategy to predict ranking for items combined with Additive Trees context [52].

Another way to use external features is to compress them through an embedding space and store these embeddings to be used later for prediction. To achieve this, deep learning architectures are used. For instance Two tower architectures can be used in order to project the user features and the item features into an embedding space and then use a dot product head or a MLP head to provide predictions [53]. The advantage of using embedding approaches in order to represent user and item features is the ability to reduce computation cost by compressing the dimension of the problem. Also we can store the embeddings for fast inference at the serving stage. Furthermore, we can combine the

embeddings learned by using item and user features with the embeddings learned using Collaborative Filtering methods. This allows for hybrid methods that can over come the issues of both methods. Also embeddings allow for different ways of ranking and predictions in the inference stage. We can use Nearest Neighbor methods to provide similar items to the user. This method is preferred in industry because it scales well with a large number of items [54]. Implementation for scalable Nearest Neighbor search such as ANNOY [55] allows for fast retrieval.

An important issue of Content-based Filtering is the need to collect features from users which is expensive. Users usually do not enter a lot of information about themselves [56] and therefore, the collected features are very sparse and may lead to suboptimal results.

2.2 Bias in Recommender systems

Bias in recommender systems comes in many forms and from multiple sources, [14]. For instance, limited user exposure can create an exposure bias or selection bias that affects the model training and results in statistical bias. The feedback loop creates a filter bubble phenomenon which leads to several drawbacks, including popularity bias and non-diverse recommendations. To understand the bias effect of recommender systems, we provide a motivation on the bias phenomenon for different types of bias.

In a traditional recommender system setting, the user is first presented with a list of recommended items. Then, the user's clicks and ratings are used to retrain the model to dynamically capture the user preference. The traditional algorithms used, i.e. Matrix Factorization (MF) [57], assume that the collected data satisfies the independent and identically distributed (iid) assumption. This assumption does not hold in the real world for multiple reasons. For example, the items presented to the user are not randomly selected and the items do not have equal probability of being recommended to the user. On the other hand, the items presented to the user do not have the same probability of being observed. For instance, the user's observation is affected by multiple biases, i.e. presentation bias [5] [6], popularity bias [7], etc. Figure 2.1 shows a typical recommender system's life-cycle and

the biases that can occur at different stages therein [4]. Starting with the user interaction phase, users choose what items to interact with from the recommendation list, which leads to selection bias. In addition, the presentation of the items can affect the probability of user interactions. This presentation can consist of the position of the item in the list, the size of the item's icon, etc. The user interaction can also be affected by the item popularity and the crowd's opinion about an item [8]. The next stage is to log the user interactions with the recommendation list such as clicks, ratings, dwell time, etc. Then, the collected data is fed to a recommendation model to be retrained every t times. The popular items receive significantly more clicks than long tail items [58]. This leads to a biased training which is dominated by popular items, since these items contribute the most to the training loss. As a result, popular items have a higher probability of being recommended than long tail items despite their true relevance to the user. We will later study this empirically in the experimental evaluation section.

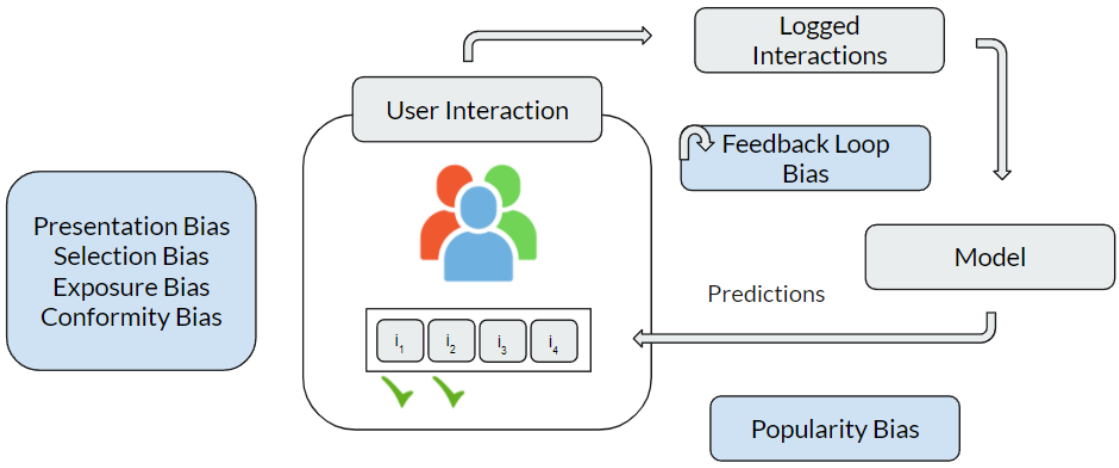


Figure 2.1. Recommender System Life-cycle and biases at different stages. Bias can be introduced in the data collection step by inheriting the user interaction biases. Bias can also be introduced in the model training phase, where popular items get recommended more frequently given that they receive a higher number of user interactions

To further investigate the effect of bias on the recommendation results, let us suppose we have a recommender system environment with k items, as shown in figure 2.2, where popular items are the square shaped items and long tail items are the oval shaped items.

The blue items are the items relevant to the user, and the solid line edges denote the items previously clicked by the user. In this example, we have three scenarios. The first scenario denotes the ideal case where all the relevant items have equal probability of being shown to the user despite their different levels of popularity. In the second scenario, we assume that popularity bias is the only bias affecting the data. Then, popular items have higher probability of being shown to the user at a higher position in the list. In this case, the items shown to the user have the same probability of being observed. Hence, after being presented to the user, popular and long tail items have equal probabilities of being observed despite their rank. In the third scenario, we relax this assumption. In fact, users tend to interact with items ranked higher in the list due to position bias [59]. As shown in figure 2.2, this leads to popular items receiving even more clicks as shown in the third scenario. In this section we detail different types of biases that affect the user experience while describing debiasing techniques used to counteract such biases.

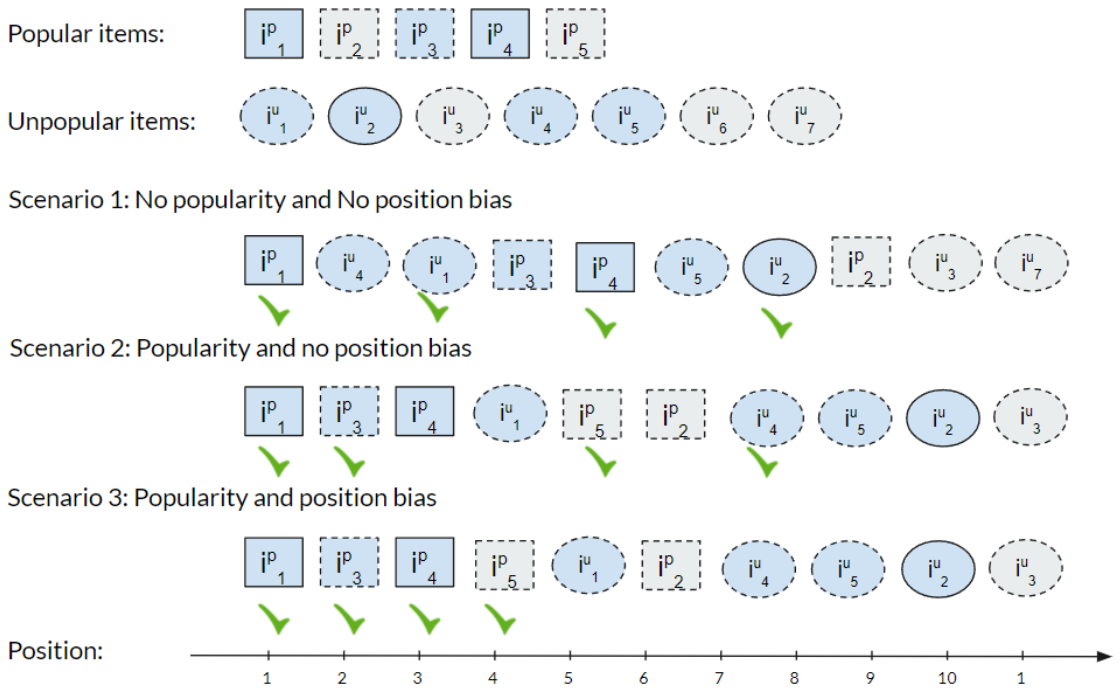


Figure 2.2. An example showing the effect of popularity and position bias by presenting the difference between the recommended list and the user interaction based on the existence of different biases. Blue items represent relevant items, solid-line items are those that have been previously clicked by the user. The tick sign represents the items clicked at the current iteration

2.2.1 Selection Bias

In contrast to classical machine learning systems where data is annotated using an expert to ensure unbiased labeling, RSs collect their data via the continuous interaction between the user and the RS algorithm. This particular setting leads to biased training data due to the selection bias of the user [60]. In fact, because the user can only interact with the exposed items which are a result of the past algorithm predictions, the labeling is conditioned on the previous recommendation iteration. Hence, the training data is confounded by other variables such as the user exposure [61], the positioning of the item in a ranked list, and the performance of the previous recommender system model. This bias is usually corrected mathematically by using Inverse Propensity Scoring methods [18, 62, 63]. IPS strategies eliminate the statistical bias in the training loss by accounting for the propensity of the users. This ensures an unbiased (in a mathematical sense) learning of the recommender system model.

2.2.2 Diversity Bias

A critical issue in RSs is redundant recommendations [64]. In fact, the inductive bias of the RS algorithms pushes for recommendations that are similar to the previously provided user interests, using some similarity measure. Examples of such algorithms are K Nearest Neighbors (KNN) and Matrix Factorization (MF) [57]. The use of these similarity measures helps to provide accurate but less diverse recommendations. This lack of diversity results in other problems such as the filter bubble and polarization [65], especially on large scale social platforms.

A popular solution to this problem is Multi-Armed Bandits strategies (MAB) [66–69]. MAB adds exploration to the recommendation process and therefore increases the diversity of the recommendations. The main intuition of the MAB strategy is to sacrifice some short-term reward in order to increase the long-term outcome. Different MAB strategies have been applied to the RS setting. A popular method is the ϵ-greedy strategy because of the simplicity of its implementation.

Recently, researchers have adopted Reinforcement Learning (RL) techniques in order to learn an "intelligent" exploitation exploration trade-off [70, 71]. However, Reinforcement Learning techniques still face criticism due to their expensive evaluation process of the proposed policies in an online setting and the logging bias introduced in an offline setting [4]. In order to correct for the logging bias, Inverse Propensity Scoring has also been proposed in this setting [72].

2.2.3 Popularity Bias

Items that get recommended more to users will collect more interactions, and this in return leads to a skewed rating distribution, with popularity bias [7] that affects future recommendations. In fact, the model has more ratings and hence insights about the popular items and this lead to more accurate recommendations for these items [73]. This loop will lead to even more disparity between the item popularity and creates a niche of underrepresented items, leading to an unfair exposure for some items. Similarly to solving the lack of diversity in a recommendation list, increasing the exposure of unpopular items may be achieved using post-processing such as re-ranking [74, 75] or exploration methods like Multi-Armed Bandits. Another family of solutions acts on the training loss by adding regularization to the error term. Other methods use multi-objective optimization through optimizing for both increasing the accuracy and reducing the popularity disparity [73].

2.2.4 Position Bias

Another major bias in recommender systems is position bias. In fact, the user is biased by the position of the presented item. Previous studies [76, 77] have shown that the user pays more attention to top ranked items and usually ignores the items that are ranked at the bottom of the list. For this reason, collecting data affected by position bias creates issues when evaluating and training future models [78]. In fact, some items will be considered as negative items because they were never clicked but the fact that they were not present in the top of the list plays a bigger factor on the decision of the user. Some

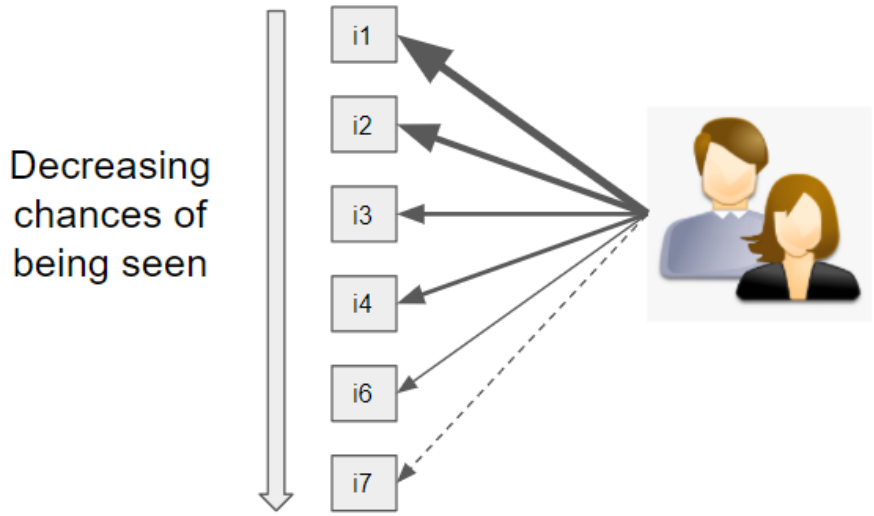


Figure 2.3. Illustrating the effect of position bias: The arrow width represents the probability of users seeing the ranked items. The position bias suggests that as the rank of the item increases in the recommendation list, it will have lower chances of being seen. This leads to relevant items not being clicked despite being in the recommendation list. This affects the collected data by producing biased data.

industries moved away from the ranked list strategy and are using other ways to represent items such as carousel or grid representation [79]. Other aspects of the position bias will still be present involving the user browsing behavior.

In order to mitigate position bias effect, several methods have been adopted. For instance, new click models have been developed [80–82] in order to take into account the causal effect of the position of the item using a multiplicative relation. In fact, [82] argues that the click of the model is an effect of first seeing the item and then being relevant. Seeing the item or being exposed to the item is directly related to the position of the item. Therefore, by constructing a click model that takes into account the position of the item, we can mitigate the effect of position bias. By tying the position of the item to the exposure of the user [83], we can also use Inverse Propensity Weighting in order to eliminate statistical bias during testing and training of the recommender system [63]. In fact, it can be shown that the exposure of the user that depends on the item's position creates a biased estimator that does not allow us to correctly evaluate the performance of the recommender system in

training and in testing.

Another way of counteracting position bias is by including such information in the training and removing it from the testing phase. This method is often used in industry [84] and in a content filtering scenario. The idea is to include the position as a feature in the training phase. Then, during the prediction phase, we assume that the position feature is constant (fixed as 1 for instance). Therefore, we eliminate the position bias effect by predicting the relevance of an item given it would be ranked in the first position.

2.2.5 Feedback Loop bias

Another important aspect of bias in recommender systems is feedback loops [10, 11]. In fact, due to the dynamic nature of recommender systems, most biases are amplified through feedback loops. In the first iteration, the user is exposed to some initial recommendations. They provide feedback to these recommendations. This feedback is affected by biases such as popularity, selection and position bias. The recorded data is then used by the next iteration of the algorithm in order to train and generate the next recommendations. This closed loop affects the evolution of the system and contributes to amplifying biases. In fact, previous research has shown that the feedback loop amplifies the bias and leads to multiple phenomena such as filter bubbles and echo chambers [15].

In order to mitigate the effect of the feedback loop, previous research has suggested using exploration methods such as bandit methods or random recommendations. Other methods tried deconvolution techniques [85] in order to understand how feedback loops affect the final ratings.

2.3 Recommender systems as a Causal Inference problem

Recommender systems can be modeled as a causal inference framework [86]. In fact, we can assume X as the treatment of the user which details that user has been exposed to a certain item by recommending that item to the user, and Y is the outcome which is the rating provided by the user. Therefore, there is a causal relation that can be infered which

is: What is the outcome of recommending an item i to the user u.

The problem with causal inference problems is the unmeasured confounders V. An unmeasured confounder is a variable that affects the exposure of the user to the treatment and the outcome Y but it cannot be measured. One example of an unmeasured confounder is popularity of the item [**?**], exposure bias [87], position bias [4], conformity bias [8].

Therefore, by using this framework, bias in recommender systems can be explained through causal graphs as proposed in [88] and as shown in Figure 2.4.

In fact, looking at Figure 2.4 (a), we can see the effect of selection bias where the user's rating is affected by which item is initially selected by the user. In fact, training on the observational data would create a skewed distribution of only selected items from all the observed items. This leads to biased training [18].

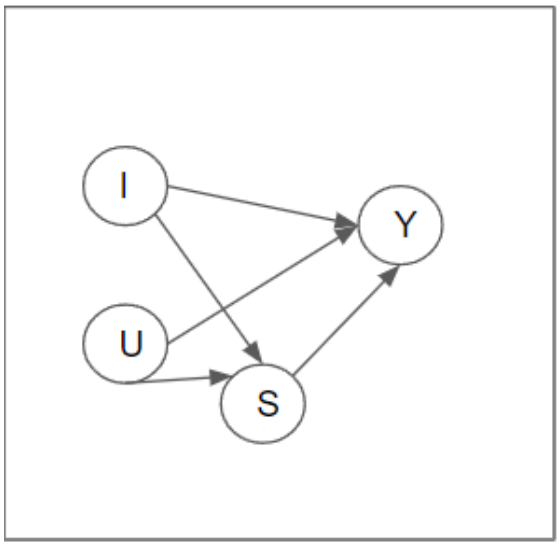
Looking at Figure 2.4 (b), we can illustrate the effect of conformity bias. In fact, users tend to be biased toward popular items and tend to rate these items higher than other items. Therefore, the popularity of an item affects the outcome Y (rating).

Figure 2.4 (c) shows the effect of position bias. In fact, position bias affects the exposure of the user. Items that are in higher position have higher chance of being selected [89]. Therefore, this selection affects which items are selected and which items are rated. Items that are highly rated also affects the next batch of recommendations where they have more chance of being recommended and therefore they appear in more high positions and this results in a feedback loop.

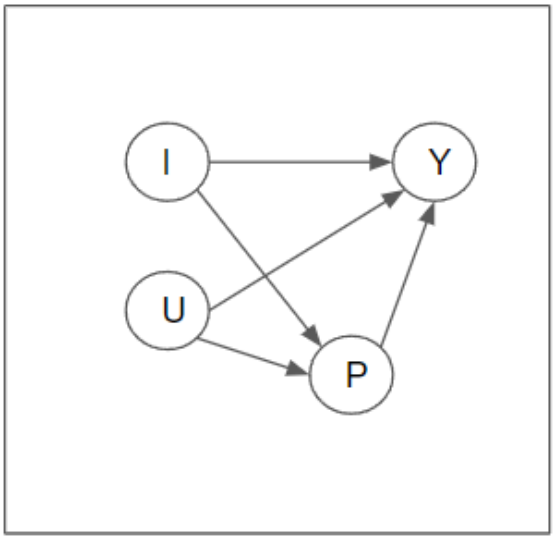
2.3.1 Causal Graph Interpretation of Recommendation System

As the bias problem can be drawn as a causal inference problem, the solution can be drawn from such framework. For instance, [18] have been used in order to account for selection bias in training and evaluation by proposing an inverse propensity scoring technique for loss functions. [86] proposed a framework for accounting for unmeasured confounders such as the exposure of the users.

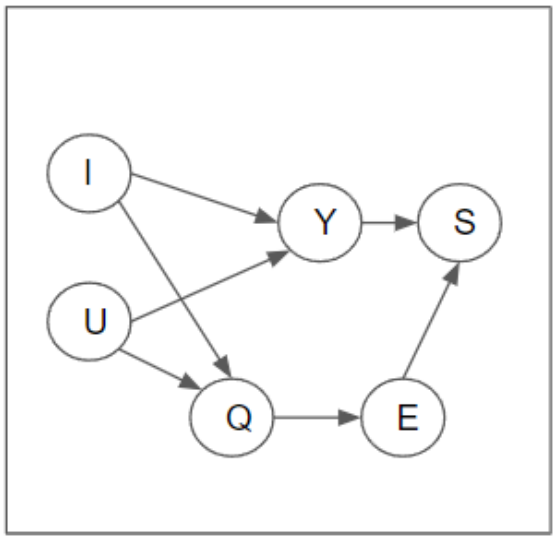
Causal approaches have also been used to counteract popularity bias [90–92], expo-



(a) Causal Graph for explaining the effect of selection bias



(b) Causal Graph explaining the effect of conformity bias bias



(c) Causal graph explaining the effect of position bias

Figure 2.4. **Causal graphs explaining different types of biases**. U,I denotes the user and item. S denotes the selection of the item, Y is the outcome or the rating, X is the treatment, P is the popularity, Q is the position and E is the exposure

sure bias [93,94] and unfairness in recommendations [95,96]. It has also been studied in the context of online learning and reinforcement learning based recommender systems where the aim is to evaluate the dynamic performance of the recommender system [97]

2.3.2 Limitation of previous bias research

A first limitation of the bias study in recommender systems is assuming that bias present in the dataset is always harmful to the user. We argue that this bias can be beneficial to certain users and we propose a personalized popularity bias by including a bias component as a Multi-Task Learning problem. This provides better accuracy results than previous approaches. Furthermore, bias in recommender systems has so far been studied in an isolated manner. In fact, researchers isolate a particular bias such as selection bias or popularity bias and then tailor a method to counteract this bias. Whereas, in real world recommender system, these biases occur simultaneously and affect each other. For instance, a user's choice in clicking an item is both affected by the popularity and the exposure of the user. Therefore, isolating these biases, although useful to gain a preliminary understanding of their effect on the recommender system, can fail to model a more truthful scenario of the practical use. We plan to tackle a multi-bias correction framework by combining the IPS framework with our proposed popularity correction model and we propose a novel causal inference framework in order to account for unmeasured confounders.

2.4 Summary

In this chapter, we summarized the different types of recommender systems. We started by explaining collaborative filtering recommender systems and discussed their shortcomings. Then we described content based filtering methods while briefly mentioning hybrid models. In the second part, we reviewed the literature on the problem of bias in recommender systems. We described the different types of biases in these systems ranging from popularity bias and exposure bias to selection bias and diversity bias. We also highlighted the limitations of previous bias research.

CHAPTER 3

POPULARITY CORRECTION MATRIX FACTORIZATION (PCMF)

In this chapter, we propose a new methodology to mitigate popularity bias. We start with the motivation and the problem description. Then, we present the notation and the problem formulation. Next, we present a new approach to popularity bias correction assuming that the data is affected only by popularity bias. Our method corrects for the bias by learning a personalized popularity bias component in both the item embedding and the user embedding. We finally present the results of an extensive experimental protocol to study the performance of the model.

3.1 Motivation and Problem Description

The goal of a recommendation system is to predict users' preferences in order to recommend relevant items. Recommendation systems are known to suffer from extreme popularity bias [16] [17], where popular items are recommended the most at the top of the recommendation list, resulting in imbalanced training data. The problem stems from the fact that popular items receive a higher number of clicks compared to long tail items. This leads to highly unbalanced training data and to a training process that is biased towards popular items. The fact that popular items receive more clicks does not imply that all users are interested in popular items. In fact, the total number of interactions is skewed towards popular items, but for a specific user, the percentage of popular items that the user interacted with tends to vary from one user to another [16]. In addition, the majority of traditional recommendation systems do not capture user-level conformity bias. In this context, conformity bias stands for the user's tendency to rate or like popular items in a similar way to the majority of users, or possibly to align with the public opinion.

For instance, [16] showed that the recommendation lists provided by Matrix Factorization consist mainly of popular items, regardless of the user preference for popular items or long tail items. This shows the disparity between the user's conformity level and what the recommender is delivering, which may hurt the user experience.

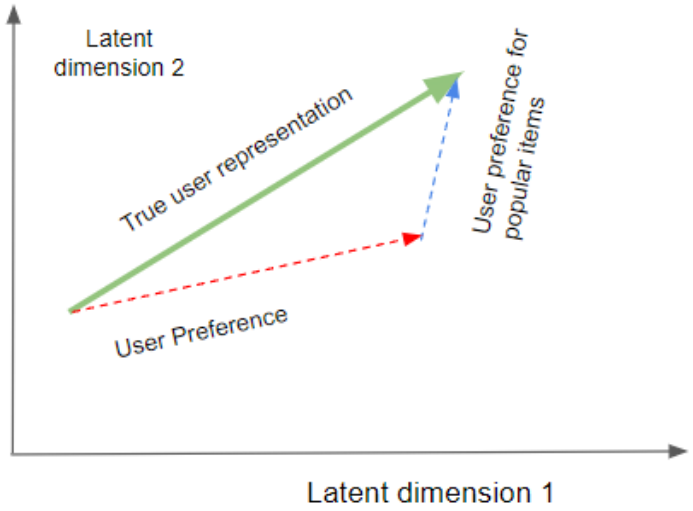


Figure 3.1. Decomposition of the true user representation into two main components. The first component is User Preference which represents the user's preference toward all items. The second component (blue vector) is the user preference for popular items.

The users' preferences for popular items vary from one user to the other. In fact, users can have different levels of conformity bias. One user may have a higher preference for popular items and may have a more agreeable personality. In this case, the popularity bias actually benefits the user since it reflects her true interest in popular items. Other users may prefer long tail items or niche items, in this case, recommending popular items may degrade their user experience [98, 99].

Figure 3.1 shows a decomposition of the user preference in a two dimensional latent space. We show that the true representation can be decomposed into a general preference vector and a bias vector that represents the degree that the user likes popular items. Learning only the general preference of the user, or counteracting the popularity bias without taking into account the relevance, will neglect an important component of the popularity bias and might hurt the performance of the model. On another hand, ignoring the pop-

ularity bias of the system will underfit the general preference of the user and will learn a representation that is biased toward the popularity bias component. In addition to that, conformity bias does not vary only between users, it also varies for the same user depending on the items' categories. For example, a user may be interested in documentaries and keen about finding "hidden gems" in this category. On the other hand, when it comes to romantic movies, she enjoys watching popular ones.

Previous work [8] [100] [101] has mainly focused on removing popularity bias without focusing on the users' possible varying preference for certain popular items. One of the most popular methods is inverse propensity scoring [100] which re-weights the items during the training based on their popularity. Long tail items weights are higher than popular items. The latter approach treats the popularity bias from the *items'* perspective. In this work, we look at the popularity bias from the *users'* perspective. Other studies [8] [101] focused on removing popularity bias by learning causal embeddings that are able to detect the underling cause of clicks. In those studies, popularity bias was considered harmful and the ultimate goal was to remove it. In our work, we advocate that popularity bias can be beneficial when studied from the *user*'s perspective. For example, if the user has a high conformity bias and enjoys popular items, recommending popular items to them would provide a better user experience. Our work focuses on detecting nuance in the user's conformity bias within item categories as well. Our goal is to improve the recommendation accuracy and to correct for popularity bias with respect to users' preference.

Motivated by the difference between users' conformity levels, we propose the Popularity Correction Matrix Factorization (PCMF) algorithm that learns the user's interest by disentangling conformity and relevance using a personalized multi-objective framework. This work falls into mitigating popularity bias during the recommendation stage.

Our main contributions are the following:

- We propose a new popularity bias correction matrix factorization method (PCMF) that learns a popularity-aware embedding for users and items to capture the user's bias towards different group of items, as well as a user-personalized bias vector to cap-

ture the level of conformity bias among different users. We integrate the popularity component in the learning process in order to learn more accurate and diverse embeddings. PCMF learns tailored or personalized recommendations that capture the user interest as well as her preference for popular items across different group of items. Our approach is modular and thus can be adapted to recommendation algorithms other than MF.

- We performed an extensive empirical experiments to test our proposed approach. Our results show that PCMF outperformed state of the art models in terms of recommendation accuracy, correcting for the popularity bias, and increasing the diversity of the recommendations. Our model showed a better generalization ability when tested on intervened test data (i.e., the popularity distribution in the test data is different form the training data).

3.2 Popularity Bias Correction Matrix Factorization (PCMF)

In this section, we present a detailed description of each component of our proposed Popularity Correction Matrix Factorization. The goal is to provide the user with accurate recommendations and to provide a personalized correction for the popularity bias for each user. Figure 3.2 shows an overview of the model architecture. PCMF is composed of 3 main components:

- The first module is responsible for learning the user's relevance embedding. The true ratings are used to learn the user interest by learning user and item embeddings. This module recovers classic Matrix Factorization.

- The second module is responsible for learning the conformity embedding. The popularity matrix is used to learn the embedding. The popularity matrix consists of the popularity of each item that the user has rated. The embedding will capture the user's preference for popular items across item categories.

- The predicted ratings are obtained using a linear combination of the item's popularity

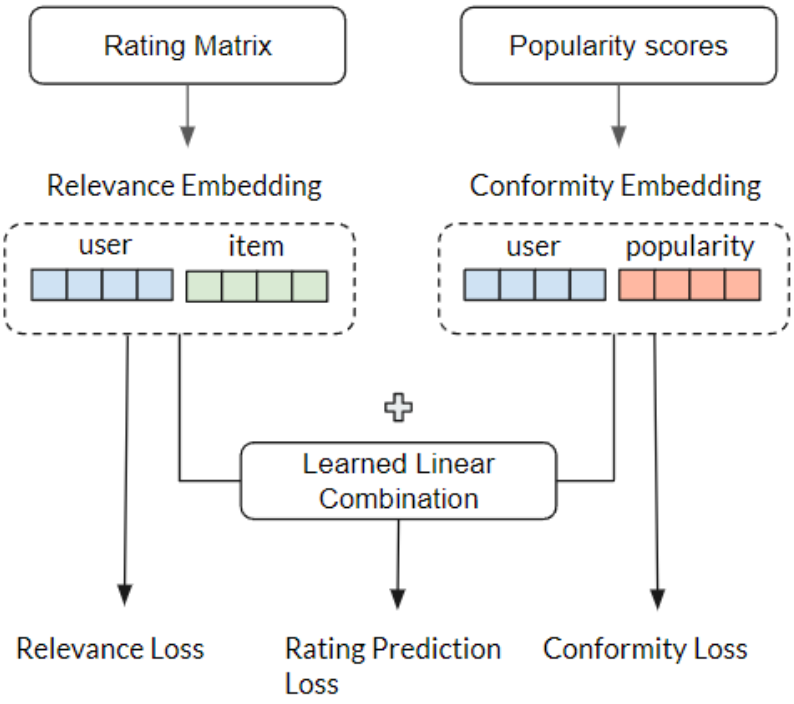


Figure 3.2. Popularity Bias Correction Matrix Factorization (PCMF). The first module denotes the relevance embedding. The second module denotes the conformity embedding, i.e., it is learned from the popularity matrix. The linear combination of the relevance score and the conformity score constructs the predicted rating. Each task has its corresponding loss function. A multi-task learning framework is used to predict ratings from the relevance and conformity embeddings

and relevance. We assume that the users' interactions with a popular item may be motivated by the users' true interest in the item or the other users' opinion about the item. The idea here is to disentangle the conformity and interest, whereas in classic matrix factorization, both entities are entangled. Given that the preference for popularity varies among users, we learn a personalized linear combination for each user.

- We use a multi-task learning framework to find a trade-off between learning the conformity embedding, the relevance embedding and the personalized bias correction vector.

In the following sections, we will describe each component of the PCMF architecture

in detail. Finally, we will describe our experimental protocol and present our empirical results on two real world datasets.

3.2.1 Notation and Problem Formulation

We start by defining the notation for a recommender system with explicit feedback data (i.e., ratings). Let U be set of the users with $m = |U|$ being the number of users, and let I be the set of items where $n = |I|$ is the number of items. Let R be the rating matrix where each entry r_{ui} represents the rating assigned by user u to item i. An entry $r_{ui} > 0$ indicates that user u rated item i. An entry $r_{ui} = 0$ does not necessary mean that user u is not interested in item i. It means that the item has not been observed. We introduce a matrix $O_{u,i} \in \{0, 1\}$ where each entry denotes if user u has been exposed to item i. $O_{u,i} = 1$ means that the user has been exposed to item i, whereas $O_{u,i} = 0$ means that the user has not been exposed to item i. Each entry $O_{u,i}$ is a Bernoulli random variable indicating whether a user u has been exposed to an item i.

3.2.2 Learning Conformity Embedding

In this section, we discuss how to learn the nuances of the user's conformity bias across diverse item groups. For example, in an e-commerce setting, a user may have a different sense of fashion and tend to like non-popular items. On the other hand, the user may like to purchase popular items when it comes to electronics. The user tendency towards clicking on popular items may vary depending on the item group. In fact, if the user's previous interactions consist mainly of long tail items, we can infer that the user likes to seek out niche items within a certain group. On the other hand, if the user clicks prominently on popular items, we can infer that the user has a preference for popular items and has a high level of conformity bias. Let Pop be the popularity matrix of size $(m = |U|, n = |I|)$, where each entry $pop_{u,i}$ represents the popularity of an item i clicked by user u. The entity $pop_{u,i}$ can be computed using the following equation 3.1:

$$pop_{u,i} = \sum_{u \in U} \mathbb{1}_{R_{u,i} > 0} \tag{3.1}$$

Notation	Definition
U	Set of users
m	Number of users $\|U\|$
I	Set of items to be recommended
n	Number of items $\|I\|$
O	Binary Observation matrix indicating if user u observed item i
R	Rating matrix where $r_{u,i}$ is the rating of user u to item i
$\widehat{R}$	Predicted rating matrix
Pop	Popularity matrix where $pop_{u,i}$ is the popularity of item i clicked by user u
$\widehat{Pop}$	Predicted popularity matrix
S	User conformity embedding
V	Item popularity embedding
P	User embedding
Q	Item embedding
α_u, β_u	Personalized bias parameter vectors
γ	L2-Regularization weights
λ	Hyperparameter controlling the personalized linear combination between relevance and conformity
η	Learning rate
I_u	Recommended list to user u

TABLE 3.1

Notation and Definitions - Chapter 3

The popularity of an item is defined as the number of interactions the item gets from all the users. The popularity scores $pop_{u,i}$ are then bucketed into five intervals using their percentiles $[per_1, per_2, per_3, per_4, per_5]$. An index of 1 denotes the less popular items and an index of 5 denotes the most popular items. Let f be a mapping function where $f : pop_{u,i} \rightarrow \{1, 2, 3, 4, 5\}$. We suggest learning embeddings that capture the user's conformity spectrum. The idea is to project users that have similar levels of conformity for a certain group of items closer to each other in the embedding space. For instance, the users that have a preference for niche items within several categories will be considered similar even though they might have different interests. For example one may have a preference for horror movies and

the other may have a preference for romantic movies. However, if two users have similar preference for niche items within the same group of items, then they will be closer in the embedding space compared to the latter example. This might sound counter-intuitive to the recommender system goal. In the next section, we will describe how to capture the user interest as well. In this section, we want to capture the user's conformity bias levels for different item groups. In order to learn the conformity embedding, we propose to use matrix factorization where the input is the popularity matrix Pop. Equation 3.2 gives the loss function to learn the popularity embedding:

$$L_{Conformity}(\widehat{Pop}) = \frac{1}{|\{(u,i) : O_{u,i} = 1\}|} \sum_{(u,i):O_{u,i}=1} (pop_{u,i} - s_u.v_i^T)^2 + \gamma(\|s_u\|^2 + \|v_i\|^2) \quad (3.2)$$

where S represents the users' conformity embedding, V represents the items' similarity embedding in terms of popularity. $\|s_u\|^2$, $\|v_i\|^2$ denote the regularization terms which are weighted by parameter γ to control for overfitting. Representing the user's conformity using an embedding is more accurate than using simple scalar scores, as was done by previous studies [100] [9] which mitigated popularity bias by re-weighting the observed rating using Inverse Propensity Scoring (IPS). This is because the IPS scores are user-independent and rely only on the item popularity scores. In other words, they assume that all users have the same behavior towards popular items and the same level of conformity bias towards all items. A *learned embedding* provides a richer representation of the user's conformity bias spectrum and enables the system to capture the different nuances of user conformity.

3.2.3 Disentangling Conformity and Relevance

The user clicks are influenced by popularity and true relevance. A good recommender system should be able to capture the general interest of the user. Motivated by the fact that user clicks are influenced by conformity and interest, we propose that the rating can be decomposed into an entity quantifying the true relevance and an other entity quantifying the influence of the popularity on the user's click. In order to do this, we represent the predicted rating $\widehat{r}_{u,i}$ as the linear combination of a relevance score $\widehat{rel}_{u,i}$ and a conformity score $\widehat{pop}_{u,i}$, as shown in equation 3.3. These terms are further weighted by parameters α_u

and γ_u. Which are **learnable** parameters that personalize the degree of a user's conformity bias toward popular items.

$$\begin{aligned} \widehat{r}_{u,i} &= \beta_u \widehat{rel}_{u,i} + \alpha_u \widehat{pop}_{u,i} \\ \widehat{rel}_{u,i} &= p_u . q_i^T \\ \widehat{pop}_{u,i} &= s_u . v_i^T \end{aligned} \tag{3.3}$$

The conformity score $s_u.v_i^T$ is computed using the conformity embedding from the previous section. Let us suppose that a user has a preference for long tail items, which would mean a low conformity score $s_u.v_i^T$. The interaction is more probably a representation of her true interest in the item. Thus, the relevance score $p_u.q_i^T$ would be high. Equation 3.4 shows a loss function to capture the relevance:

$$L_{relevance}(\widehat{R}) = \frac{1}{|\{(u,i) : O_{u,i} = 1\}|} \sum_{(u,i):O_{u,i}=1} [r_{u,i} - (\beta_u p_u . q_i^T + \alpha_u s_u . v_i^T)]^2 + \gamma(\|p_u\|^2 + \|q_i\|^2) \tag{3.4}$$

Where R is the rating matrix, s_u and v_i are the conformity embeddings from the previous section, α_u, β_u are **personalized** trade-off parameter controlling the relative contribution of the relevance component $p_u.q_i^T$ and the popularity component $s_u.v_i^T$ for each user. The parameter γ controls the L2-regularization terms $\|p_u\|$ and $\|q_i\|$. In the inference phase, the relevance of the items and the user-specific conformity bias are used to compute the final recommendations. This means that the items are recommended in descending order of their $\hat{r_{u,i}}$, calculated using Eq. 3.3.

3.2.4 Multi-task Learning

During the learning phase, our goal is to learn to predict the rating $\widehat{R}$ matrix. In this case, we have two tasks to optimize: learning the conformity embedding and learning the true relevance. The two tasks will be learnt simultaneously since they mutually improve each other. The parameter λ controls for the trade-off between learning the conformity embedding and the relevance embedding, $\lambda \in [0, 1]$. The loss function can formally be

written as follows:

$$L_{clicks}(\widehat{R}, \widehat{Pop}) = L_{relevance}(\widehat{R}) + \lambda L_{conformity}(\widehat{Pop}) \tag{3.5}$$

λ can be seen as a parameter controlling for the importance of conformity bias. The conformity bias importance could depend on the application domain, such as book recommendation or movie recommendation. λ equals 1 means the predicted rating is based only on conformity bias. When λ equals 0, the algorithm is equivalent to traditional matrix factorization (MF). For the learning step, Stochastic Gradient Descent can be used to search for the optimal parameters. The update equations to compute the embeddings for relevance and conformity, and to learn the personalized bias weights, can be derived to result in the following.

$$p_u^{(t+1)} \leftarrow p_u^{(t)} - \eta[-2\beta_u^{(t)}(q_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| p_u^{(t)} \right\|] \tag{3.6}$$

$$q_i^{(t+1)} \leftarrow q_i^{(t)} - \eta[-2\beta_u^{(t)}p_u^{(t)}(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| q_i^{(t)} \right\|] \tag{3.7}$$

$$s_u^{(t+1)} \leftarrow s_u^{(t)} - \eta[-2\lambda(v_i^{(t)})^T(pop_{u,i} - \widehat{pop}_{u,i}^{(t)}) - 2\alpha_u^{(t)}(v_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| s_u^{(t)} \right\|] \tag{3.8}$$

$$v_i^{(t+1)} \leftarrow v_i^{(t)} - \eta[-2\lambda s_i^{(t)}(pop_{u,i} - \widehat{pop}_{u,i}^{(t)}) - 2\alpha_u^{(t)}s_i^{(t)}(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| v_i^{(t)} \right\|] \tag{3.9}$$

$$\alpha_u^{(t+1)} \leftarrow \alpha_u^{(t)} - \eta[-2s_i^{(t)}(v_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)})] \tag{3.10}$$

$$\beta_u^{(t+1)} \leftarrow \beta_u^{(t)} - \eta[-2p_i^{(t)}(q_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)})] \tag{3.11}$$

Algorithm 3.1 describes the steps of the PCMF approach. To reduce the number of parameter of our model and mitigate the risk of overfitting, we use the relation $\alpha_u = 1 - \beta_u$ in our model implementation.

3.3 Experimental Results

In this chapter, we will test the performance of our Popularity bias Correction Matrix Factorization (PCMF) algorithm. We test our method on real-world datasets and compare its performance to widely used baselines. We design our experiments to evaluate the effectiveness of our proposed approaches at providing accurate recommendation as well as correcting for popularity bias. The research questions we aim to answer are:

Algorithm 3.1 Popularity bias Correction Matrix Factorization (PCMF)

Input: Rating matrix $R : \{r_{u,i} : (u,i) \in O\}$, Popularity matrix $Pop : \{pop_{u,i} : (u,i) \in O\}$, Regularization parameter γ, Trade-off parameter λ
Output: User Conformity latent factors $S_{1:U}$, $V_{1:I}$, User relevance latent factors $P_{1:U}$, $Q_{1:I}$, Personalized bias vector α_u, β_u

```
Randomly initialize S_{1:U}, V_{1:I}, P_{1:U}, Q_{1:I}, α_u, β_u
while not converged do
  for u ← 1 : U do
    Update S_{1:U}, P_{1:U} using Eq 3.8 & Eq 3.6
    Update α_{1:U}, β_{1:U} using Eq 3.10 & Eq 3.11
  end for
  for i ← 1 : I do
    Update V_{1:I}, Q_{1:I} using Eq 3.9 & Eq 3.7
  end for
end while
return S_{1:U}, P_{1:U}, V_{1:I}, Q_{1:I}
```

- **RQ1**: How does our proposed approach **PCMF** compare to other state-of the art methods in terms of performance on predicting recommendation relevance?
- **RQ2**: How does our proposed approach **PCMF** compare to other state-of the art methods in terms of promoting diverse non-popular items?
- **RQ3**: How does our proposed approach **PCMF** generalize on test data with different popularity distributions?
- **RQ4**: What do the relevance and popularity embeddings learn in the latent space?
- **RQ5**: What is the role of the personalized bias parameter vector α_u in predicting relevant and diverse recommendations?
- **RQ6**: What is the role of the added conformity embedding loss and how does the Relevance-Conformity tradeoff parameter λ affect the recommendation performance?

We start by describing the experimental setting including the data used, the metrics, the baselines and the experimental set-up. Then, we will present the performance evaluation of our popularity bias correction approach in order to answer the research questions **RQ1**-**RQ6**. We start with an empirical evaluation of the popularity bias severity in recommendation systems, specifically matrix factorization (MF). To do so, we start by grouping

the items into five buckets based on their popularity using percentiles. We predict the percentage of the items in each group by dividing by the total number of items. Then, we count the average frequency of each group in the recommended lists provided by traditional Matrix Factorization (MF) using the Movielens 100k dataset. Figure 3.3 shows that MF suffers from a severe popularity bias. Group 5 represents the most popular items. Even though items belonging to group 5 are less than 1% of the data, they are recommended the most frequently. On average, 50% of the top 20 recommendation lists are popular items. On the other hand, long tail items, representing more than 60% (i.e. the majority) of the data, appear in less than 1% in the recommendation lists.

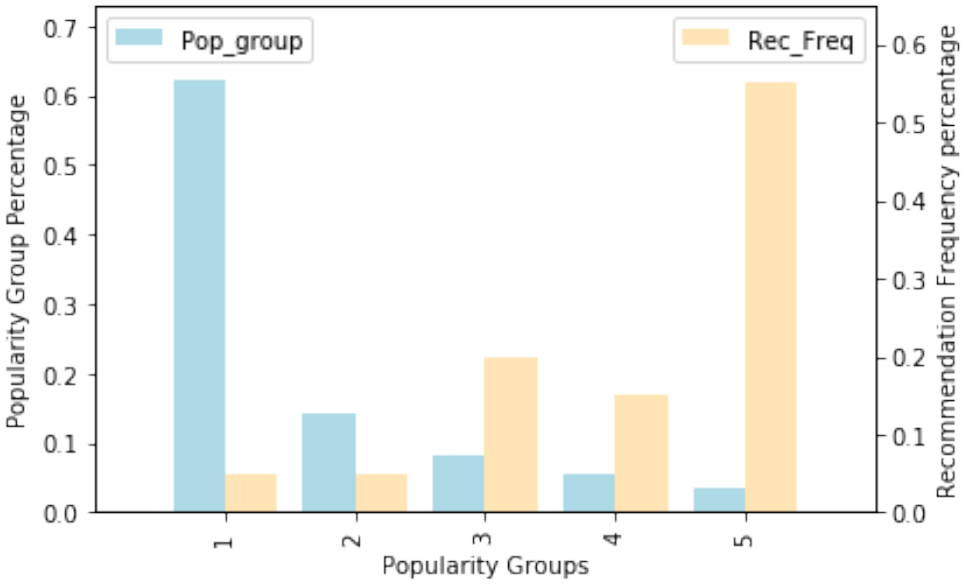


Figure 3.3. Items in the Movielens 100k dataset are divided into five groups depending on their popularity level using percentiles. The least popular items belong to group 1 and the most popular items belong to group 5. The blue bars represent the percentage of each group in the data. As expected, long tail items represents the majority of the data and popular items represents a small fraction of the data. The yellow bars represent the average percentage of each group in the recommendation lists of size 20 computed by Matrix Factorization (MF). Popular items are recommended the most despite being less than 1% of the data. This graph shows that MF suffers from a severe popularity bias.

3.3.1 Experimental Setting

3.3.1.1 Datasets:

We use two publicly available datasets in order to evaluate the performance of our models.

- **Movielens-100K**[1]**:** A movie rating dataset with ratings from 1 to 5. The data includes around 100k ratings collected from 943 users for 1682 movies. The data sparsity is 95%.
- **Movielens-1M**[2]**:** A larger version of the Movielens 100K with 1M ratings, 6000 users, and 4000 items. We use it to evaluate the applicability of the model on larger datasets.

Table 3.2 shows the different statistics of the used datasets. We filtered out users that have less 20 interactions which allows for reducing the sparsity of the data and learning better embeddings.

TABLE 3.2

Dataset Statistics

Data	Users	Items	Interactions	Sparsity
Movielens-100K	943	1682	100k	93.7%
Movielens-1M	6000	4000	1M	95.7%

3.3.1.2 Metrics:

Our goal is to measure the performance of our model on three tasks: the rating prediction performance, the recommendation performance, and the ability of the model to recommend diverse items, i.e. less popular items. In order to evaluate the prediction performance of the model, we used the Root Mean Square Error (RMSE) defined in equation 3.12, where N is the number of ratings in the test set. The **lower** the RMSE, the **better**.

[1] `https://grouplens.org/datasets/movielens/100k/`
[2] `https://grouplens.org/datasets/movielens/1m/`

$$RMSE(R, \widehat{R}) = \sqrt{\frac{\sum \left(r_{u,i} - \widehat{r}_{u,i}\right)^2}{N}} \tag{3.12}$$

In order to evaluate the quality of the recommended list ranking, we will use the following metrics: Normalized Discounted Cumulative Gain (NDCG@K), Precision@k, and Recall@K. We compute NDCG@K using equation 3.13:

$$\begin{aligned} NDCG@K &= \frac{DCG@K}{IDCG@K} \\ DCG@K &= \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{log_2(i+1)} \end{aligned} \tag{3.13}$$

IDCG@K stands for the DCG@K for the ideal ranking. K stands for the length of the recommended list and i is item rank. The NDCG penalizes relevant items being ranked lower in the list. Hence, the **higher** the NDCG@K is the **better** the ranking is. Precision@K is computed using 3.14:

$$Precision@K = \frac{|\{relevant\ items\ @K\} \bigcap \{retrieved\ items\ @K\}|}{|K|} \tag{3.14}$$

Precision@K estimates how many relevant items are in the recommendation list of length K. The **higher** the precision the **better** the ranking is. Recall@K is the fraction of relevant items that have been retrieved in the list of length K. The **higher** the recall the **better** the ranking is

$$Recall@K = \frac{|\{relevant\ items\ @K\} \bigcap \{retrieved\ items\ @K\}|}{|\{relevant\ items\}|} \tag{3.15}$$

To evaluate the popularity in the recommended lists, we use the Average Popularity. pop_i in equation 3.16 refers to the popularity of item i computed using equation 3.1. Our goal is to reduce the number of popular items in I_u. Hence, the **lower** the average popularity, the **better** the model is at reducing popularity bias.

$$Avg\_Pop = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|I_u|} \sum_{i \in I_u} pop_i \tag{3.16}$$

We user Intra-list Diversity (ILD) [102] to measure the diversity within the recommended list. The individual intra-list diversity is computed as the average of the pairwise distances between items belonging to the recommended list I_u. Individual ILD is used to measure the diversity within a recommended list for user u and to measure the novelty as well. ILD is computed using equation 3.17, where dist stands for the cosine based distance (1 - cosine). The **higher** the IDL is the **better** the recommendation list is in terms of diversity.

$$ILD = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|I_u|(1 - |I_u|)} \sum_{(i_k, i_j) \in I_u} dist(i_k, i_j) \tag{3.17}$$

We used the minimum intra-list diversity (Min-ILD) as well, which is the minimum individual intra-list diversity computed using equation **??**.

$$MinILD = min_{u \in U} \left\{ \sum_{(i_k, i_j) \in I_u} dist(i_k, i_j), \right\} \tag{3.18}$$

3.3.1.3 Baselines:

We compare our method to four collaborative filtering approaches: ItemPop [103], Matrix Factorization (MF) [57], Matrix Factorization with Inverse Propensity weights (IPS-MF) [100], and Multi Armed Bandits (MAB) [104].

- **ItemPop** [103]: ItemPop is a non-personalized recommendation method that recommends to the user the most popular item. We use it as a benchmark for performance.
- **Matrix Factorization (MF)** [57]: Matrix factorization is a popular CF technique that learns an embedding of users and items by decomposing the rating matrix into two embedding vectors. It uses the dot product to map the embedding to predicted ratings.
- **IPS-MF** [100]: Inverse propensity Matrix factorization is an unbiased MF model where the observed ratings for popular items are down-weighted since they have higher probability of being observed. Our goal here is to compare simple scalar based (score)

debiasing to embeddings that capture the user's personalized tendency to interact with popular items.

- **Multi Armed Bandits (MAB)** [104]: Multi-Armed Bandits (MAB) is a popular post-processing method that introduces exploration to the final recommendations by introducing noise into the dataset.

3.3.1.4 Experimental set-up:

In this section, we describe our experimental set-up. We start with the data splitting process. Then, we describe the data pre-processing procedure. Finally, we describe the hyper-parameter tuning process and present the hyper-parameters used to train Popularity Correction Matrix Factorization (PCMF).

Data splitting: The ideal set-up to evaluate the model's ability to remove bias is to test it on an unbiased dataset. In this case, all the items in the test set should have a uniform probability of being observed by the user. Collecting the test data in real-life is challenging. For instance, providing users with random recommendations will typically degrade the user experience. Similar to [8] [9], we created an intervened dataset to simulate the latter scenario. We called the test data in this scenario the **Skew** data. For a better evaluation, We will test our model in two settings as explained below. The first setting, called the **regular** setting, is the traditional recommender system evaluation set-up. The second setting is the intervened setting, using the Skew test data.

- **Regular**: The data is randomly split into 70% training, 10%validation, and 20% test. Note that in this setting the training, validation, and test data have the same popularity distribution.

- **Skew**: The model is evaluated on a skewed test data, simulating the random-off policy. We started by constructing our test and validation data by sampling 30% of the full data where each item has a probability of the inverse popularity of being selected. This means highly popular items have a lower chance of being selected in the test

data. For instance, if an item is 10 times more popular than the least popular item, then it would have a 0.1 probability of being selected in the test data. The popularity of an item pop_i is the number of times an item has been rated. Let $P(i \in T)$ be the probability of item i being selected in the test data. Equation 3.19 shows how to compute $P(i \in T)$.

$$\begin{aligned} pop_i &= \sum_{u \in U} \mathbb{1}_{O_{u,i}>0} \\ P(i \in T) &= \frac{min_{j \in I}(pop_j)}{pop_i} \end{aligned} \tag{3.19}$$

In order to limit the number of new items in the test data, we cap the probability of an item being selected in the test set to 0.9. In other words, the least popular items that have $P(i \in T) > 0.9$ are set to $P(i \in T) = 0.9$. So, the least popular items are present in the training data as well. Note that the popularity distribution in the training data is different from that in the validation and test data. The remaining 70% of the data represent the training data.

Data Prepossessing: The popularity matrix, as defined by equation 3.1, denotes the average item popularity computed by counting the number of interactions each item gets. Note that the range of the popularity scores is different from the rating range which may cause the loss function to be biased towards the popularity embedding reconstruction. To avoid this problem which risks leading to a vanishing contribution of the rating reconstruction loss, we split the popularity scores into five buckets based on their percentiles $[per_1, per_2, per_3, per_4, per_5]$. Then, the least popular item will have a score 1 and the most popular item will have a score 5. This results in *scaled* popularity scores which range between 1 and 5 just like the ratings. As a result, the rating and popularity loss components will end up contributing equally to the combined loss.

Hyper-parameter tuning: Hyper-parameter tuning for all baselines was done using Optuna [105] employing a Tree-structured Parzen Estimator as a sampler. The hyper-parameter tuning was done using 10% validation data. The hyper-parameters to tune

are: the relevance embedding's latent dimensionality, the conformity embedding's latent dimensionality, the weight parameter λ, the gradient update's learning rate, and the L2 regularization weight. We varied the learning rate between [0.001, 0.1]. The embedding dimension sizes varied in {8, 16, 32, 64, 128} for the relevance and conformity embedding, separately. This means that each embedding can have a different dimensionality. We varied the trade-off parameter λ in [0.1, 1] and the L2 regularization weight in [0.0001, 0.1]. The baselines are tuned using the same strategy for a fair comparison. For the optimizer, we chose between Adam [106] and the Adaptive Gradient Algorithm (Adagrad) [107]. We varied the batch size in {64, 128, 256, 512}. We selected the hyper-parameters based on the evolution of root mean square error (RMSE) on the validation dataset. Note that each of the baselines was trained using the hyper-parameters providing its own best results.

3.3.2 Performance Evaluation of PCMF in the Regular Setting

In this section, we will answer the first two research questions **RQ1** and **RQ2**. We report PCMF's performance results on the Movielens-100k and Movielens-1M data sets in the regular setting (where the data is used as is and not skewed to reflect bias). The convergence of the algorithm is declared when the RMSE on the validation set does not decrease for five epochs. We run each experiment 5 times and we report the mean and the standard deviation. First, we evaluate the model on the recommendation performance which includes the rating prediction accuracy and the ranking quality. Second, we evaluate the model on the popularity bias correction task. The values in bold denote a significant improvement ($p - values < 0.05$).

RQ1: How does our proposed approach compare to other state-of the art methods in terms of recommendation relevance performance? To answer this question, we investigate the recommendation accuracy metrics on two real world datasets, Movielens-100K and Movielens-1M, as shown in Table 3.3 and Table 3.4, respectively.

- **Results on Movielens-100K**: our model outperforms the baselines in terms of

TABLE 3.3

Accuracy performance of PCMF compared to different baseline on Movielens-100K. We observe that our model has higher RMSE and Precision while having comparable performance in terms of ranking precision, NDCG and RMSE.

Dataset	Movielens-100K - Regular Setting			
Metric	RMSE	Precision @5	Recall@5	NDCG@5
MF	0.94 +/- 0.002	0.22 +/- 0.004	0.6 +/-0.013	0.81 +/1 0.002
IPSMF	0.95 +/- 0.008	0.33 +/- 0.003	**0.63 +/- 0.014**	0.83 +/- 0.004
ItemPop	NA	0.28	0.5	0.8
PCMF	**0.91 +/- 0.001**	**0.348 +/- 0.002**	0.625 +/- 0.012	0.83+/- 0.003

TABLE 3.4

Accuracy performance of PCMF compared to different baselines on Movielens-1M. We notice that our model outperforms all the baselines on all the performance metrics

Dataset	Movielens-1M - Regular Setting			
Metric	RMSE	Precision @5	Recall@5	NDCG@5
MF	1.15 +/- 0.006	0.39+/- 0.001	0.43 +/-0.001	0.76 +/- 0.008
IPSMF	1.1 +/- 5e-06	0.29 +/-	0.38 +/- 0.0003	0.69 +/- 0.005
ItemPop	NA	0.34	0.39	0.75
PCMF	**0.91 +/- 0.004**	**0.42 +/- 0.005**	**0.46 +/- 0.006**	**0.77 +/- 0.001**

RMSE and precision. PCMF improved the ranking precision by 58% compared to Matrix Factorization (MF) and by 6% compared to IPSMF. PCMF gives better NDCG compared to MF, and comparable results to IPSMF. In terms of recall, PCMF provides the second best performance after IPSMF. However, the performance on the recall of PCMF is better than MF. Overall, PCMF provides better results in terms of recommendation ranking quality on Movielens-100K.

- **Results on Movielens-1M**: PCMF shows similar performance compared to Movielens-100K. Our model outperforms the baselines on all recommendation accuracy metrics. Compared to the state-of-the art bias correction model IPSMF, our model improved the precision by 45%, the recall by 20% and NDCG by 10%.

To summarize, PCMF outperformed baselines in terms of recommendation quality on both Movielens-1M and Movielens-100K. Based on these results, we conclude that our model learns more accurate embeddings through modelling the preference of the user to popular items. We also notice that our model outperforms both MF and IPSMF in terms of rating prediction and ranking performance. Which means that adding the popularity correction component helps predict both accurate rankings and accurate ratings.

RQ2: How does our proposed approach compare to other state-of the art methods in terms of promoting diverse non-popular items? To answer this question, we investigate the bias metrics on two real world datasets, Movielens-100K and Movielens-1M as shown in Table 3.5 and Table 3.6 respectively.

- **Results on Movielens-100K**: In terms of popularity bias correction, PCMF reduced the average popularity by 20% compared to MF. This means that our model recommends more accurate long tail items making the recommendation lists not dominated by popular items. PCMF provides more diverse and novel recommendation lists. In fact, PCMF increased the novelty by 9% in terms of ILD compared to MF and MF + MAB; and by 12% compared to IPSMF.

- **Results on Movielens-1M**: Similar performance has been achieved on the ML-1M dataset. PCMF reduces the average popularity by 42% compared to MF and MF + MAB, and by 50% compared to IPSMF. In terms of diversity, PCMF increased the diversity by 20% on average compared to the baselines.

Overall, PCMF succeeded to significantly decrease the popularity bias of the recommender system. It also outperformed other debiasing strategies such Multi-Armed Bandits and Inverse Propensity weighting on several bias metrics. Although our model is not designed to specifically reduce popularity bias, it manages to keep a low level of bias (in terms of popularity and diversity) in both datasets.

TABLE 3.5

Bias results on Movielens 100K using a random split. We notice that our method manages to have comparable performance compared to MAB which is a post-processing method

Dataset	Movielens - 100K - Regular Setting			
Metric	Avg Pop	ILD	MILD	Gini
MF	0.26 +/- 0.01	0.6 +/- 0.01	0.33 +/- 0.008	0.18 +/- 0.005
MF + MAB	**0.17 +/- 0.002**	0.6 +/- 0.015	0.4 +/- 0.02	0.264 +/- 0.007
IPSMF	0.22 +/- 0.011	0.58 +/- 0.008	0.356 +/- 0.012	0.167 +/- 0.004
ItemPop	0.44	0.51	0.25	**0.03**
PCMF	0.21 +/- 0.002	**0.65 +/- 0.007**	0.38 +/- 0.004	0.19 +/- 0.006

TABLE 3.6

Bias results on Movielens 1M dataset. We notice that our method manages to outperform all the baselines in terms of decreasing the average popularity and increasing the diversity of the recommendations.

Dataset	Movielens - 1M - Regular Setting			
Metric	Avg Pop	ILD	MILD	Gini
MF	0.35 ± 0.002	0.51 ± 0.002	0.34 ± 0.004	0.08 ± 0.002
MF + MAB	0.35 ± 0.005	0.52 ± 0.004	0.378 ± 0.003	0.07 ± 0.002
IPSMF	0.44	0.51	0.25	0.03
ItemPop	0.39	0.47	0.26	0.03
PCMF	**0.2 ± 0.004**	**0.61 ± 0.005**	**0.4 ± 0.004**	0.13 ± 0.006

3.3.3 Performance Evaluation of PCMF on the Skew data Setting

In this section, we want to answer **RQ3**. The main goal is to test whether our method can perform well with a heavily biased dataset. Hence, this experiment will mimic a real world scenario. In fact, in the real world, there is always a distribution shift that makes the test data differ in distribution from the training data. In our experiment, the skew test data has more unpopular items than the training data, and hence the popularity distribution of the test is different from that of the training set. In fact, if the model manages to generalize on the test data, then this would demonstrate the robustness of our approach and its generalization ability even for unpopular and new items.

RQ3: How does our proposed approach PCMF generalize on test data with different popularity distribution? Table 3.7 and Table 3.8 list the results that answer this question. In fact we notice that the Skew setting challenge resulted in a drop in accuracy for all the other methods. However, our model maintains its superior accuracy compared to the other baselines. Our method has a 20% better recall and 20% better RMSE with an increase in the NDCG which shows a superior ranking performance.

TABLE 3.7

Accuracy performance of PCMF compared to different baseline on Movielens-100K using a skew split setting challenge. We observe that our model has a higher accuracy performance compared to previous approaches. This indicates that our model is able to learn accurate embeddings to represent the preference of the user and has a better generalisation ability

Dataset	Movielens-100K - Skew Setting Challenge			
Metric	RMSE	Precision @5	Recall@5	NDCG@5
MF	1.19 +/- 0.006	0.27 +/- 0.001	0.5 +/-0.005	0.76 +/- 0.009
IPSMF	0.98 +/- 0.004	0.3 +/- 0.003	0.58 +/- 0.004	0.77 +/- 0.001
ItemPop	NA	**0.35**	0.41	0.71
PCMF	**0.95 +/- 0.001**	0.3 +/- 0.003	**0.59 +/- 0.006**	**0.78 +/- 0.001**

In terms of bias (Table 3.8), we notice that our proposed algorithm, PCMF, outperforms all other debiasing approaches and has the highest diversity score in terms of Intra-List Diversity (ILD) and Min-Intra-List-Diversity (MILD). PCMF also has a similar coverage of popular items compared to IPSMF. These results show that our method learns an accurate representation of the users even if the distribution of the future data is different from the one used in training. This is useful to gauge the performance of PCMF in the challenging realistic scenario of distribution shift and shows higher robustness.

3.3.4 PCMF Learned Embedding

In this section, we investigate **RQ4**. We mainly want to know what is captured by the embedding that is learned by our PCMF model.

TABLE 3.8

Bias results on Movielens-100K using a skew split setting challenge. We notice that our method manages to have comparable performance to IPSMF in terms of average popularity and outperforms other baselines in terms of diversity.

Dataset	Movielens-100K Skew Setting Challenge			
Metric	Avg Pop	ILD	MILD	Gini
MF	0.3 +/- 0.006	0.57 +/- 0.009	0.32 +/- 0.007	0.16 +/- 0.004
MF + MAB	0.3 +/- 0.005	0.57 +/- 0.007	0.32 +/- 0.003	0.16 +/- 0.003
IPSMF	0.2 +/- 0.002	0.58 +/- 0.01	0.36 +/- 0.001	0.17 +/- 0.006
ItemPop	0.46	0.45	0.24	**0.04**
PCMF	**0.2 +/- 0.002**	**0.65 +/- 0.007**	**0.4 +/- 0.004**	0.23 +/- 0.006

RQ4: What do the relevance and popularity embeddings learn in the latent space? In figure 3.4, we project all the users into a compressed representation using TSNE [108] on a two dimensional space. In fact, we see that our model learns two kinds of information, one that is captured by the popularity embedding and another that is captured by the relevance embedding. This is in contrast to Matrix Factorization (MF), where all the embeddings are covering a single dimension. This shows that learning a second embedding from the popularity information in the data can help capture an additional dimension of the user preference.

In Figure 3.5, we plot the representation of items on a two dimensional space using the embeddings that are learned by the popularity component of our model. We also color code the points so that items with higher popularity scores will have a darker color. We notice how the embeddings learn the popularity of items by clustering together items with similar popularity. Basically, this shows that if two items are close to each other in the latent space, then the items will have similar popularity scores. This again shows the ability of incorporating the popularity information of the items in the model without explicitly using it as a feature.

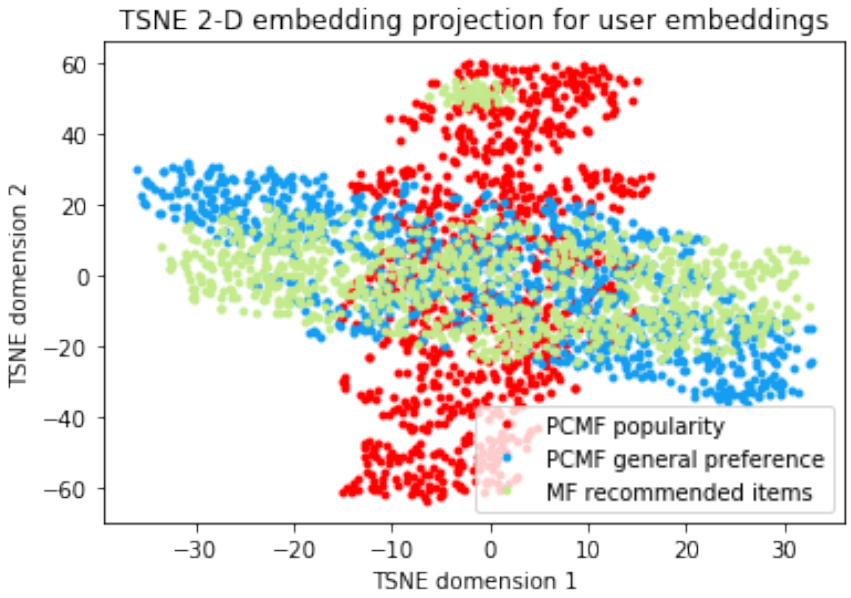


Figure 3.4. Representation of user embeddings on a two dimensional space. We notice that our model learns two different dimensions referring to the preference toward popular items and the general interest of the user

3.3.5 Ablation Study

In this section, we are interested in answering **RQ5** and **RQ6**. We start by investigating the usefulness of including a learnable personalized parameter α_u compared to a constant hyperparameter for all users. Then we study the effect of λ as a balancing parameter for the multitask learning.

RQ5: What is the role of the personalized-parameter α_u in predicting relevant and diverse recommendations? To answer this question, we use a constant α as a hyperparameter instead of a user personalized learnable vector. This means that all users will have the same balance between popularity preference and relevance. Table 3.9 shows the results of both variants of the model. We notice that a personalized α_u yields a better accuracy compared to the non-personalized variation. We also notice that using a constant α returns better bias which is expected since the lower accuracy means that more exploration is introduced into the results. Hence, the personalized α_u is an additional learnable parameter that helps learn a personalized preference for popular items for each user.

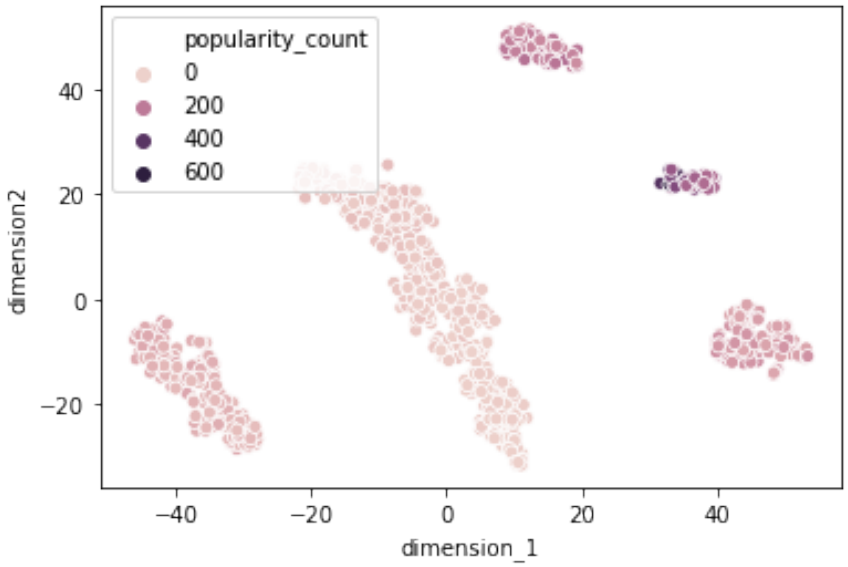


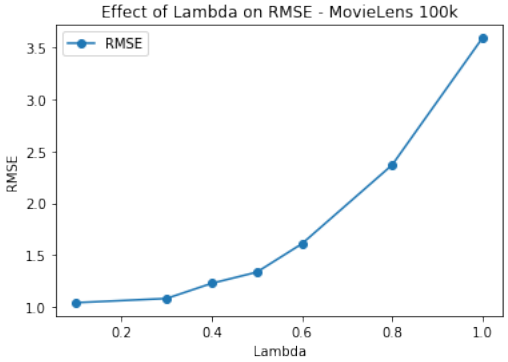
Figure 3.5. Representation of the learned popularity embedding of the items, color-coded to represent the popularity of the items. We notice that popular items are clustered together, meaning that our model is able to capture such a property through the embeddings

TABLE 3.9

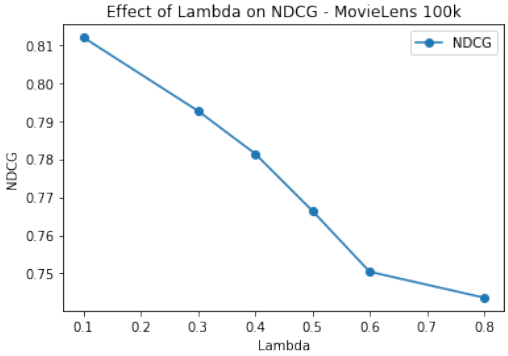
Comparing a different variation of PCMF using a constant lambda as a trade-off parameter. We notice that using a constant lambda yields better debiasing metrics but only at the expense of a significant drop in accuracy. This shows that learning a personalized popularity preference gives more relevant results.

Dataset	Movielens - 100K							
Metric	Bias Metrics				Accuracy Metrics			
	Avg Pop	ILD	MILD	Gini	RMSE	P@5	R@5	NDCG@5
PCMF - Constant alpha	**0.16 +/- 0.022**	**0.74 +/- 0.009**	**0.48 +/- 0.019**	**0.2 +/- 0.034**	1 +/- 0.01	0.3 +/- 0.01	0.57 +/- 0.01	0.8 +/- 0.002
PCMF	**0.2 +/- 0.004**	0.61 +/- 0.005	**0.4 +/- 0.004**	0.13 +/- 0.006	**0.91 +/- 0.001**	**0.348 +/- 0.002**	**0.625 +/- 0.012**	**0.83 +/- 0.003**

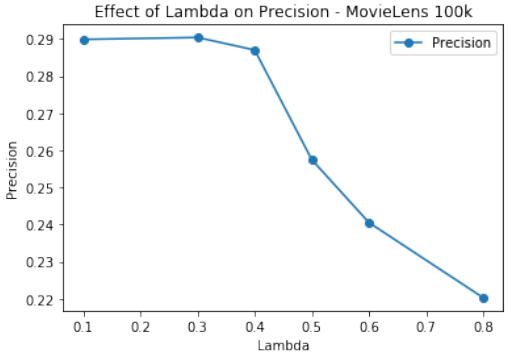
RQ6: What is the role of the added conformity embedding and how does the trade-off parameter λ effect the recommendation performance? In order to study the effect of the added conformity embedding, we monitored the performance of the PCMF algorithm while varying the trade-off parameter λ. We vary λ in $[0, 1]$ and we report RMSE, NDCG, precision, recall, and the average popularity. Figure 3.6 shows the results. Figure 3.6 (a) shows that the RMSE increased when λ increased. Increasing λ means that we attribute the majority of the predicted ratings to conformity at the expense of relevance. This leads to deteriorating the rating prediction, as is shown by the decrease in the RMSE. Figure 3.6 (b) shows the effect of varying λ on NDCG. Similar to the RMSE, NDCG decreases when the recommendation is based on the conformity and not on the relevance. In fact, when λ increases, the contribution of learning the relevance embedding to the loss function decreases and the convergence of the algorithm is biased towards optimizing the conformity embedding loss. We noticed a similar behavior for the Precision and Recall metrics, as shown in Figure 3.6 (c) and Figure 3.6 (d). We noticed, however, that the decrease in precision and recall is not as fast as the one we observed with RMSE and NDCG. This means that for a range of λ, we can decrease the popularity bias without loosing much in terms of accuracy. Figure 3.6 (e) shows the effect of varying λ on the average popularity. As expected, the higher λ is, the higher the penalization when recommending popular items. It means that the weight we attribute to the relevance embedding is low during the training. As a result, the recommendations are getting closer to random recommendations which leads to less popularity bias. To conclude, for a range of λ, PCMF is able to reduce popularity bias without affecting the ranking accuracy. This range varies depending on the application and the data, and λ needs to be tuned to control the trade-off between accuracy and bias correction.



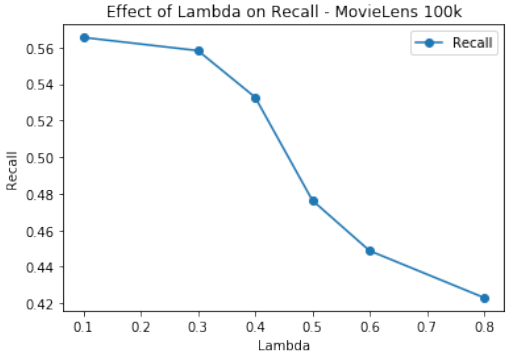
(a) Effect of λ on the RMSE



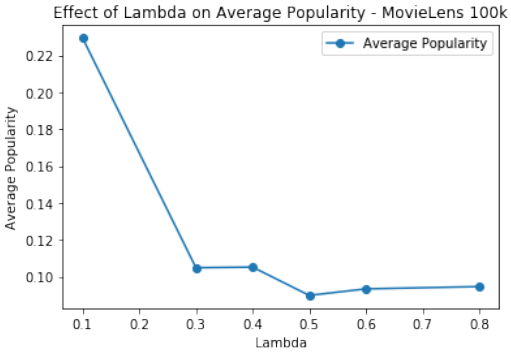
(b) Effect of λ on the NDCG.



(c) Effect of λ on the Precision.



(d) Effect of λ on the Recall.



(e) Effect of λ on the Average Popularity.

Figure 3.6. **Effect of varying parameter λ on the different evaluation metrics. We observe that for $\lambda \in [0.1, 0.3]$, PCMF decreases the average popularity significantly without decreasing the precision and the recall of the recommendation. Recall that the parameter λ controls the trade-off between recommendation accuracy and bias correction**

3.4 Summary

In this chapter, we proposed a novel approach to learn a better representation of the user's interest while accounting for varying conformity and popularity bias. We performed an extensive experimental evaluation on several real world datasets in order to assess the performance of our model in terms of relevance and bias. We concluded that PCMF is able to learn a better representation of users and items in two latent embedding spaces, and provides more accurate results while reducing the popularity bias and increasing the diversity of the recommendations. Finally, we performed an ablation study to assess the effect of the different components of our model.

CHAPTER 4

MULTI-BIAS CORRECTION MATRIX FACTORIZATION

In this chapter, we extend the work presented in Chapter 3 by proposing a new approach for handling two types of bias simultaneously by combining **(1)** the IPS debiasing framework [100] to account for biases in the data collection step and **(2)** our popularity correction matrix factorization (PCMF) presented in Chapter 3, to account for the model popularity bias during the training.

4.1 Motivation and Problem Description

In chapter 3, we presented the PCMF algorithm that tackles popularity bias by disentangling the user's conformity bias from the true relevance. We simplified the problem by assuming that the system suffers only from popularity bias. However, this assumption does not hold in most real world recommendation systems, since the logged data is also affected by other biases such as selection bias, exposure bias, presentation bias, etc. Those biases emerge from the users' interactions with the recommended items. If an item is not clicked or rated, this does not necessarily imply that the user did not like it. This is in part because the user can not rate items that she has not been exposed to. The exposure of the user is typically controlled by the recommender system's previous outputs and the social circle of the user. Presentation bias may change the click behavior of the user as well. For instance, items that are presented higher in the recommendation list have a higher probability of being observed/ examined by the user [59] [109], and the user is typically not exposed to the items ranked low in the recommendation list. Hence, the missing feedback might be a result of an exposure bias and not a lack of relevance to the user. To conclude, the feedback received by the system, i.e. ratings in this context, can be distorted by multiple

biases.

In this chapter, we focus on mitigating two biases simultaneously, i.e. exposure bias and popularity bias. In Chapter 3, we corrected for popularity bias by learning separate embeddings for relevance and conformity using the empirical risk minimization framework and the Average Over All (AOA) estimator [110]. In order to derive a robust learning framework given the presence of exposure bias during the data collection step, we adopt a counterfactual modelling approach using Inverse Propensity Scoring (IPS). It is worth noting that to the best of our knowledge, our work is the first work that studies the effect of popularity bias and exposure bias simultaneously in the collaborative filtering recommendation system setting. We also present extensive empirical results to study the importance of correcting for two biases during the recommendation training phase. The remaining of the chapter is organized as follows: we start by presenting a counterfactual estimator to train PCMF while accounting for exposure bias in addition to the popularity bias, then we present our experimental results using 2 real-world datasets to test the performance of our algorithm and to test the effectiveness of accounting for two biases, and finally we summarize the cumulative takeaways from both Chapter 3 and Chapter 4.

4.2 Multi-bias Correction Matrix Factorization

4.2.1 Notation and Problem Statement

In this chapter, we adapt the same notation as chapter 3. We define a recommender system using explicit feedback data, where U is set of the users and I is the set of items. R represents the rating matrix where each entry r_{ui} represents the rating assigned by user u to item i. O represents the observation matrix $O_{u,i} \in \{0, 1\}$ where each entry denotes if user u has been exposed to item i. $O_{u,i} = 1$ means that the user has been exposed to item i, whereas $O_{u,i} = 0$ means that the user has not been exposed to item i. Each entry $O_{u,i}$ is a Bernoulli random variable indicating if a user u has been exposed to an item i. For ease of reading, we include below, the same notation table from Chapter 3, with the added newly defined variables we will need for this chapter (see table 4.1).

Notation	Definition
U	Set of users
I	Set of items to be recommended
O	Binary Observation matrix indicating if user u observed item i
R	Rating matrix where $r_{u,i}$ is rating of user u to item i
$\widehat{R}$	Predicted rating matrix
Pop	Popularity matrix where $pop_{u,i}$ is the popularity of an item i clicked by user u
$\widehat{Pop}$	Predicted popularity matrix
S	User conformity embedding
V	Item popularity embedding
P	User embedding
Q	Item embedding
$P(O_{ui} = 1)$	Propensity of the user u and item i
α_u, β_u	Personalized bias parameter vectors
γ	L2-Regularization weights
λ	Hyperparameter controlling the personalized linear combination between relevance and conformity
η	Learning rate
$\delta_{u,i}(.,.)$	Recommender system performance evaluation measure
ξ	Parameter control for the severity of position bias
I_u	Recommended list to user u

TABLE 4.1

Notation and Definitions - Chapter 4

In the explicit feedback setting, one goal is to predict an accurate rating matrix $\widehat{R}$. In chapter 3, we used empirical risk minimization framework to train our model. In order to estimate the predicted ratings, we used the Average Overall All (AOA) estimator. We start by presenting the loss function for PCMF in Equation 4.16.

$$
\begin{aligned}
&L^{AOA}_{clicks}(\widehat{R}, \widehat{Pop}) = \frac{1}{|\{(u,i) : O_{u,i} = 1\}|} \sum_{(u,i):O_{u,i}=1} [r_{u,i} - \widehat{r}_{u,i}]^2 + \lambda[pop_{u,i} - \widehat{pop}_{u,i}]^2 \\
&\widehat{r}_{u,i} = \beta_u p_u . q_i^T + \alpha_u s_u . v_i^T \\
&\widehat{pop}_{u,i} = s_u . v_i^T
\end{aligned}
\tag{4.1}
$$

Where R is the true rating matrix with entry $r_{u,i}$, $\widehat{R}$ is the predicted rating matrix with entry $\widehat{r}_{u,i}$ which is the linear combination between the relevance and the conformity scores. Pop is the popularity matrix with $pop_{u,i}$ representing the popularity score of item i clicked by user u, $\widehat{Pop}$ represent the predicted popularity matrix with entry $\widehat{pop}_{u,i}$. The relevance score is the dot product between the user u embedding vector p_u and the item i embedding vector q_i. The conformity score is the dot product between the user u conformity embedding vector s_u and the item i popularity vector v_i. α_u, β_u are **personalized** trade-off parameter controlling the relative contribution of the relevance component $p_u.q_i^T$ and the popularity component $s_u.v_i^T$ for each user. We define $\delta_{u,i}(.,.)$ as the Mean Square Error (MSE) as shown in Equation 4.2.

$$\begin{aligned} \delta_{u,i}(R, \widehat{R}) &= (r_{u,i} - \widehat{r}_{u,i})^2 \\ \delta_{u,i}(Pop, \widehat{Pop}) &= (pop_{u,i} - \widehat{pop}_{u,i})^2 \end{aligned} \tag{4.2}$$

The AOA estimator used to learn PCMF can be written as Equation 4.3:

$$L_{clicks}^{AOA}(\widehat{R}, \widehat{Pop}) = \frac{1}{|\{(u,i) : O_{u,i} = 1\}|} \sum_{(u,i):O_{u,i}=1} \delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop}) \tag{4.3}$$

The AOA estimator uses the average over only the collected (or observed) ratings. In fact, in real world applications, recommender systems operate in a partial-information environment where only a few ratings are available for each user. In an ideal setting, the ratings are estimated assuming that we have full-knowledge information about the system. Equation 4.4 formalizes the PCMF loss function for the rating prediction task in the ideal setting [100].

$$L_{clicks}^{ideal}(\widehat{R}, \widehat{Pop}) = \frac{1}{|U|\,|I|} \sum_{u \in U} \sum_{i \in I} \delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop}) \tag{4.4}$$

This requires collecting the users' ratings for all the items in the system. The scenario is not feasible in real life applications with millions or even billions of items, which leads to using the observed ratings (AOA estimator) to predict the missing ratings. The AOA estimator is statistically unbiased under the condition that the data is Missing Completely At Random

(MCAR). In other words, the collected ratings are a result of the user's interactions with recommendation lists including items chosen randomly.

Proposition 4.2.1. *Assuming the data r_{ui} is missing Completely At Random, the AOA estimator L^{AOA}_{clicks} is an unbiased estimator of the ideal loss L^{ideal}_{clicks}*

Proof. In order to prove that an estimator is unbiased, we need to show that $E(L^{AOA}_{clicks}) = L^{ideal}_{clicks}$ Before starting the proof, we should note that the AOA loss could also be written as

$$L^{AOA}_{clicks}(\widehat{R}, \widehat{Pop}) = \sum_{u \in U} \sum_{i \in I} O_{u,i}(\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})) \tag{4.5}$$

$$E(L^{AOA}_{clicks}) = \sum_{u \in U} \sum_{i \in I} E(O_{u,i})(\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})) \tag{4.6}$$

and since O_{ui} is a Bernoulli variable, $E(O_{ui}) = P(O_{ui} = 1)$. Thus, we have

$$E(L^{AOA}_{clicks}) = \sum_{u \in U} \sum_{i \in I} P(O_{ui} = 1)(\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})) \tag{4.7}$$

If the data r_{ui} is MCAR, then this means that for all u and i, $P(O_{ui} = 1) = \frac{1}{|U||I|}$

Therefore:

$$E(L^{AOA}_{clicks}) = \frac{1}{|U|\,|I|} \sum_{u \in U} \sum_{i \in I} (\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})) \tag{4.8}$$

□

However, the MCAR assumption does not hold in the recommendation system setting. In this context, it means the users' missing ratings are missing at random, which is not true. For instance, a rating can be missing due to multiple reasons: The user may not have been exposed to the item, neither through the recommendation system, nor through their social circle, marketing campaigns, etc. In this case, the item may not have been observed by the user and thus can not be rated. This means that the missing data is Missing Not At Random (MNAR), which leads to poor generalisation ability and misleading judgments of the recommendation results based on the offline evaluation metrics. The misjudgment error is due to multiple biases affecting the data, such as exposure bias where users can not rate items that they have not been exposed to, or popularity bias where popular items get

recommended more often than long tail items or are pushed higher in the recommendation list. Under the presence of the latter biases, the AOA estimator $L^{AOA}_{clicks}(\widehat{R}, \widehat{Pop})$ becomes a biased estimate of the true performance $L^{ideal}_{clicks}(\widehat{R}, \widehat{Pop})$, meaning that [111]:

$$\mathbb{E}_O(L^{AOA}_{clicks}(\widehat{R}, \widehat{Pop})) \neq L^{ideal}_{clicks}(\widehat{R}, \widehat{Pop}) \tag{4.9}$$

Our goal is to tackle the challenge of accounting for several biases simultaneously. As a start, we will explore considering two sources of biases: popularity bias and exposure bias. Chapter 3 described our Popularity bias-aware Correction Matrix Factorization method (PCMF) under the assumption that the data is only affected by popularity bias. In this chapter, we relax the latter assumption by also considering the existence of exposure bias. We will use the inverse propensity estimator to train PCMF in order to mitigate both exposure and popularity bias.

4.2.2 Multi-bias Correction Matrix Factorization Estimator

The AOA estimator, using the observed data, will lead to a biased estimation of the PCMF performance. We assume that each item has a non-zero marginal probability $P(O_{u,i} = 1)$ of being observed. This probability is called propensity [100]. Using a counterfactual modeling framework, the goal is to get an unbiased estimate of $\delta_{u,i}(R, \widehat{R})$ and $\delta_{u,i}(Pop, \widehat{Pop})$ via an Inverse Propensity scoring (IPS estimator). In order to mitigate the effect of exposure bias in the training phase, we use the Inverse Propensity Weighting (IPW) framework [100] by weighting each rated item by its propensity of being observed.

$$L^{IPS}_{clicks}(\widehat{R}, \widehat{Pop}) = \frac{1}{|U||I|} \sum_{(u,i):O_{u,i}=1} \frac{1}{P(O_{u,i} = 1)} .(\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})) \tag{4.10}$$

Theorem 4.2.2. *The IPS-PCMF loss defined by:*

$$L^{IPS}_{clicks}(\widehat{R}, \widehat{Pop}) = \frac{1}{|U||I|} \sum_{(u,i):O_{u,i}=1} \frac{(\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop}))}{P(O_{u,i} = 1)} \tag{4.11}$$

is an unbiased estimator of the ideal loss L^{ideal}_{clicks}

Proof.

$$\mathbb{E}_{O_i}[L^{IPS}_{clicks}(\widehat{R}, \widehat{Pop})] = \frac{1}{|U|\,|I|} \mathbb{E}_{O_i}[\sum_{(u,i):O_{u,i}=1} \frac{\delta_{u,i}(R, \widehat{R}) + \lambda \delta_{u,i}(Pop, \widehat{Pop})}{P(O_{u,i} = 1)}]$$

By linearity of expectation:

$$= \frac{1}{|U|\,|I|} \sum_{(u,i):O_{u,i}=1} \mathbb{E}_{O_i}[\frac{\delta_{u,i}(R,\widehat{R}) + \lambda\delta_{u,i}(Pop,\widehat{Pop})}{P(O_{u,i}=1)}]$$

$$= \frac{1}{|U|\,|I|} \sum_u \sum_i \mathbb{E}_{O_i}[\frac{\delta_{u,i}(R,\widehat{R}) + \lambda\delta_{u,i}(Pop,\widehat{Pop})}{P(O_{u,i}=1)}.O_{u,i}]$$

$$= \frac{1}{|U|\,|I|} \sum_u \sum_i \mathbb{E}_{O_i}[O_{u,i}].\frac{\delta_{u,i}(R,\widehat{R}) + \lambda\delta_{u,i}(Pop,\widehat{Pop})}{P(O_{u,i}=1)}$$

$$= \frac{1}{|U|\,|I|} \sum_u \sum_i P(O_{u,i}=1).\frac{\delta_{u,i}(R,\widehat{R}) + \lambda\delta_{u,i}(Pop,\widehat{Pop})}{P(O_{u,i}=1)}$$

$$= \frac{1}{|U|\,|I|} \sum_u \sum_i \delta_{u,i}(R,\widehat{R}) + \lambda\delta_{u,i}(Pop,\widehat{Pop})$$

$$\mathbb{E}_O[L^{IPS}_{clicks}] = L_{ideal}(\widehat{R},\widehat{Pop})$$

Line 4 results from the fact that $O_{u,i}$ is a Bernoulli random variable, thus $\mathbb{E}[O_{u,i}] = P(O_{u,i}=1)$. Line 5 is due to the fact that $P(O_{u,i}=1) > 0$, as all items have a non zero probability of being observed. □

4.2.3 Propensity Score Estimation

In order to correctly estimate the effect of the exposure bias, the propensity scores need to be calculated. As it is impossible to know exactly the exposure model of the user, we use models that approximate this exposure based on different assumptions. One popular [112] way to model the propensity score is using the popularity model where:

$$P(O_{u,i}=1) = \frac{\sum_{u\in U} \mathbb{1}_{O_{u,i}>0}}{|U|} \tag{4.12}$$

This model assumes that the exposure is directly correlated to the popularity of the item. Although this is a valid assumption (the more popular the item, the higher its exposure across all users), it neglects other sources of exposure such as location, the user's social circle, marketing campaigns, etc.

Another way to calculate the probability of exposure is using the position or rank of the items when shown in a recommendation list. This requires the position information

to be available in the data. One proposed model to estimate the exposure from the rank of the observed item is [113]:

$$P(O_{u,i} = 1|rank(i, I_u)) = \left(\frac{1}{rank(i, I_u)}\right)^{\xi} \tag{4.13}$$

Where I_u is the recommended item list for user u and ξ is a parameter to control for the severity of the position bias: The higher ξ, the higher the position bias. Assuming that the probability of an item being observed depends only on the position bias, the rank based propensity will account for the position bias in the data. The main drawback of the above modeling approach is that it requires rank information which is usually hard to collect and is not available in most public datasets.

Another way to estimate the propensity score is by using Poisson factorization [114]. In fact the exposure of each user differs for each individual and usually depends on many factors such as the social circle, user history and previously interacted items. More involved models can be used in order to estimate the propensity score. One model that was shown to have great potential [112] at modeling the exposure is Poisson Factorization [115–117]:

$$P(O_{u,i} = 1) \sim \text{Poisson}(\theta_u^T \phi_i) \tag{4.14}$$

Where θ_u and ϕ_i are representations of users and items in a latent space verifying the below equation:

$$\theta_u, \phi_i \sim \Gamma(k, \theta) \tag{4.15}$$

where k and θ are parameters of the Poisson distribution. In our experiments we use Poisson Factorization as our model for the propensity scoring function for both IPS and IPS-PCMF.

4.2.4 Update equations and algorithm

We now show the derivations that are needed to obtain the update equations for an algorithm that will learn the optimal parameters for dual (popularity and exposure) bias-aware recommendation. We use the following combined MSE loss in order to train our model.

$$L^{AOA}_{clicks}(\widehat{R}, \widehat{Pop}) = \frac{1}{|U||I|} \sum_{(u,i):O_{u,i}=1} \frac{1}{P(O_{ui}=1)} .[(r_{u,i} - (\beta_u p_u . q_i^T + \alpha_u s_u . v_i^T))^2 + \lambda(pop_{u,i} - s_u . v_i^T)^2 + \gamma(\|p_u\|^2 + \|q_i\|^2 + \|s_u\|^2 + \|v_i\|^2)] \tag{4.16}$$

The update equations are similar to the PCMF model except for adding the IPS weight $\frac{1}{P(O_{ui}=1)}$. Denoting ν as the learning parameter, the latent factor vectors for the user and item relevance and popularity embeddings, as well as the user personalization coefficients for relevance and popularity conformity bias β_u and α_u, respectively, are given by

$$p_u^{(t+1)} \leftarrow p_u^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2\beta_u^{(t)}(q_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| p_u^{(t)} \right\|] \tag{4.17}$$

$$q_i^{(t+1)} \leftarrow q_i^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2\beta_u^{(t)} p_u^{(t)}(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| q_i^{(t)} \right\|] \tag{4.18}$$

$$s_u^{(t+1)} \leftarrow s_u^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2\lambda(v_i^{(t)})^T(pop_{u,i} - \widehat{pop}_{u,i}^{(t)}) - 2\alpha_u^{(t)}(v_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| s_u^{(t)} \right\|] \tag{4.19}$$

$$v_i^{(t+1)} \leftarrow v_i^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2\lambda s_i^{(t)}(pop_{u,i} - \widehat{pop}_{u,i}^{(t)}) - 2\alpha_u^{(t)} s_i^{(t)}(r_{u,i} - \widehat{r}_{u,i}^{(t)}) + 2\gamma \left\| v_i^{(t)} \right\|] \tag{4.20}$$

$$\alpha_u^{(t+1)} \leftarrow \alpha_u^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2s_i^{(t)}(v_i^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)})] \tag{4.21}$$

$$\beta_u^{(t+1)} \leftarrow \beta_u^{(t)} - \frac{\eta}{P(O_{ui}=1)}[-2p_i^{(t)}(q_{u,i}^{(t)})^T(r_{u,i} - \widehat{r}_{u,i}^{(t)})] \tag{4.22}$$

IPS-PCMF is a two-stage approach. First, we train the exposure model on the observation matrix O to compute the propensity scores $P(O_{u,i} = 1)$. Then, we use the learnt propensity scores to train our PCMF model. Algorithm 4.1 describes the steps of the IPS-PCMF approach.

4.3 Experimental Results

In this section we present our evaluation experiments which aim to answer the following research questions:

- **RQ 5.1:** How effective is IPS-PCMF at removing the exposure bias in the data on an initially unbiased test dataset, compared to other baselines?

Algorithm 4.1 Dual Popularity and Exposure Bias Correction MF

Input: Rating matrix $R : \{r_{u,i} : (u,i) \in O\}$, Popularity matrix $Pop : \{p_{u,i} : (u,i) \in O\}$, Regularization parameter γ, Trade-off parameter λ, Observation matrix O
Output: User Conformity latent factors $S_{1:U}$, $V_{1:I}$, User relevance latent factors $P_{1:U}$, $Q_{1:I}$, Personalized bias vector α_u, β_u

```
Learn propensity scores P_{u,i} from observation matrix O using Equation 4.14
Randomly initialize S_{1:U}, V_{1:I}, P_{1:U}, Q_{1:I}, α_u, β_u
while not converged do
  for u ← 1 : U do
    Update S_{1:U}, P_{1:U} using Eq 4.19 & Eq 4.17
    Update α_{1:U}, β_{1:U} using Eq 4.21 & Eq 4.22
  end for
  for i ← 1 : I do
    Update V_{1:I}, Q_{1:I} using Eq 4.20 & Eq 4.18
  end for
end while
return S_{1:U}, P_{1:U}, V_{1:I}, Q_{1:I}
```

- **RQ 5.2:** How do the relevance of the IPS-PCMF recommendations compare to other baselines on a biased dataset?
- **RQ 5.3:** How does the debiasing performance of IPS-PCMF compare to other baselines?
- **RQ 5.4:** How effective is the IPS scoring at improving the results of IPS-PCMF?

4.3.1 Experimental Setting

4.3.1.1 Dataset:

In order to evaluate the performance of our models we used two real world datasets, namely the Movielens-100K and Yahoo!R3 Music dataset.

- **Movielens-100K** [1]: This dataset contains around 1700 items and 1000 users with 100K collected ratings. It has a density of 4%.
- **Yahoo!R3** [2]: This dataset has 300K ratings as training data and 54K ratings as test data. The main characteristic of this dataset is that it contains an "unbiased" test-set.

[1] `https://grouplens.org/datasets/movielens/100k/`
[2] `http://webscope.sandbox.yahoo.com/`

The 54K-ratings test set is collected by showing to the users random recommendations and asking the user to provide feedback. Therefore, the data is free from exposure bias. We use the unbiased dataset to be able to evaluate an unbiased performance of our model

4.3.1.2 Baselines:

We compare our method mainly to Inverse Propensity Scoring (IPS) [18] combined with MF. As MF is still a leading model providing state of the art results on Collaborative Filtering tasks [57]. Combined with IPS, IPS-MF is a state of the art method to learn unbiased recommendations [100]. We also compare our method to a Multi-Armed Bandits strategy in order to assess the exploration ability of our model [104].

4.3.1.3 Experimental set-up

In this section, we discuss the implementation details of the proposed approach. We use Poisson Factorization in order to estimate the propensity score as defined by equation 4.14. The PCMF component used is similar to the model introduced in the previous chapter. We tune our models using Optuna [105] on a separate validation set. We adopt a 80-10-10 split on the Movielens-100K data and 80-20 split on the Yahoo!R3 dataset, while using the unbiased Yahoo!R3 dataset for testing. All the models were implemented using the Pytorch framework [118] and executed on a Tesla V100 cluster. We report accuracy and bias metrics.

4.3.1.4 Metrics

The aim of IPS-PCMF is to provide relevant, unbiased and diverse recommendations by using an intentional mitigation of both exposure and popularity bias. For this reason, we use accuracy metrics such as RMSE for rating prediction, NDCG for Ranking, and Precision and Recall to measure the accuracy of retrieval. We also use the Yahoo!R3 unbiased test data in order to assess the unbiaseness of the result. An unbiased prediction would yield a

better RMSE on a randomly collected test data.

In order to assess the diversity of the predictions, we use the same metrics defined in Chapter 3. Hence, we mainly focus on how our proposed methods affect (1) the popularity level of the results through the average popularity metric and (2) the diversity of the recommendations through the ILD and MILD metrics.

4.3.2 Experimental results

RQ 5.1: How effective is IPS-PCMF at removing the exposure bias in the data on an initially unbiased test dataset, compared to other baselines? Table 4.2 shows the accuracy metrics of our model compared to different baselines. We notice that our model outperforms previous baselines in terms of RMSE and Precision. This result is expected since we use an unbiased estimator of the MSE in order to optimize the performance of the model. One future possibility that can also improve the ranking performance of the model is to use a ranking based loss such as BPR [119].

TABLE 4.2

Accuracy Performance of PCMF-IPS on the Yahoo!R3 unbiased test Dataset. Our model succeeds to significantly improve the RMSE compared to IPS+MF.

Dataset	Yahoo! R3 Dataset			
Metric	RMSE	Precision @5	Recall@5	NDCG@5
MF	1.41 +/- 0.00015	0.02 +/- 0.0006	0.56 +/- 0.011	0.46 +/- 0.019
MF+MAB	1.41 +/- 0.002	0.02 +/- 0.002	0.54 +/- 0.05	0.47 +/- 0.048
IPS+MF	1.37 +/- 0.007	0.034 +/- 7e-05	**0.71 +/- 0.002**	0.5 +/- 0.007
PCMF	1.209 +/- 0.005	0.025 +/- 0.0001	0.49 +/- 0.002	0.49 +/- 0.002
PCMF+IPS	**1.2 +/- 0.005**	**0.035 +/- 0.0001**	0.5 +/- 0.002	0.5 +/- 0.01

We observe that our model manages to maintain a lower error compared to IPS+MF. This means that the popularity component in our model does indeed help improve the learning performance, while also keeping the results unbiased. Therefore we conclude that our model is effective at learning accurate and unbiased recommendations, and that it accomplishes this significantly better than the compared baselines, which answers RQ 5.1.

RQ 5.2: How do the relevance of the IPS-PCMF recommendations compare to other baselines on a biased dataset? Table 4.3 shows the accuracy performance of our model on the Movielens-100K dataset. In this experiment, we manage to outperform the competitive baselines on most metrics, which demonstrates that our model is able to provide more accurate predictions than the compared baselines, when tested on biased datasets.

TABLE 4.3

Accuracy performance of PCMF + IPS compared to different baselines on Movielens-100K. We observe that our model has lower RMSE than the other baselines, including the state of the art IPS+MF, while reaching similar levels of precision, recall and NDCG as IPS+MF

Dataset	Movielens - 100K -			
Metric	RMSE	Precision @5	Recall@5	NDCG@5
MF	0.94 +/- 0.002	0.22 +/- 0.004	0.6 +/-0.013	0.81 +/1 0.002
MF+MAB	0.932 +/- 0.012	0.21 +/- 0.003	0.56 +/- 0.013	0.79 +/- 0.002
IPS+MF	0.95 +/- 0.008	0.33 +/- 0.003	**0.63 +/- 0.014**	0.83 +/- 0.004
PCMF	0.91 +/- 0.001	0.34 +/- 0.002	0.625 +/- 0.012	0.83+/- 0.003
PCMF + IPS	**0.90 +/- 0.007**	0.34 +/- 0.009	0.62 +/- 0.01	0.83 +/- 0.003

We particularly observe that our model significantly improves the RMSE performance (while other metrics remained at similar levels to the state of the art baseline IPS+MF). This is is due, as we have previously explained, to using an accurate and unbiased estimation of the Mean Squared Error function, when training the model. Also, we can conclude that correcting for two biases simultaneously (PCMF+IPS) further improves the RMSE compared to correcting for only popularity bias (PCMF).

RQ 5.3: How does the debiasing performance of IPS-PCMF compare to other baselines? Table 4.4 and Table 4.5 show the results in terms of bias metrics of our model compared to other baselines. We notice that PCMF+IPS significantly improves the diversity of the recommendations on the Yahoo!R3 dataset. In fact, it improves the Average popularity by 30 % compared to IPS+MF and improves the ILD metric by 17 % and the MILD metric by 10 %, compared to IPS+MF and improves MILD by 16 % compared to MF and Multi-Armed Bandits.

We also notice that on the biased dataset, our model manages to maintain a good level of diversity. In fact, it improves the ILD metric by 8 % compared to IPS+MF, MF and MAB, while keeping a comparable Average Popularity metric compared to MF.

We conclude that our model manages to generate diverse recommendations, while also achieving a high coverage by recommending items from the long tail.

TABLE 4.4

Bias Performance of our IPS-PCMF on Movielens-100K. Our model manages to maintain the same popularity results provided by PCMF while increasing the diversity of the recommendation list

Dataset	Movielens - 100K			
Metric	Avg Pop	ILD	MILD	Gini
MF	0.26 +/- 0.01	0.6 +/- 0.01	0.33 +/- 0.008	**0.18 +/- 0.005**
MF + MAB	**0.17 +/- 0.002**	0.6 +/- 0.015	0.4 +/- 0.02	0.264 +/- 0.007
IPSMF	0.2 +/- 0.011	0.6 +/- 0.02	0.39 +/- 0.016	0.2 +/- 0.022
PCMF	0.21 +/- 0.002	0.65 +/- 0.007	0.38 +/- 0.004	0.19 +/- 0.006
PCMF + IPS	0.2 +/- 0.003	0.65 +/- 0.01	0.4 +/- 0.002	0.19 +/- 0.005

TABLE 4.5

Bias Performance on the Yahoo!R3 dataset. Our model outperforms all the baselines in terms of decreasing the popularity and providing diverse recommendations to the user.

Dataset	Yahoo!R3 Dataset			
Metric	Avg Pop	ILD	MILD	Gini
MF	0.2 +/- 0.0013	0.76 +/- 0.001	0.57 +/- 0.006	0.21 +/- 0.003
MF + MAB	0.19 +/- 0.013	0.77 +/- 0.02	0.58 +/- 0.013	0.23 +/- 0.033
IPSMF	0.23 +/- 0.001	0.72 +/- 0.0008	0.62 +/- 0.001	0.133 +/- 0.002
PCMF	0.156 +/- 0.004	0.797 +/- 0.009	0.64 +/- 0.001	**0.21 +/- 0.005**
PCMF + IPS	**0.14 +/- 0.003**	**0.82 +/- 0.006**	**0. 68 +/- 0.01**	0.23 +/- 0.004

RQ 5.4: How effective is the IPS scoring at improving the results of IPS-PCMF?

In our previous results, we compared IPS-PCMF to the vanilla (single bias) version PCMF, presented in Chapter 3. We now evaluate their performance in terms of relevance and

diversity on both the Yahoo!R3 and Movielens-100K datasets.

- Performance on the Movielens 100K dataset: IPS-PCMF improves upon the performance of PCMF in terms of RMSE, meaning that we manage to get better rating prediction accuracy. We also improve the diversity of the results and the popularity level by recommending more unpopular and diverse items. This shows that adding the IPS debiasing weights helps improving the quality of the recommendations.

- Performance on the Yahoo!R3 dataset: The effect of the debiasing strategy on PCMF is even more noticeable on this dataset since the IPS component helps in reducing the bias in the estimation of the performance via the debiased loss function and therefore yields a better performance on the test set. IPS-PCMF improves the accuracy of the results across all metrics (RMSE, Precision, Recall and NDCG). It also succeeds in providing significantly less biased recommendations by improving the popularity bias by 10% and providing more diverse recommendation by scoring 0.82 on the ILD metric and 0.68 on the MILD metric vs. 0.79 and 0.64 for the PCMF respectively.

Based on our results, we conclude that adding IPS to PCMF helps reduce the exposure bias further as demonstrated by the improved results in terms of both accuracy and diversity.

4.3.3 Future directions

Our multi objective framework combined with IPS training provides an appealing way to combine multiple biases, an area that until now, has been understudied in the literature. In fact, tackling multiple biases at the same time is important to reduce their combined effect. As biases do not act individually and are often inter-related, it is important to study their joint effect on different modeling approaches. In the future, further debiasing techniques could be added to study their effect on the recommendation system. One way to achieve this is to study post-processing techniques such as by using Multi-Armed Bandits in combination with our recommendation algorithm. Another area of expansion would be to study the iterative behavior of our algorithm. In fact, recommender systems work in a

dynamic environment and this dynamic aspect should be incorporated into the modeling of different debiasing approaches.

4.4 Summary

In this chapter, we proposed a multi-bias framework that helps reduce the popularity and exposure bias simultaneously through incorporating the IPS training framework within the PCMF modeling. We provided a problem statement and motivation behind our approach by studying the limitations of previous models, then we described our modeling approach and provided a theoretical analysis of our method. Finally we presented an extensive empirical evaluation on different datasets to test the performance of our algorithm in terms of accuracy and bias, in comparison with competitive baselines.

CHAPTER 5

BRIDGE FUNCTION FOR BIAS CORRECTION

In this chapter, we present a new approach to adjust for confounders in recommendation systems. In order to do this, we reframe the recommendation system within a causal framework and then propose to use a proximal causal inference framework [22] to adjust for confounders. Confounders are variables that may affect the predicted ratings leading to biased estimates that can be misleading. This may mislead the prediction of the user preference, and in turn yield a weaker recommendation performance.

5.1 Motivation and Problem Description

The goal of recommendation systems is to provide the users with useful recommendations. Given the users past interactions with the system (i.e. ratings or clicks), the recommendation algorithm learns the users' preferences and interests and provide the users with a potential list of items they will like. The recommendation algorithm learns to predict the users' ratings for items that they did not previously rate. Then based on these predictions, the user is provided with a list of items they are predicted to rate highly, according to the recommendation algorithm predictions. In this section, we will reframe the recommendation system problem as a causal inference problem, where the rating predictions will be considered to be causal predictions. We focus on Matrix Factorization, however our approach is applicable to any base recommendation model.

In a traditional setting, Matrix Factorization (MF) [57] is trained on the observed ratings. The recommendations provided by MF would be unbiased only if the training data was independent and identically distributed (iid) [100]. This would imply that the ratings would have been collected randomly, which is, however, not the case in a recommendation

system setting. In fact, users generally do not rate movies that they did not watch. This means that the ratings are not collected randomly, they are selected by the recommendation algorithm and self-selected by the user. In other words, MF's estimates using observed data are biased. In addition to the bias from missing ratings, even the non-missing ratings themselves may incorporate a variety of inherent user biases, such as conformity bias, presentation bias, or preference for popular items, etc. All these diverse types of bias may interact together and affect both the input ratings and the output recommendations (and hence the user's exposure to certain items).

Framing the recommendation problem as a causal inference problem will give us the opportunity to study the recommender system's behavior in a cause-effect setting. Modeling the recommendations with respect to the causes could in turn provide a more robust system. The recommendation (analogous to treatments in a medical study) can be seen as treatments and a rating can be considered to be an outcome variable. Hence predicting the users' ratings for unseen movies is equivalent to answering the following "counterfactual" causal question:

"*What would the rating be if the user was in a setting where they were asked to watch and then rate the movie*?"

Because the "observed" ratings are not collected randomly, it is challenging to answer the above causal question since using only the observed data is biased. In order to have an unbiased estimate of the treatment effect (which would also require the ratings for *unseen* movies), one must account for all possible confounders [120]. This assumption is known as the *ignorability* or *exchangeability* assumption [120], which is impossible to verify in real life. *Confounders* are variables that may affect both the movies recommended to the user (the treatment) and the user's ratings (the outcome). Specifying the possible confounders is subjective and is often left to domain-expert judgments [22]. For example, a user may choose to watch a movie because she is a fan of the leading actor; in this case the preference for the leading actor is a confounder. The challenge is the fact that not all counfounders can be measured or observed. For instance, the user's tendency to be influenced by public ratings, may affect their rating of a given movie. Social influence may also affect the user's

rating. The intensity of the conformity bias is yet another example of an immeasurable confounder. This is why in practice, one may hope to find *proxies* that are a good reflection of the unmeasured confounder.

In order to overcome the above challenges, we adopt a *proximal causal inference framework* [22] to deal with the presence of unobserved confounders. This framework acknowledges the fact that the observed covariates are imperfect proxies for the underlying confounder mechanisms, and presents an approach to measure the potential outcome in settings where the exchangeability assumption does not hold. Our *proximal causal recommender system* builds on classical matrix factorization (MF) and aims to correct for unobserved confounders. The approach leverages two negative control variables [121] to adjust for unobserved confounders. One *negative control outcome variable*, which is a variable that is causally independent of the *exposure* variable responsible for *exposing* the user to an item via a prior recommendation and affects only the ratings (outcome variable). The second is a *negative control exposure variable*, a variable that affects only the *exposure*, but is causally independent of the outcome (ratings). If there exists a confounding bridge function [121] that captures the link between the ratings (potential outcome) and the negative outcome control distribution, then the bridge function can be identified using the negative control exposure variable (i.e., when the user does not get exposed to the item through a recommendation). The bridge function is a given transformation of the observed variable and can be considered as a mediator that capture the unmeasured confounders' effect on the observed data. The expected value of the learnt bridge function can therefore provide ratings that are adjusted for *unobserved* confounders. This step can be seen as a preprocessing step, where the bridge function is used to *reconstruct* the observed rating while accounting for unobserved confounders. These adjusted ratings can then be used to predict potential ratings.

The proximal causal recommender aims to correct for the potential unobserved confounders. Our approach is a two-stage approach. First, we learn the bridge function to adjust for unobserved confounders using a pair of negative control variables that we will

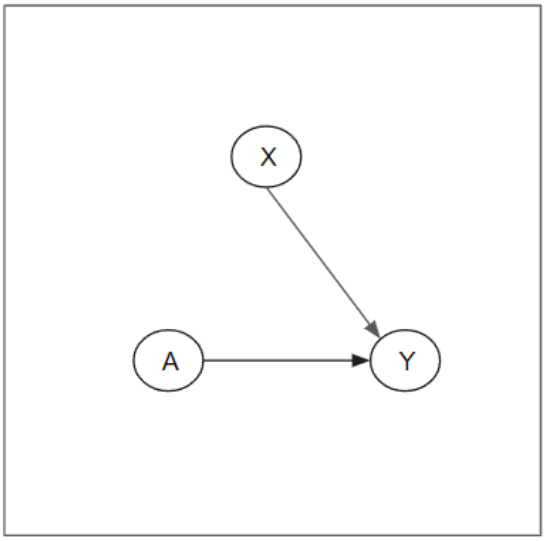
define in the upcoming sections. Second, we fit a MF model on the adjusted ratings that are predicted by the bridge function learnt in the first stage. This procedure is expected to correct for the bias in the predicted ratings with respect to unobserved confounders.

5.2 Bridge Function for Bias Correction

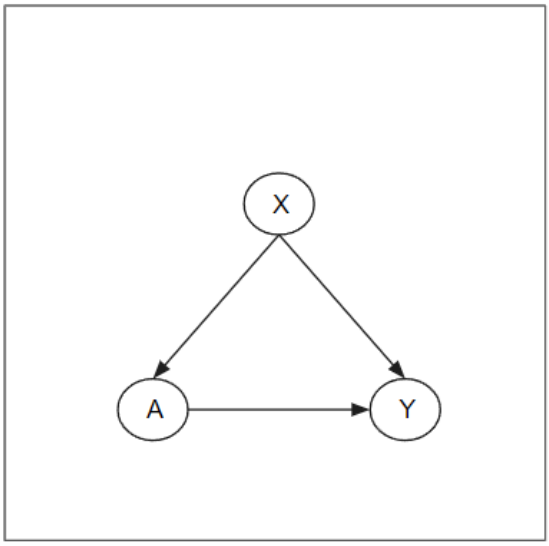
In this section, we present the components of our proposed approach. We start by the notation used and the problem formulation. Then, we provide causal graph interpretations of existing approaches as well as our proposed approach for comparison purposes and to distinguish our contribution from existing work. Next, we state the assumptions that need to be met to guarantee the identification of the causal effects. Finally, we present our design choices and the proposed loss functions for the learning task.

5.2.1 Notation and Definition

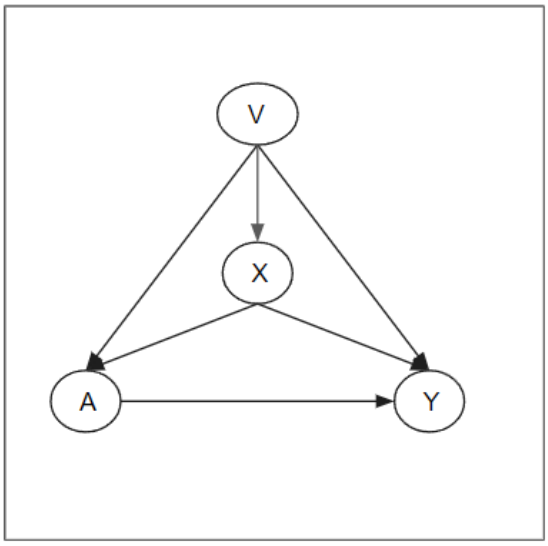
Let A be a binary *exposure* variable, indicating whether user u has received the "treatment" (which is to be exposed to the item via a previous recommendation and prior to rating it). The exposure variable A controls which items will be recommended to the users. Let U and I denote the set of users and the set of items, respectively. The exposure variable A is a binary matrix where $a_{u,i} = 1$ if the user u has rated item i and $a_{u,i} = 0$ otherwise. Let Y be the outcome variable representing the ratings. Following the convention in causal inference, let $y_{u,i}(1)$ denote the potential rating assigned by user u to item i, if she were to watch the movie. $y_{u,i}(1)$ represents the potential outcome under an intervention or treatment, meaning the rating that user u will give to item i, if she was asked to watch it and then rate it. $y_{u,i}(1)$ is observed if user u rated item i, otherwise it is unobserved. For the recommendation system problem, the unobserved ratings are $y_{u,i}(0) = 0$, since users can not rate movies that they have not seen. Table 5.1 summarises the notation used in this chapter.



(a) Causal Graph for Classic MF



(b) Causal Graph for IPS MF in the presence of observed confounders X



(c) Causal Graph in the presence of Observed confounders (X) and Unobserved Confounders (V)

Figure 5.1. **Causal graphs of different Recommendation systems, where X denotes user and item features, A stands for the exposure variable, Y denotes the potential rating, and V denotes an unobserved confounder**

Notation	Definition
U	Set of users
I	Set of items to be recommended
A	Binary Exposure variable indicating if user u observed item i
Y	Rating matrix where $y_{u,i}(a=1)$ is the rating of user u to item i
$\widehat{Y}$	Predicted rating matrix
X	User and item covariates
W	Negative Control Outcome variable (NCO)
Z	Negative Control Exposure variable (NCE)
h	Outcome Confounding Bridge
$\gamma_{u,i}(.,.)$	Recommender system performance evaluation measure
$\perp\!\!\!\perp$	independent of
$y(a_{ui}=1)$	treatment effect (Rating) under a positive exposure $a_{ui}=1$
$y(a_{ui}=0)$	treatment effect (Rating) under a negative exposure $a_{ui}=0$

TABLE 5.1

Notation and Definitions used in Chapter 5

5.2.2 Causal Graph Interpretation of Recommendation System

In this section, we introduce a causal view of classic recommendation approaches and compare it to our proposed approach. A causal graph is a directed acyclic graph G where the nodes represent the set variables and the edges represent the Cause-effect relationship between the variables. Figure 1(a) represents the causal graph, without loss of generality, of the majority of collaborative filtering recommendation approaches, including matrix factorization (MF). The variable A represents the exposure variable indicating whether the user has been exposed to a movie or not (analogous to whether a patient has received a treatment or not in a medical study). The algorithms rely on user and item features, denoted as X, to predict the user's ratings for unseen items assuming the existence of no confounder between the exposure variable and the ratings. Matrix Factorization (MF), for instance, follows the causal graph in figure 1(a). MF learns the similarity between users and items using the dot product between the user and item features X (user and item embeddings). Given the theory around potential outcomes, the rating estimates are unbiased assuming ignorability

/ exchangeability stating that $(y(a_{u,i} = 1), y(a_{u,i} = 0)) \perp\!\!\!\perp A$ [120]. This means that the user ratings would be the same independently of being exposed to all the items or none of them. This, of course, does not hold in the real world recommendation setting since the user's ratings are not independent of the user being exposed to the items. For instance, the users generally do not rate movies they have not been exposed to and the recommendation process is rather personalized and not random.

Given the fact that user and item features X affect the exposure variable A, the process by which the user receives recommendations, we present a more realistic causal graph in Figure 1(b). The causal graph shown in figure 1(b) depicts how recommendation systems operate in an observational setting, where observed confounders X, representing user and item features affect both the ratings and the exposure. Theory around potential outcomes states that, an unbiased estimate of the ratings is possible when controlling for all variables (covariates) that can affect both the treatment and the outcome [122] [123]. This is known as the *conditional ignorability assumption* [122] [123], stating that $(y(a_{u,i} = 1), y(a_{u,i} = 0)) \perp\!\!\!\perp A|X$. Note that the conditional ignorability assumption is untestable. Suppose, we collect all user and item counfounders satisfying the conditional ignorability assumption, classical causal inference controls for the confounders in the outcome model. In real world applications, given the curse of dimensionaly, it may be challenging to control for all confounders. Research [124] [125] has shown that it is sufficient to control for the propensity of an item being observed given the observed confounders X, such that $P(a_{u,i} = 1|X)$.

Assumption 5.2.1. $(y(a_{u,i} = 1), y(a_{u,i} = 0)) \perp\!\!\!\perp A|P(A = 1|X)$.

This has led to the Inverse Propensity Weighed Matrix Factorization [100]. Inverse propensity training has gained popularity in recent years in the recommendation system community, where it has been used to correct for the bias in the training phase. The estimator follows Equation 5.1.

$$L^{IPS}(\widehat{Y}|P) = \frac{1}{|\{(u,i) : a_{u,i} = 1\}|} \sum_{(u,i):a_{u,i}=1} \frac{\gamma_{u,i}(Y, \widehat{Y})}{P(a_{u,i} = 1|X)} \tag{5.1}$$

Where $\gamma_{u,i}(.,.)$ denotes a recommender system performance evaluation measure. The estimator in Equation 5.1 can be shown to be statistically unbiased. The inverse propensity weighed Matrix Factorization performance depends on the propensity scores' estimation, since the propensity scores are unknown and need to be estimated from the observed data.

Estimating propensity solely relies on the collection of rich observed covarites that capture all sources of confounding, so that the conditional ignorability holds. The challenge is the presence of potential *unmeasured* confounding variables (V), as shown is Figure 1(c). In this case, the ignorability assumption fails, and the inverse propensity weighting correction method may become severely biased and potentially misleading [22]. In recommendation systems, such unobserved unmeasured confounders exist. For instance, the user's conformity level while rating the movies may impact the outcome and may impact the exposure as well. The user's conformity in this case is an *unobserved* confounder that is difficult to quantify. In this work, we will adopt the potential outcome framework for proximal causal learning [22] which offers the opportunity to correct for the bias, when conditional ignorability on the basis of measured confounders fails. The approach relies on a pair of negative control variables to learn a bridge function that captures the effect of the unobserved confounder of the ratings. Then, after learning the bridge function, the adjusted ratings are used to predict the potential ratings using MF. The predictd ratings are unbiased under a set of assumptions. In the next section, we will present a brief summary of the theory behind it and we will state the sufficient assumptions for non-parametric identification of the causal effect [22].

5.2.3 Theory of Identification with Negative Control Variables (NC)

In this section, we present the sufficient assumptions for identification, using two negative control variables following [22]. Specifically, we use one *Negative Control Outcome* (NCO) and one *Negative Control Exposure* (NCE) as defined below in the following assumptions. We will start by giving an overview of the background for the *Proximal Causal Inference*. We should note that the following theoretical analysis was proven in [22] in

the context of a causal inference problem, and we here adopt it for the special case of recommendation taking the ratings as outcomes.

As discussed in the previous section, the conditional ignorability assumption fails in the presence of unobserved confounder (V). This assumption is relaxed by making the *latent ignorability assumption* 5.2.2 [22]:

Assumption 5.2.2. *(Latent ignorability):* $(y(a_{u,i} = 1), y(a_{u,i} = 0)) \perp\!\!\!\perp A|X, V$

Assumption 5.2.2 states that the user and item features X and the unobserved confounder V are sufficient to capture the confounding between the potential ratings $y(a_{u,i})$ and the exposure variable A. Note that the latent ignorability assumption has no restrictions about the nature of the unobserved confounder V. This assumption, which is a weaker assumption than the conditional ignorability assumption, leads to Equation 5.2 which states that the potential rating can be recovered by conditioning on the observed covariates X, the unobserved confounder V and the exposure variable A [22]:

$$E[Y(a_{u,i})] = E[E(Y|X, V, A = a_{u,i})] \tag{5.2}$$

The proposed approach for handling unmeasured confounders relies on the presence of two negative control variables that satisfy the following conditions. The first variable is called *negative control outcome* and can be defined as follows [22]:

Assumption 5.2.3. *(Negative Control Outcome (NCO)): The NCO W is an auxiliary variable such that: $W \perp\!\!\!\perp A|X, V$.*

The Negative Control Outcome (NCO) is an *auxiliary* variable that is associated with the unmeasured confounder but is not causally affected by the exposure variable controlling for what items users are exposed to. An example of NCO is the item average rating *before* the user had rated it. Let us suppose that the user watched and rated the movie at instant t, this exposure could not not causally affect the item average rating at instant $t - 1$ because the future cannot causally affect the past. In the next section, we will formally define our choice of W.

The second type of auxiliary variables is the *negative control exposure* that needs to satisfy the following assumption [22]:

Assumption 5.2.4. *(Negative Control Exposure (NCE)): The NCO Z is an auxiliary variable such that: $Z \perp\!\!\!\perp Y|A, X, V$ and $Z \perp\!\!\!\perp W|A, X, V$.*

Assumption 5.2.4 states that the NCE is an auxiliary variable that does not affect the potential ratings and the NCO given the treatment, the latent confounder V and the user and item features X. For example, popularity can be considered an NCE. This is because popularity affects what the recommender system presents to the user as well as what items the user click on, but it does not affect the rating value (assuming in general that the user rates an item according to their own preference). In the next section, we will formally define our choice for NCE.

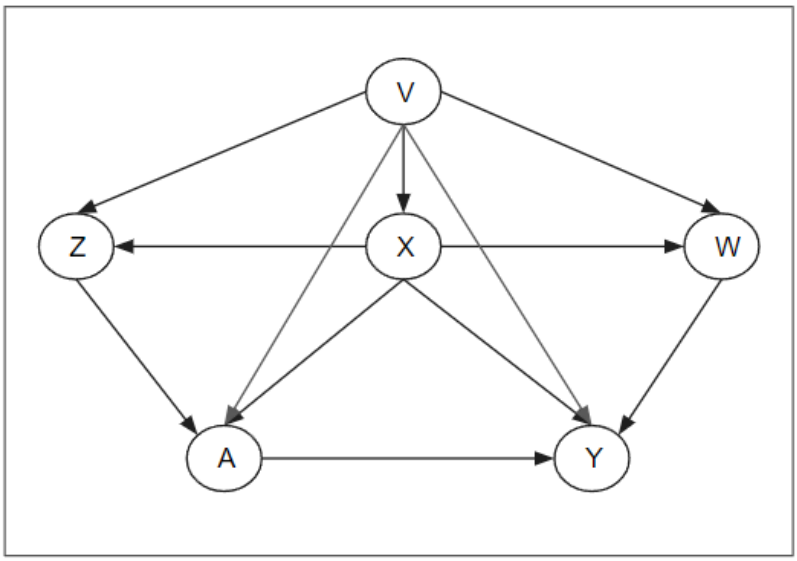


Figure 5.2. Proximal causal graph for a recommendation system where A denotes the exposure variable, Y denotes the outcome variable, X denotes the user and item covariates, V denotes the latent confounder, W denotes the negative control outcome (NCO),and Z denotes the negative control exposure (NCE).

Figure 5.2 shows a more complete causal graph for recommendation systems. The graph allows for the presence of observed covariates X (user and item features) as well as latent confounder V. Note that confounder V does not need to be specified since the theory does not require it. It is preferable to know the latent confounder to facilitate the choice

of the negative control variables, since the negative control variables need to be associated with the latent confounder. In this work, we did not specify the latent confounder.

Assuming a valid selection of NCO and NCE, then according to [22], there exists a function h, called the *outcome confounding bridge function*, relating the confounder's effects on the negative control outcome W to the confounder's effects on the potential rating Y, as shown in the following assumption.

Assumption 5.2.5. *(Outcome Confounding Bridge): there exists a function $h(W, a_{u,i}, X)$ such that for $a_{u,i} \in \{1, 0\}$:*

$$E[Y|A = a_{u,i}, X, V] = E[h(W, a_{u,i}, X)|A = a_{u,i}, X, V)]$$

Assumption 5.2.5 claims that the effect of confounder V on the ratings Y is equal to the effect of confounder V on a transformation h of W. This assumption entails that the NCO W captures the variability in the confounder V. Given assumption 5.2.2 - 5.2.3 - 5.2.5, the potential ratings' mean can be identified as shown in proposition 5.2.6 [22].

Proposition 5.2.6. *Under assumption 5.2.2 - 5.2.3 - 5.2.5, for $a_{u,i} \in \{1, 0\}$ we have:*

$$E[Y(a_{u,i})] = E[h(W, a_{u,i}, X)] \tag{5.3}$$

Proof. Using Assumption 5.2.5, we have:

$$E[Y|A = a_{u,i}, XV] = E[h(W, a_{u,i}, X)|A = a_{u,i}, X, V)]$$

Therefore using an expectation over V, we obtain

$$E(E[Y|A = a_{u,i}, X, V]) = E(E[h(W, a_{u,i}, X)|A = a_{u,i}, X, V)])$$

Using Assumption 5.2.2 which entailed Eq. 5.2, we have

$$E(E[Y|A = a_{u,i}, X, V]) = E(E(Y(a_{ui})|V)) = E(Y(a_{ui}))$$

Using Assumption 5.2.3 we obtain:

$$E(E(h(W, a_{ui}))|X, A = a_{ui}) = E(E(h(W, a_{ui})|V)) = E(h(W, a_{ui}))$$

We finally get our result:

$$E[Y(a_{u,i})] = E[h(W, a_{u,i}, X)]$$

□

What Proposition 5.2.6 suggests is that it is enough to know $h(W, a_{ui}, X)$ in order to determine the effect $E(Y(a_{ui}))$ without additional assumptions. Mainly, it is enough to learn a transformation of the NCO variable and use it to determine $E(Y(a_{ui}))$. The remaining challenge is identifying the bridge function h. Without access to the unobserved confounders, one cannot totally identify h. We thus use another proxy variable NCE Z in order to identify h. In fact, [22] proposed a framework for identifying the bridge function using the NCE variable. According to [22], the effect of the NCE variable Z on the ratings Y is the same as the effect of Z on a transformation h of W thanks to Assumption 5.2.4. Therefore we get

$$E(Y|Z, A) = E(h(W, A)|Z, A). \tag{5.4}$$

It was further proven in [22] that h is a unique solution to Equation 5.4. We use this result in order to learn the function h.

To summarize, our goal is to learn the potential ratings $\hat{Y}$ under an unobserved confounder V. The problem is that the unobserved confounder cannot be correctly measured. Therefore, we use a bridge function h to estimate its effect on Y. The bridge function is a transformation of the Negative Control Outcome (NCO) variable W that also depends on the treatment A. To identify the bridge function h, we use a NCE variable Z. In fact, Equation 5.4 allows us to learn the function h in an unbiased way. We summarize all the steps of our approach in Figure 5.3.

Note that the identification process does not put any restrictions on the bridge function type. In other words, the function h can be parametric, semi-parametric, or non-parametric. In the next section, we will discuss an estimation protocol of the potential

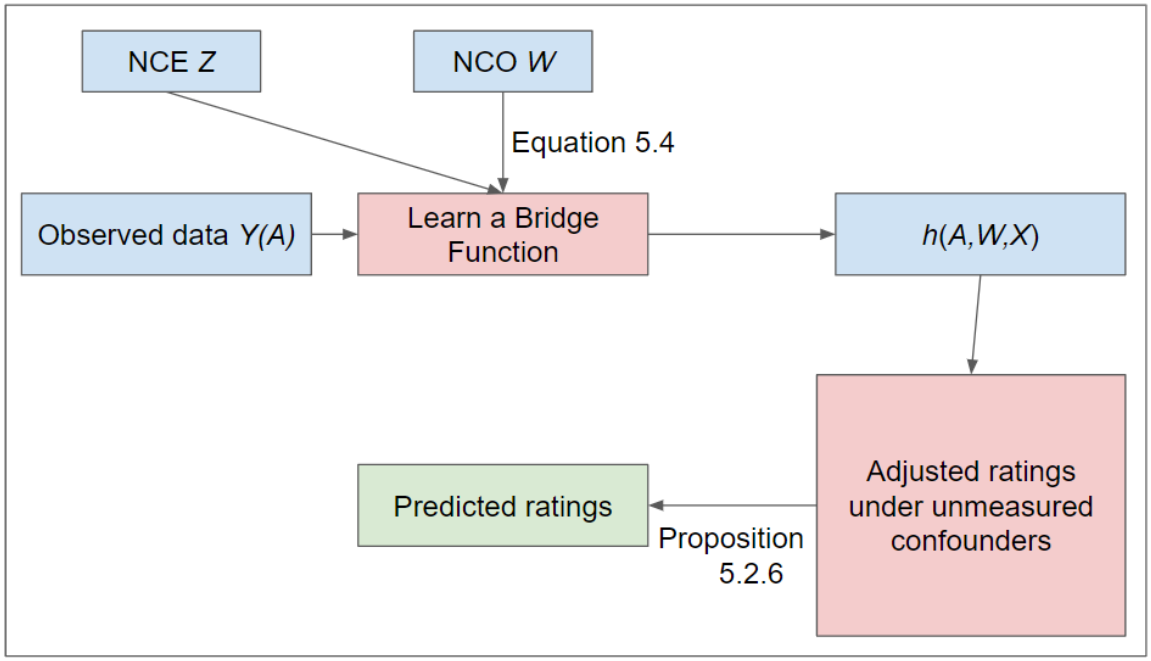


Figure 5.3. Pipeline for learning the de-confounded ratings. We start by learning a bridge function using NCO and NCE. Then we use the predicted ratings from the bridge function h to learn unbiased ratings

rating for matrix factorization. Note that our estimation approach is valid for the majority of recommendation algorithms, which makes our approach modular and easy to implement.

5.2.4 Bridge Matrix Factorization

In this section, we propose to debias matrix factorization using negative control variables and a bridge function in order to adjust for unobserved confounders. We start by presenting our choice for the negative control variables and the bridge function h. As mentioned before, the theory does not put any restrictions on the choice of the bridge function. Finally, we propose a two-stage approach to estimate the bridge function and the (adjusted) rating outcomes $\hat{y}(a_{u,i} = 1)$.

5.2.4.1 Negative Control Variables

We propose a potential choice for the Negative Control Outcome (NCO) and the Negative Control Exposure (NCE) adhering to Assumption 5.2.3 and Assumption 5.2.4

respectively. According to Assumption 5.2.3, NCO is a variable W that directly affects the rating Y and is *not* causally affected by the exposure A. In the recommender system context, we can set W to be the user's average rating at $t-1$. This is because the exposure at t does not affect the user's average rating at time $t-1$, as required by the causal independence of the past on the future. The NCO can therefore be computed using Equation 5.5:

$$w_u^t = \frac{\sum_{k=1}^{t-1} y_u^k}{n_{t-1}}, \tag{5.5}$$

where w_u^t stands for NCO at time t for user u, y_u^k stands for the user's rating at time stamp k, and n_{t-1} stands for the number of items that the user has rated up until timestamp $t-1$.

We now propose a potential choice for the Negative Control Exposure (NCE) Z. According to Assumption 5.2.4, Z is a variable that affects the exposure variable A, which denotes the set of items that are exposed (via recommendations) to the user. By definition of the NCE, Z should be causally independent of the outcome variable Y (potential rating) and the NCO W given the observed and unobserved confounders. However it has to causally affect the exposure A (the set of items recommended or *exposed* to the user up to time t). The average popularity of the items can be used as a Negative Control Exposure (NCE). In fact, as proven in Chapter 3, the more popular the item is, the more likely it is to be recommended (hence exposed to the user). Thus the popularity affects the exposure variable A which controls what items the users are exposed to. Also, the more popular the item is, the more likely that the user had been exposed to it either through a recommendations or from the user's social circle. Furthermore, we assume that the popularity of the items does not causally affect the user's individual rating on the item because we can argue (and assume) that *after* the user watches a movie, the typical user's rating will be a reflection of their own interest and satisfaction with the movie. The Negative Control Exposure z_i can therefore be defined using Equation 5.6:

$$z_i = \sum_{u \in U} \mathbb{1}_{y_{u,i}>0}, \tag{5.6}$$

since the popularity for each item is defined by the number of ratings, i.e., $(y_{u,i} > 0)$ an item

gets from all users. Popularity over time can also be used to capture the time dependency in the exposure. However, in this work, we focus on the popularity in a static setting.

5.2.4.2 Bridge Function

After defining the negative control outcome W and the negative control exposure Z, We introduce our choice for the bridge function. As we have mentioned, the theory does not put any restrictions on the bridge function, which can be parametric, semi-parametric, or non-parametric. In this work, we propose a parametric bridge function using Neural Networks. The adjusted ratings $y_{u,i}^{bridge}(a)$ can thus be computed using Equation 5.7, where x_u and x_i are the user and item latent features, respectively; $w_{u,i}$ and $z_{u,i}$ denote the NCO and NCE, respectively,; and a is the boolean exposure:

$$y_{u,i}^{bridge}(a) = h(W, Z, X, A) = max(0, x_u^T.x_i.a + \alpha w_{u,i} + \lambda z_{u,i} + \beta_0) \tag{5.7}$$

Figure 5.4 shows the overall architecture of the bridge model.

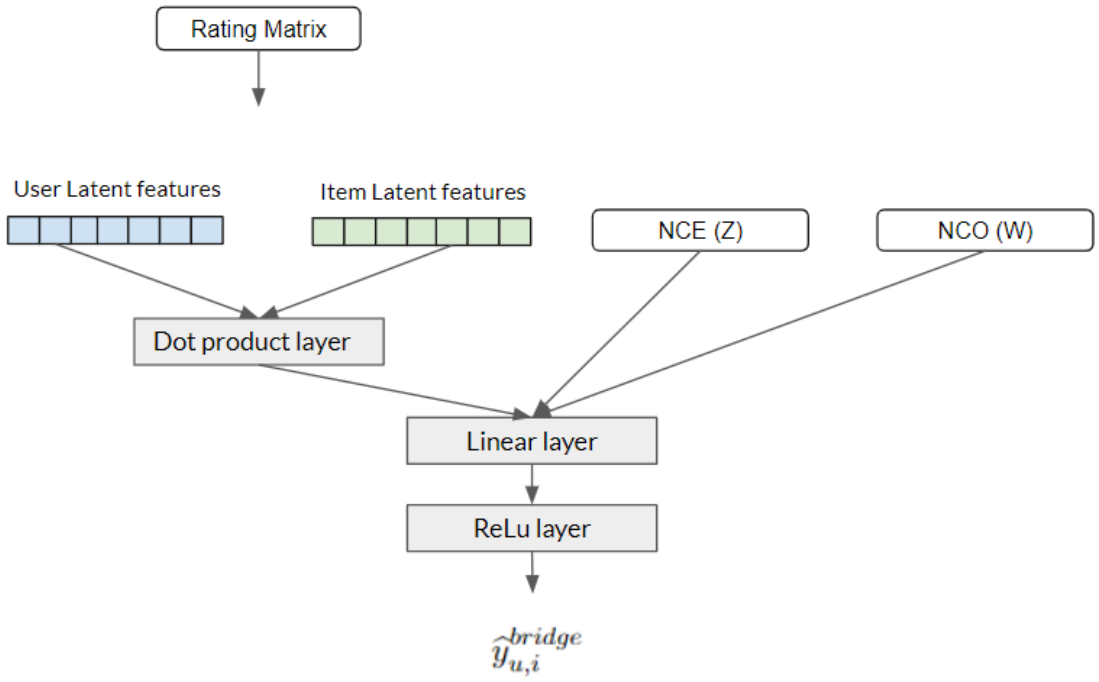


Figure 5.4. Bridge Function Network Architecture.

In order to learn the bridge function h, we minimize the following loss function:

$$L^{bridge}(\widehat{y}_{u,i}^{bridge}) = \frac{1}{|U||I|} \sum_{u,i \in U,I} (\widehat{h}(W, Z, X, A) - Y_{ui})^2 \tag{5.8}$$

It is easy to notice that the loss defined in Equation 5.8 will yield an unbiased (least squares) estimate of the outcome (rating) Y_{ui} with respect to the unmeasured confounder V since $E(h(A, W|Z)) = E(Y|Z, A))$

5.2.4.3 Matrix Factorization using Negative Control Variables and a Bridge Function

In the final stage of building the debiased recommendation model, and after generating $\hat{y}_{ui}^{bridge}$ using the procedure described in the previous section, we use the learned output ratings as input to train a Matrix Factorization model. The bridge function therefore serves as an intermediate data that generates a training data that is more robust to confounders, thanks to including the NCO and NCE variables during the learning process. These learned ratings will then serve for learning a more robust final model that takes into account the effect of the unknown causal confounders V from the training data ($\hat{y}_{ui}^{bridge}$), as shown in Figure 5.3. We present the final debiased recommendation learning procedure in Algorithm 5.1

Algorithm 5.1 Bias correction learning using a causal bridge function

Input: Rating matrix Y, NCO W, NCE Z
Output: Predicted ratings $\hat{Y}$

Learn bridge function h by minimizing the loss in equation 5.8
Generate adjusted ratings (outcomes) using the bridge function: $\hat{Y}_{ui}^{bridge} = h(Y, W, Z)$
Learn a Matrix Factorization (MF) model $\mathbf{P}, \mathbf{Q}$ by training on adjusted ratings $\hat{Y}_{ui}^{bridge}$
Generate final predicted ratings $\hat{Y_{ui}}$ using the MF model's latent factor parameters: $\hat{Y_{ui}} = \mathbf{p}_u \cdot \mathbf{q}_i$
return $\hat{Y_{ui}}$

5.3 Experimental Results

In this section, we evaluate the performance of our proposed approaches. The first approach focuses on adjusting for the unobserved confounders assuming the data is inde-

pendent and identically distributed i.i.d. Then, we relax this assumption by using the IPS estimator to account for the exposure bias. We test our method on real-world datasets and compare its performance to widely used baselines. We design our experiments to evaluate the effectiveness of our proposed approach at providing unbiased recommendations. The research questions we aim to answer are:

- **RQ1:** How does our proposed approach for adjusting for unobserved confounders compare to other state of the art approaches in terms of providing unbiased recommendations?
- **RQ2:** Does adjusting for observed confounders and unobserved confounders further improve the results? This can be considered as treating two biases simultaneously.

In the following, we start by describing the experimental settings, including the evaluation protocol, data used, and the baseline methods. Then, we present our experimental results in order to answer RQ1 and RQ2.

5.3.1 Experimental Setting

5.3.1.1 Evaluation Protocol in a Causal Setting

A classic recommender system evaluation protocol is to split the rating data randomly into training and test sets. The performance metrics of the recommendation are then reported on the test data by computing the average metric over all users. The problem with this standard methodology is the test data is biased which in turn yields a biased evaluation. In fact, the test data is biased towards popular items and active users. In addition, the test data hides inherent biases underlying the training data itself. In order to have an unbiased estimation of performance, we need to have access to *all* potential outcomes, which is not feasible in a recommendation system setting. One way to mitigate this problem is to have a random test data, where the users are provided with a randomly selected subset of items, that they need to rate to provide the ground truth ratings for evaluation. The latter approach, ensures that the test data is collected randomly and therefore the

average over all users of the performance metrics will be unbiased. However, such a random test data is expensive to collect as part of a recommendation system platform in the real world, since it will hurt the user experience. That said, we found two public datasets that include a training data that has been collected from the organic user interaction with the recommendation system and a test data that has been collected randomly. These datasets are the Yahoo!R3 dataset [126] and the coat shopping dataset [100].

5.3.1.2 Datasets

We used two publicly available datasets with a test data collected randomly to evaluate the performance of our models:

- **Yahoo!R3 [126]:** Yahoo!R3 is a music rating dataset with ratings from 1 to 5. The training dataset contains over 300k song ratings collected from the user interaction with the Yahoo music recommendation service. This means that the training ratings are self-selected by the users. The training data contains 15400 users and 1000 items. The test data was collected by asking 5400 users to rate 10 randomly chosen songs. The test data thus contains 54000 ratings. The data was collected during a survey for research purposes.

- **Coat [100]:** Coat is a rating shopping dataset. The data contains ratings for 290 users and 300 items. The training data consists of 7000 ratings. Each user was asked to rate 24 coats from the inventory. The user gets to choose what items to rate based on their self exploration. However, for the test data collection, each user was asked to rate 16 randomly picked items.

Table 5.2 summarizes the statistics of both datasets for the training and test data.

5.3.1.3 Evaluation Metrics

Our goal is to measure the performance of our models on two tasks: the rating prediction performance and the ranking performance. In order to predict the rating prediction

TABLE 5.2

Dataset Statistics

Data	Users	Items	Interactions
Yahoo!R3 - Train	15400	1000	300k
Yahoo!R3 - Test	5400	1000	54000
Coat - Train	290	300	6960
Coat - Test	290	300	4640

performance, we compute the Mean Square Error (MSE) defined in Equation 5.9. The lower the MSE the better.

$$MSE(R, \widehat{R}) = \frac{\sum_{(u,i)\in O} \left(r_{u,i} - \widehat{r}_{u,i}\right)^2}{|R|} \tag{5.9}$$

In order to evaluate the quality of the recommended lists' ranking, we use the following metrics: Normalized Discounted Cumulative Gain (NDCG@K) (see Equation 3.13), Precision@k (see Equation 3.14), and Recall@K (see Equation 3.15). The higher these ranking metrics, the better the ranking quality.

5.3.1.4 Baselines

We compare our method to Matrix Factorization (MF) [57] and two state of the art debiasing frameworks: Matrix Factorization with Inverse Propensity weights (IPS-MF) [100], and the deconfounded Matrix Factorization [120].

- **Matrix Factorization (MF)** [57]: Matrix factorization is a popular CF technique that learns an embedding of users and items by factoring the rating matrix into two embedding vectors. It uses the dot product to map the embedding to predicted ratings.
- **IPS-MF** [100]: Inverse propensity Matrix factorization is an unbiased MF model to correct for selection bias.
- **Deconfounded MF [120]**: Deconfounded MF is a debiasing technique relying on estimating a substitute for the unobserved confounders.

5.3.1.5 Experimental set-up

For both data sets, we split the training data into 90% train and 10% validation. Then the hyper-parameter tuning was done using the 10% validation data, for each evaluated approach, including the baselines, using Optuna [105]. We varied the learning rate between [0.001, 0.1]. The embedding dimension sizes were varied in {8, 16, 32, 64, 128} for matrix factorization, and the L2 regularization weight varied between [0.0001, 0.1]. For the optimizer, we chose between Adam and the Adaptive Gradient Algorithm (Adagrad). We varied the batch size in {64, 128, 256, 512}. We finally selected the hyper-parameters based on the evolution of the mean square error (RMSE) on the validation dataset.

5.3.2 Performance Evaluation

In this section, we will answer the two research questions **RQ1** and **RQ2**. We report our performance results on the Yahoo!R3 and Coat datasets. The convergence of the algorithm is declared when the RMSE on the validation set no longer decreases for five epochs. We run each experiment 5 times and we report the mean and the standard deviation.

TABLE 5.3

Accuracy performance of Bridge MF compared to different baselines on Yahoo!R3. We observe that our model has lower MSE and higher NDCG, while having comparable performance in terms of Precision and Recall.

Dataset	Yahoo!R3			
Metric	MSE	Precision @5	Recall@5	NDCG@5
MF [57]	1.98 +/- 0.002	0.02 +/- 0.0006	0.56 +/- 0.011	0.46 +/- 0.019
IPSMF [100]	1.87 +/- 0.008	0.034 +/- 7e-05	0.71 +/- 0.002	0.5 +/- 0.007
MF+Bridge	1.34 +/- 0.0003	0.034 +/- 1.5 e-05	**0.71 +/- 0.0001**	**0.55 +/- 5e-05**
IPSMF+Bridge	**1.29 +/- 0.0005**	0.034 +/- 9e-05	0.7 +/- 0.0017	**0.55 +/- 0.0004**

RQ1: How does our proposed approach for adjusting for unobserved confounders compare to other state of the art approaches in terms of providing unbiased recommendations? Table 5.3 and Table 5.4 shows the performance of Bridge

Matrix Factorization compared to the baselines. We reported the results in two different tables to add the deconfounded recommender system baseline. In fact, we reported the results reported in the original paper [120], where the precision was not reported and NDCG was computed differently from our work. More specifically, they introduced a threshold to determine whether an item is relevant or not. If $rating > 3$, the item's relevance is the rating; otherwise the relevance is set to 0. In our case, we do not introduce a threshold since, we argue that it introduces bias, and we instead use the rating as relevance without thresholding. For example, if an item is rated 1, then the relevance is 1; and if the item is rated 5, then the relevance is 5. Table 5.3 and table 5.4 show that Bridge MF outperforms the baselines in terms of MSE, Recall@5 and NDCG@5, while shows comparable performance in terms of Precision@5. Note that the Yahoo!R3 test set is not affected by confounders, since the data was collected randomly. Bridge MF provides more accurate recommendations on this unbiased test dataset, which means that our model adjusts better for confounders. This is to be expected, since IPSMF relies on the propensity estimation that depends on the collected *observed* confounders. In contrast, Bridge MF relies on the same *observed* confounders, but it also assumes the presence of an *unmeasured* confounder that may affect the ratings, namely the exposure and the observed confounders. If the MSE does not improve, then this would mean that we do not have unmeasured confounders in our system. Note that the MSE reported in our results on the Yahoo!R3 data is different from the one reported in [100], since the propensities were estimated using Naive Bayes, which in turn requires the presence of unbiased data. In our experiments, we did not use an *unbiased* data to estimate the propensities since in the real world scenario, it is unlikely to have access to unbiased data. Table 5.5 and Table 5.4 show the performance of our proposed approach on the coat dataset. Bridge MF showed a better performance compared to all the baselines. Thus the bridge function enables a better rating prediction accuracy and better ranking performance. For the IPSMF, we used the propensities provided in the data.

TABLE 5.4

Accuracy performance of Bridge MF compared to different baselines on the Coat Dataset. We observe that our model has lower MSE and higher NDCG, Precision, and Recall.

Dataset	Yahoo!R3		Coat	
Metrics	MSE	Recall@5	MSE	Recall@5
MF [57]	1.98	0.56	1.21	0.56
IPSMF [100]	1.87	**0.71**	1.089	0.56
Deconfounded MF [120]	1.768	0.64	1.341	0.569
MF+Bridge	1.34	**0.71**	**1.00**	**0.61**
IPSMF+Bridge	**1.29**	0.7	**1.00**	0.6

TABLE 5.5

Accuracy performance of Bridge MF compared to different baselines on the Coat Dataset. We observe that our model has lower MSE and higher NDCG, Precision, and Recall.

Dataset	Coat			
Metric	MSE	Precision @5	Recall@5	NDCG@5
MF [57]	1.21 +/- 0.02	0.076 +/- 0.002	0.56 +/- 0.014	0.54 +/- 0.002
IPSMF [100]	1.08 +/- 0.009	0.077 +/- 0.005	0.56 +/- 0.005	0.53 +/- 0.002
MF+Bridge	**1.00 +/- 0.0006**	**0.08 +/- 0.0015**	**0.61 +/- 0.015**	**0.56 +/- 0.0006**
IPSMF+Bridge	**1.00 +/- 0.005**	0.078 +/- 0.002	0.6 +/- 0.025	**0.56 +/- 0.002**

RQ2: Does adjusting for observed confounders and unobserved confounders further improve the results? This can be seen as treating two biases simultaneously. In the next experiment, we used the IPS estimator to estimate the predicted ratings in the second stage of Bridge MF. According to the results in Tables 5.3, 5.4, and 5.5, using the IPS estimator did improve the rating prediction, but did not improve the ranking quality of the recommender system. Our interpretation is that the adjusted ratings provided by the bridge model succeeded to adjust for the confounders. Hence adding an IPS estimator afterwards did not further improve the results.

5.4 Summary

In this chapter, we proposed a bias correction approach inspired from proximal causal inference. The approach relies on a pair of negative control variables to identify the causal effects of the exposure on the potential ratings, and this in turns allows adjusting or correcting the ratings for bias before using them for training a recommendation model. We evaluated our approach on two real-world datasets, and found that Bridge MF produced promising results by outperforming the state of the art models. One limitation of Bridge MF is the fact that it relies on a valid choice of negative control variables which is untestable and depends on the expert's judgment. Another limitation is that the choice of the bridge function inherits the model bias which may raise concerns about introducing bias in the results.

CHAPTER 6

CONCLUSION

The main goal of this dissertation is to propose novel debiasing approaches for recommendation systems, that go beyond current efforts in debiasing by addressing some of their known limitations. In Chapter 3, we introduced a novel approach to mitigate popularity bias in the data by disentangling relevance and user conformity. Our approach outperformed state of the art methods in terms of recommendation accuracy as well as reducing popularity bias and increasing the diversity in the recommended lists. We represented the conformity bias using learned embeddings, which allowed for a richer representation. Given that the users are affected by popular items differently, we learn a personalized user bias vector to capture the preferences of each user for popular items. This approach overcomes the limitation of using scalars to capture the conformity bias which is not accurate and does not capture user-item specific conformity levels. Our approach is modular and can be easily adapted to other collaborative filtering algorithms as base models. One limitation of our proposed approach is assuming that the recommender system is affected only by popularity bias. This assumption does not hold in real world recommendation systems.

In Chapter 4, we relaxed the assumption made in Chapter 3 and allow a recommender system to be affected by both exposure bias and popularity bias. We proposed a multi-bias framework that helps reduce the popularity and exposure bias simultaneously through incorporating the IPS training framework along with PCMF modeling, that we proposed in Chapter 3. Our proposed approach outperformed PCMF in terms of recommendation performance as well as decreasing popularity bias. Our experimental results showed the importance of correcting for multiple biases simultaneously.

In Chapter 5, we studied the recommender system from a causal perspective since it

improves the generalization and robustness of the model. We proposed a novel 2-stage debiasing approach inspired from proximal causal inference that aims at adjusting for observed and unobserved confounders. The approach leverages a pair of negative control variables to adjust for unobserved confounders. The fist stage is to learn the bridge function to adjust for observed and unobserved confounders. The second stage, consists of learning a matrix factorization model on the adjusted ratings. Our approach showed promising results by outperforming state of the art models. One limitation of bridge matrix factorization is the fact that it relies on the valid choice of negative control variables which is untestable and depends on the expert's judgment. Another limitation is that the choice of the bridge function inherits the model bias which may raise concerns about introducing bias in the results.

One area of expansion of our research is to study the iterative behavior of our proposed algorithms. In fact, recommender systems act in a dynamic environment and this aspect should be incorporated in the modeling of different debiasing approaches.

REFERENCES

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 263–272.

[2] Nishigandha Karbhari, Asmita Deshmukh, and Vinayak D. Shinde, "Recommendation system using content filtering: A case study for college campus placement," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 963–965.

[3] Robin Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[4] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He, "Bias and debias in recommender system: A survey and future directions," 2020.

[5] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel, "Unbiased learning-to-rank with biased feedback," *CoRR*, vol. abs/1608.04468, 2016.

[6] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft, "Unbiased learning to rank with unbiased propensity estimation," *CoRR*, vol. abs/1804.05938, 2018.

[7] Harald Steck, "Item popularity and recommendation accuracy," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, New York, NY, USA, 2011, RecSys '11, p. 125–132, Association for Computing Machinery.

[8] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Depeng Jin, and Yong Li, "Disentangling user interest and conformity for recommendation with causal embedding," 2021.

[9] Dawen Liang, Laurent Charlin, and David M. Blei, "Causal inference for recommendation," 2016.

[10] Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto, *Debiasing the Human-Recommender System Feedback Loop in Collaborative Filtering*, p. 645–651, Association for Computing Machinery, New York, NY, USA, 2019.

[11] Sami Khenissi and Olfa Nasraoui, "Modeling and counteracting exposure bias in recommender systems," 2020.

[12] Dietmar Jannach and Michael Jugovac, "Measuring the business value of recommender systems," *ACM Trans. Manage. Inf. Syst.*, vol. 10, no. 4, dec 2019.

[13] Hao Wang, Zonghu Wang, and Weishi Zhang, "Quantitative analysis of matthew effect and sparsity problem of recommender systems," in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2018, pp. 78–82.

[14] Sami Khenissi, Boujelbene Mariem, and Olfa Nasraoui, "Theoretical modeling of the iterative properties of user discovery in a collaborative filtering recommender system," in *Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020, RecSys '20, p. 348–357, Association for Computing Machinery.

[15] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli, "Degenerate feedback loops in recommender systems," *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, Jan. 2019.

[16] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher, "The unfairness of popularity bias in recommendation," *CoRR*, vol. abs/1907.13286, 2019.

[17] Himan Abdollahpouri and Masoud Mansoury, "Multi-sided exposure bias in recommendation," *CoRR*, vol. abs/2006.15772, 2020.

[18] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims, "Recommendations as treatments: Debiasing learning and evaluation," *CoRR*, vol. abs/1602.05352, 2016.

[19] Khalil Damak, Sami Khenissi, and Olfa Nasraoui, "Debiased explainable pairwise ranking from implicit feedback," *CoRR*, vol. abs/2107.14768, 2021.

[20] Andrew Ying, Wang Miao, Xu Shi, and Eric J. Tchetgen Tchetgen, "Proximal causal inference for complex longitudinal studies," 2021.

[21] Eric J Tchetgen Tchetgen, Andrew Ying, Yifan Cui, Xu Shi, and Wang Miao, "An introduction to proximal causal learning," 2020.

[22] Wang Miao, Xu Shi, and Eric Tchetgen Tchetgen, "A confounding bridge approach for double negative control inference on causal effects," 2018.

[23] Peter Boström and Melker Filipsson, "Comparison of user based and item based collaborative filtering recommendation services," 2017.

[24] Y. Koren, R. Bell, and Ch. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer 42, 8*, pp. 30–37, 2009.

[25] Robert M. Bell and Yehuda Koren, "Lessons from the netflix prize challenge," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 75–79, dec 2007.

[26] Sebastian Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[27] Amy Nicole Langville, Carl Dean Meyer, Russell Albright, James Cox, and David Duling, "Algorithms, initializations, and convergence for the nonnegative matrix factorization," *CoRR*, vol. abs/1407.7299, 2014.

[28] Neil Houlsby, Jose Miguel Hernandez-Lobato, and Zoubin Ghahramani, "Cold-start active learning with robust ordinal matrix factorization," in *Proceedings of the 31st International Conference on Machine Learning*, Eric P. Xing and Tony Jebara, Eds., Bejing, China, 22–24 Jun 2014, vol. 32 of *Proceedings of Machine Learning Research*, pp. 766–774, PMLR.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, USA, 2009, UAI '09, p. 452–461, AUAI Press.

[30] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin, "Simplify and robustify negative sampling for implicit collaborative filtering," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 1094–1105, Curran Associates, Inc.

[31] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin, "Simplify and robustify negative sampling for implicit collaborative filtering," 2020.

[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, New York, NY, USA, 2001, WWW '01, p. 285–295, Association for Computing Machinery.

[33] Hailong Wen, Guiguang Ding, Cong Liu, and Jianming Wang, "Matrix factorization meets cosine similarity: Addressing sparsity problem in collaborative filtering recommender system," in *Web Technologies and Applications*, Lei Chen, Yan Jia, Timos Sellis, and Guanfeng Liu, Eds., Cham, 2014, pp. 306–317, Springer International Publishing.

[34] Ludovico Boratto and Salvatore Carta, "Using collaborative filtering to overcome the curse of dimensionality when clustering users in a group recommender system," in *Proceedings of the 16th International Conference on Enterprise Information Systems - Volume 2*, Setubal, PRT, 2014, ICEIS 2014, p. 564–572, SCITEPRESS - Science and Technology Publications, Lda.

[35] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua, "Neural collaborative filtering," 2017.

[36] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in *Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020, RecSys '20, p. 240–248, Association for Computing Machinery.

[37] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo, "Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations," 2020.

[38] Dongqing Liu and Gurbir Singh, "A recurrent neural network based recommendation system," 2016.

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," 2019.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[41] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*, New York, NY, USA, 2015, WWW '15 Companion, p. 111–112, Association for Computing Machinery.

[42] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara, "Variational autoencoders for collaborative filtering," 2018.

[43] Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei, "Avoiding latent variable collapse with generative skip models," 2019.

[44] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li, "Sequential recommendation with graph neural networks," 2021.

[45] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui, "Graph neural networks in recommender systems: A survey," 2021.

[46] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec, "Graph convolutional neural networks for web-scale recommender systems," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, Jul 2018.

[47] Jinbo Song, Chao Chang, Fei Sun, Xinbo Song, and Peng Jiang, "Ngat4rec: Neighbor-aware graph attention network for recommendation," *CoRR*, vol. abs/2010.12256, 2020.

[48] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro, "Content-based recommender systems: State of the art and trends," *Recommender systems handbook*, pp. 73–105, 2011.

[49] Benedikt Schifferer, Gilberto Titericz, Chris Deotte, Christof Henkel, Kazuki Onodera, Jiwei Liu, Bojan Tunguz, Even Oldridge, Gabriel De Souza Pereira Moreira, and Ahmet Erdem, "Gpu accelerated feature engineering and training for recommender systems," in *Proceedings of the Recommender Systems Challenge 2020*, New York, NY, USA, 2020, RecSysChallenge '20, p. 16–23, Association for Computing Machinery.

[50] Longteng Xu, Jiwei Liu, and Yu Gu, "A recommendation system based on extreme gradient boosting classifier," in *2018 10th International Conference on Modelling, Identification and Control (ICMIC)*, 2018, pp. 1–5.

[51] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2016, KDD '16, p. 785–794, Association for Computing Machinery.

[52] Christopher J. C. Burges, Krysta M. Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu, "Learning to rank using an ensemble of lambda-gradient models," in *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14*. 2010, YLRC'10, p. 25–35, JMLR.org.

[53] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao, "Improving deep learning for airbnb search," 2020.

[54] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, New York, NY, USA, 2020, KDD '20, p. 2311–2320, Association for Computing Machinery.

[55] Erik Bernhardsson, *Annoy: Approximate Nearest Neighbors in C++/Python*, 2018, Python package version 1.13.0.

[56] Sanjeevan Sivapalan, Alireza Sadeghian, Hossein Rahnama, and Asad M. Madni, "Recommender systems in e-commerce," in *2014 World Automation Congress (WAC)*, 2014, pp. 179–184.

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

[58] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He, *Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System*, p. 1791–1800, Association for Computing Machinery, New York, NY, USA, 2021.

[59] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay, "Accurately interpreting clickthrough data as implicit feedback," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2005, SIGIR '05, p. 154–161, Association for Computing Machinery.

[60] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei, "Modeling user exposure in recommendation," in *Proceedings of the 25th International Conference on World Wide Web*, Republic and Canton of Geneva, CHE, 2016, WWW '16, p. 951–961, International World Wide Web Conferences Steering Committee.

[61] Wei Ma and George H Chen, "Missing not at random in matrix completion: The effectiveness of estimating missingness probabilities under a low nuclear norm assumption," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 14871–14880. Curran Associates, Inc., 2019.

[62] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi, "Doubly robust joint learning for recommendation on data missing not at random," Long Beach, California, USA, 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6638–6647, PMLR.

[63] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft, "Unbiased learning to rank with unbiased propensity estimation," in *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, New York, NY, USA, 2018, SIGIR '18, p. 385–394, Association for Computing Machinery.

[64] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4511–4515, 2010.

[65] Natali Helberger, Kari Karppinen, and Lucia D'Acunto, "Exposure diversity as a design principle for recommender systems," *Information, Communication & Society*, vol. 21, no. 2, pp. 191–207, 2018.

[66] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra, "Explore, exploit, and explain: Personalizing explainable recommendations with bandits," in *Proceedings of the 12th ACM Conference on Recommender Systems*, New York, NY, USA, 2018, RecSys '18, p. 31–39, Association for Computing Machinery.

[67] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, ICML '08, p. 784–791, Association for Computing Machinery.

[68] Chang Li, Haoyun Feng, and Maarten de Rijke, "Cascading hybrid bandits: Online learning to rank for relevance and diversity," in *Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020, RecSys '20, p. 33–42, Association for Computing Machinery.

[69] Javier Parapar and Filip Radlinski, "Diverse user preference elicitation with multi-armed bandits," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2021, WSDM '21, p. 130–138, Association for Computing Machinery.

[70] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li, "Drn: A deep reinforcement learning framework for news recommendation.," in *WWW*, Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, Eds. 2018, pp. 167–176, ACM.

[71] Yong Liu, Yinan Zhang, Qiong Wu, Chunyan Miao, Lizhen Cui, Binqiang Zhao, Yin Zhao, and Lu Guan, "Diversity-promoting deep reinforcement learning for interactive recommendation," 2019.

[72] Olivier Jeunen, David Rohde, Flavian Vasile, and Martin Bompaire, "Joint policy-value learning for recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, New York, NY, USA, 2020, KDD '20, p. 1223–1233, Association for Computing Machinery.

[73] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, New York, NY, USA, 2017, RecSys '17, p. 42–46, Association for Computing Machinery.

[74] Himan Abdollahpouri, Robin D. Burke, and Bamshad Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," *ArXiv*, vol. abs/1901.07555, 2019.

[75] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac, "What recommenders recommend: an analysis of recommendation biases and possible countermeasures," *User Modeling and User-Adapted Interaction*, vol. 25, no. 5, pp. 427–491, July 2015.

[76] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay, "Accurately interpreting clickthrough data as implicit feedback," *SIGIR Forum*, vol. 51, no. 1, pp. 4–11, aug 2017.

[77] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay, "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search," *ACM Trans. Inf. Syst.*, vol. 25, no. 2, pp. 7–es, apr 2007.

[78] Katja Hofmann, Anne Schuth, Alejandro Bellogín, and Maarten de Rijke, "Effects of position bias on click-based recommender evaluation," in *Advances in Information Retrieval*, Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann, Eds., Cham, 2014, pp. 624–630, Springer International Publishing.

[79] Nicolò Felicioni, Maurizio Ferrari Dacrema, and Paolo Cremonesi, "Measuring the user satisfaction in a recommendation interface with multiple carousels," 05 2021.

[80] Olivier Chapelle and Ya Zhang, "A dynamic bayesian network click model for web search ranking," in *Proceedings of the 18th International Conference on World Wide Web*, New York, NY, USA, 2009, WWW '09, p. 1–10, Association for Computing Machinery.

[81] Georges E. Dupret and Benjamin Piwowarski, "A user browsing model to predict search engine click data from past observations.," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2008, SIGIR '08, p. 331–338, Association for Computing Machinery.

[82] Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Shuai Li, Ruiming Tang, Xiuqiang He, Jianye Hao, and Yong Yu, *A Graph-Enhanced Click Model for Web Search*, p. 1259–1268, Association for Computing Machinery, New York, NY, USA, 2021.

[83] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork, "Position bias estimation for unbiased learning to rank in personal search," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, 2018, pp. 610–618.

[84] Paul Covington, Jay Adams, and Emre Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

[85] Ayan Sinha, David F. Gleich, and Karthik Ramani, "Deconvolving feedback loops in recommender systems," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, Eds., 2016, pp. 3243–3251.

[86] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei, *Causal Inference for Recommender Systems*, p. 426–431, Association for Computing Machinery, New York, NY, USA, 2020.

[87] Mengyue Yang, Quanyu Dai, Zhenhua Dong, Xu Chen, Xiuqiang He, and Jun Wang, "Top-n recommendation with counterfactual user preference simulation," *CoRR*, vol. abs/2109.02444, 2021.

[88] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He, "Bias and debias in recommender system: A survey and future directions," 2020.

[89] Andrew Collins, Dominika Tkaczyk, Akiko Aizawa, and Jöran Beel, "A study of position bias in digital library recommender systems," *CoRR*, vol. abs/1802.06565, 2018.

[90] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang, "Causal intervention for leveraging popularity bias in recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. jul 2021, ACM.

[91] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin, "Disentangling user interest and popularity bias for recommendation with causal embedding," *CoRR*, vol. abs/2006.11011, 2020.

[92] Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu, "Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation," *CoRR*, vol. abs/2109.07946, 2021.

[93] Shuyuan Xu, Juntao Tan, Shelby Heinecke, Jia Li, and Yongfeng Zhang, "Deconfounded causal collaborative filtering," *CoRR*, vol. abs/2110.07122, 2021.

[94] Dugang Liu, Pengxiang Cheng, Hong Zhu, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming, *Mitigating Confounding Bias in Recommendation via Information Bottleneck*, p. 351–360, Association for Computing Machinery, New York, NY, USA, 2021.

[95] Yongkai Wu, Lu Zhang, and Xintao Wu, "Counterfactual fairness: Unidentification, bound and algorithm," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 7 2019, pp. 1438–1444, International Joint Conferences on Artificial Intelligence Organization.

[96] Yongkai Wu, Lu Zhang, and Xintao Wu, "On discrimination discovery and removal in ranked data using causal graph," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, New York, NY, USA, 2018, KDD '18, p. 2536–2544, Association for Computing Machinery.

[97] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang, "A survey of deep reinforcement learning in recommender systems: A systematic review and future directions," *CoRR*, vol. abs/2109.03540, 2021.

[98] Xiaoying Zhang, Junzhou Zhao, and John C.S. Lui, "Modeling the assimilation-contrast effects in online product rating systems: Debiasing and recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, New York, NY, USA, 2017, RecSys '17, p. 98–106, Association for Computing Machinery.

[99] Tien T. Nguyen, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan, "User personality and user satisfaction with recommender systems," *Information Systems Frontiers*, vol. 20, no. 6, pp. 1173–1189, dec 2018.

[100] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims, "Recommendations as treatments: Debiasing learning and evaluation," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. 2016, ICML'16, p. 1670–1679, JMLR.org.

[101] Stephen Bonner and Flavian Vasile, "Causal embeddings for recommendation," *CoRR*, vol. abs/1706.07639, 2017.

[102] Arda Antikacioglu, Tanvi Bajpai, and R. Ravi, "A new system-wide diversity measure for recommendations with efficient algorithms," 2019.

[103] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng, "Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison," in *Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020, RecSys '20, p. 23–32, Association for Computing Machinery.

[104] Gangan Elena, Kudus Milos, and Ilyushin Eugene, "Survey of multiarmed bandit algorithms applied to recommendation systems," *International Journal of Open Information Technologies*, vol. 9, no. 4, pp. 12–27, 2021.

[105] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR*, vol. abs/1907.10902, 2019.

[106] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[107] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[108] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[109] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel, "Unbiased learning-to-rank with biased feedback," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2017, WSDM '17, p. 781–789, Association for Computing Machinery.

[110] Mathieu Rouaud, "Probability, statistics and estimation," *Propagation of uncertainties*, vol. 191, 2013.

[111] Harald Steck, "Training and testing of recommender systems on data missing not at random," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2010, KDD '10, p. 713–722, Association for Computing Machinery.

[112] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei, "Modeling user exposure in recommendation," in *Proceedings of the 25th International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland, 2016, WWW '16, pp. 951–961, International World Wide Web Conferences Steering Committee.

[113] Saúl Vargas and Pablo Castells, "Rank and relevance in novelty and diversity metrics for recommender systems," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, New York, NY, USA, 2011, RecSys '11, p. 109–116, Association for Computing Machinery.

[114] Prem Gopalan, Jake M. Hofman, and David M. Blei, "Scalable recommendation with poisson factorization," 2014.

[115] Prem Gopalan, Laurent Charlin, and David M. Blei, "Content-based recommendations with poisson factorization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Cambridge, MA, USA, 2014, NIPS'14, pp. 3176–3184, MIT Press.

[116] Prem Gopalan, Francisco J.R. Ruiz, Rajesh Ranganath, and David M. Blei, "Bayesian nonparametric poisson factorization for recommendation systems," *Journal of Machine Learning Research*, vol. 33, pp. 275–283, 1 2014.

[117] Prem Gopalan, Jake M. Hofman, and David M. Blei, "Scalable recommendation with poisson factorization," *CoRR*, vol. abs/1311.1704, 2013.

[118] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 8024–8035. Curran Associates, Inc., 2019.

[119] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," *CoRR*, vol. abs/1205.2618, 2012.

[120] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei, "The deconfounded recommender: A causal inference approach to recommendation," *CoRR*, vol. abs/1808.06581, 2018.

[121] Wang Miao, Xu Shi, and Eric Tchetgen Tchetgen, "A confounding bridge approach for double negative control inference on causal effects," *arXiv preprint arXiv:1808.04945*, 2018.

[122] Donald B Rubin, "Causal inference using potential outcomes," *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 322–331, 2005.

[123] Judea Pearl, *Causality*, Cambridge University Press, 2 edition, 2009.

[124] PAUL R. ROSENBAUM and DONALD B. RUBIN, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, vol. 70, no. 1, pp. 41–55, 04 1983.

[125] Paul R. Rosenbaum and Donald B. Rubin, "Reducing bias in observational studies using subclassification on the propensity score," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 516–524, 1984.

[126] Benjamin M. Marlin and Richard S. Zemel, "Collaborative prediction and ranking with non-random missing data," in *Proceedings of the Third ACM Conference on Recommender Systems*, New York, NY, USA, 2009, RecSys '09, p. 5–12, Association for Computing Machinery.

CURRICULUM VITAE

NAME: Mariem Boujelbene

ADDRESS: Computer Science and Engineering Department
J.B Speed School of Engineering
University of Louisville
Louisville, KY 40292
United States of America.

EDUCATION:
Ph.D., Computer Science & Engineering, May 2022
University of Louisville, *Louisville, KY, USA.*
M.Sc. in Computer Science, May 2019
University of Louisville, *Louisville, KY, USA.*
B.Eng in Applied Mathematics, September 2017
Ecole Polytechnique de Tunisie, *Tunisia*

WORK EXPERIENCE:
Graduate Assistant, *University of Louisville*, KY, USA, 2017 - 2022
Machine Learning Engineer intern, *Linkedin*, New York, USA, May - August 2021

AWARDS:

CSE Doctoral Award, 2022

Dean Citation for academic excellence , 2019, 2022

Speed School Fellowship, 2020

CSE Masters Award, 2019

PUBLICATIONS:

1.**Mariem Boujelbene**, Khalil Damak, Asuman Cagla Acun Sener, Jeffrey Lloyd Hieb, Campbell R Bego, Patricia A Ralston, Olfa Nasraoui. 2020. A Data-science Approach to Flagging Non-retention in Engineering Enrollment Data. 2020 ASEE Virtual Annual Conference Content Access

2.Sami Khenissi, **Mariem Boujelbene**, Khalil Damak, Olfa Nasraoui. 2022. A New Attention Framework for Promoting Diverse Recommendations. Under Review. ACM SIGKDD 2022

3. Sami Khenissi, **Boujelbene Mariem**, and Olfa Nasraoui. 2020. Theoretical Modeling of the Iterative Properties of User Discovery in a Collaborative Filtering Recommender System. In Fourteenth ACM Conference on Recommender Systems (RecSys '20). Association for Computing Machinery, New York, NY, USA, 348–357.