5-2019

Modeling and counteracting exposure bias in recommender systems.

Sami Khenissi
University of Louisville

MODELING AND COUNTERACTING EXPOSURE BIAS IN RECOMMENDER SYSTEMS

By
Sami Khenissi
M.SC in Computer Science, 2019

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

Masters of Science
in Computer Science

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

May 2019

MODELING AND COUNTERACTING EXPOSURE BIAS IN RECOMMENDER SYSTEMS

By

Sami Khenissi
M.SC in Computer Science, 2019

Thesis approved on

April 25 2019

by the following Thesis Committee:

Thesis Director
Dr. Olfa Nasraoui

Dr. Hichem Frigui

Dr. Scott Sanders

DEDICATION

For all my loved ones.

ACKNOWLEDGMENTS

I would like to thank Dr Olfa Nasraoui for her support and guidance through this project. I would like also to thank the committee members for the time and attention they are providing to review my work. I would like to thank Wenlong Sun and Behnoush Abdollahi my labmates and friends who helped me with their expertise and insghts. I want to thank Mariem Boujelbene for the infinite support. I want to thank my parents and sisters for being there for me despite the distances and time difference.

This project was supported by NSF Grant IIS- 1549981

ABSTRACT

MODELING AND COUNTERACTING EXPOSURE BIAS
IN RECOMMENDER SYSTEMS

Sami Khenissi

April 25 2019

Recommender systems are becoming widely used in everyday life. They use machine learning algorithms which learn to predict our preferences and thus influence our choices among a staggering array of options online, such as movies, books, products, and even news articles. Thus what we discover and see online, and consequently our opinions and decisions, are becoming increasingly affected by automated predictions made by learning machines. Similarly, the predictive accuracy of these learning machines heavily depends on the feedback data, such as ratings and clicks, that we provide them. This mutual influence can lead to closed-loop interactions that may cause unknown biases which can be exacerbated after several iterations of machine learning predictions and user feedback. Such machine-caused biases risk leading to undesirable social effects such as polarization, unfairness, and filter bubbles.

In this research, we aim to study the bias inherent in widely used recommendation strategies such as matrix factorization and its impact on the diversity of the recommendations. We also aim to develop probabilistic models of the bias that is borne from the interaction between the user and the recommender system and to develop debiasing strategies for these systems.

We present a theoretical framework that can model the behavioral process of the user by considering item exposure before user interaction with the model. We also track diversity metrics to measure the bias that is generated in recommender systems, and thus study their effect throughout the iterations. Finally, we try to mitigate the recommendation system bias by engineering solutions for several state of the art recommender system models.

Our results show that recommender systems are biased and depend on the prior exposure of the user. We also show that the studied bias iteratively decreases diversity in the output recommendations. Our debiasing method demonstrates the need for alternative recommendation strategies that take into account the exposure process in order to reduce bias.

Our research findings show the importance of understanding the nature of and dealing with bias in machine learning models such as recommender systems that interact directly with humans, and are thus causing an increasing influence on human discovery and decision making.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Machine learning models make several assumptions in order to provide unbiased predictions. One of these assumptions is the fact that the data is collected randomly. Ideally, the collected data should follow a normal distribution. Usually in a data science project, human experts are responsible for collecting and labeling the data. For example, a classic problem of building a machine learning model that can classify images of dogs and cats need labeled data of images. Images of cats and dogs will be collected randomly by an expert and the data will be labeled so that we have an oracle for labeling training and testing data.

Recommender systems should be treated differently because of the way the data is collected. Indeed, the user is responsible for data collection and labeling. This happens because most recommender systems collect interest data such as views or clicks from users in order to make future predictions. The user then sees the recommendations and provides the next batch of ratings. This closed feedback loop generally narrows the available data to only the items that the user has been exposed to. As shown in Figure 2, the closed loop shows how the collected data is affecting the quality of the recommender system. To have an unbiased system, the data shown to the user should be randomly selected and then the user should rate all the seen items (to eliminate the user bias) which is far from a real-world application. This process causes several serious problems for both the users and the items. For instance, the user may experience a filter bubble. When the model fails to learn all of the users' diverse interests, it can keep providing the same types of recommendations again and again. This is closely related to the exploitation and exploration problem. This can also cause polarization when the recommendations are related to a sensitive subjects like political articles in a news recommendation system. Items in a recommender system may suffer from underexposure. The closed feedback loop can cause an unfair exposure between the different items (movies, books, articles...) which may result in a skewed rating distribution and thus a minority of popular items and a large amount of unpopular or underexposed items. This may affect the sales made by online merchants, especially because new companies tend to rely on recommender systems to promote their products.

In this thesis, we try to model the exposure of the user to recommend items. Then we develop a model that can counteract this exposure bias. We test our model using different quantitative measures.

The rest of this thesis is organized as follows: We start by giving an overview of the previous work and the background in chapter 2. Then we formulate our research questions and provide our methodology in chapter 3. Finally, we present our experimental results in chapter 4 and make our conclusions in chapter 5.

1 Objectives

The main contributions of this thesis can be summarized as follows:

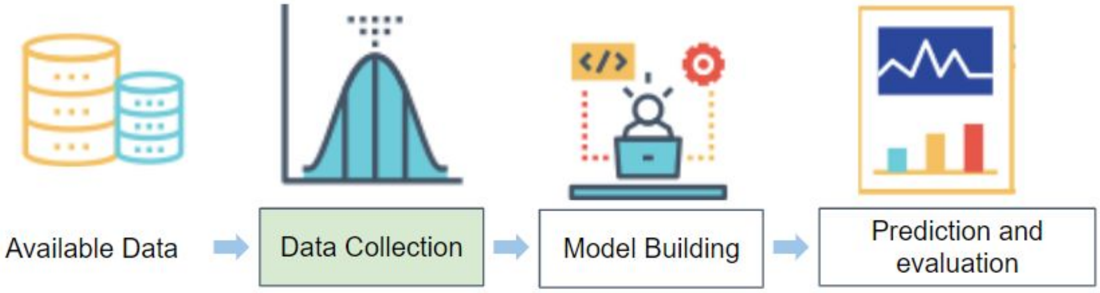


Figure 1. Classical machine learning pipeline

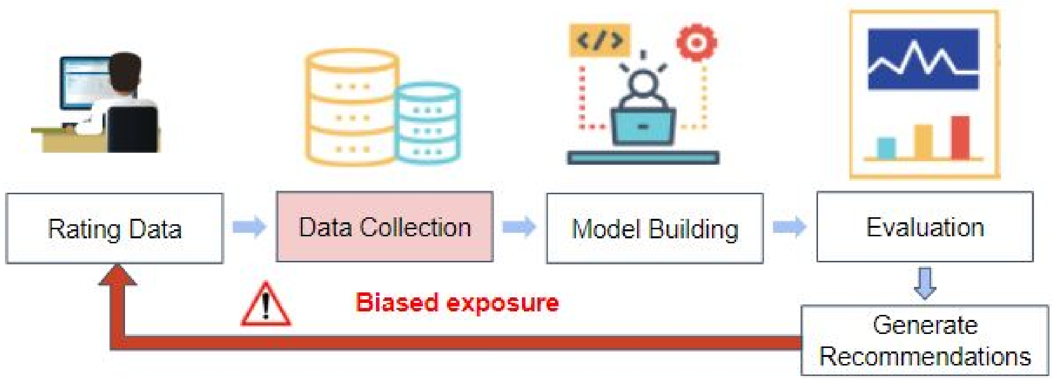


Figure 2. Recommender system pipeline

- Presenting an experimental framework to test various exposure models
- Developing a new debiasing strategy that can mitigate the effect of the exposure bias
- Developing an experimental protocol that shows how to simulate the feedback loop, track its effect using a real-life dataset and evaluate debiasing strategies.

CHAPTER II

LITERATURE REVIEW AND BACKGROUND

1 Recommender Systems

Definition and Background

Recommender Systems are intelligent systems that recommend items to a user that are predicted to be desirable by the user [15]. [51]. There are two types of recommender system models in the literature:

- Content Based Filtering [13, 5]
- Collaborative Filtering [28, 14]

In Content-Based Filtering methods, the model uses various features of the users and items such as geographical data, preferences, search history, movie genre... These features can be used to build a simple supervised classifier using Logistic regression or other known classification models to classify the item as ”should be recommended” or ”should not be recommended”.

The second type of Recommender System is Collaborative Filtering. In this type, the model uses only available ratings or interactions from the users to the existing items. Collaborative Filtering is the most common technique used in Recommender

Systems. This technique tries to make predictions about a user's likes/dislikes based on other groups of users with similar tastes to the target user. The origin of Collaborative Filtering can be tracked to the mid-1990s,[47] when researchers decided to look into the structure of ratings and provide an intelligent model to predict future ratings. The first Collaborative Filtering model was developed by David Goldberg [12]. It is the first commercial Recommender System that recommends documents from newspapers to users.

To build a good Collaborative Filtering Model, a basic assumption should be met. If user A and user B rate the same items and have similar behavior, they should continue this behavior across other unknown items [26].

In the next section, we will review some previous work done on Recommender Systems and more specifically Collaborative Filtering since it is the backbone of our project.

Techniques and algorithms

Introduction to Collaborative Filtering techniques

As stated before, Recommender Systems can be tracked to the 90s. After "Tapestry" [12], many researchers adopted the term Collaborative Filtering. Collaborative Filtering can be categorized into two main groups: Model-based and Memory based [3, 9].

In the memory-based approach, the idea is to transform the input ratings into a matrix and perform a neighborhood approach, either item-based or user-based. In the user-based approach, the system recommends items to the active user based on similar interests from his/her neighbors. A similarity function is defined so that we map each new user to similar previous ones and use their current rating to predict the new rating using an aggregation method [22]

For the item-based approach, as adopted by some companies like Amazon [34], the recommender system recommends items to the user based only on his/her previous ratings. The recommended items will be similar to the previous items rated by that user [43]. Like the previous method, a similarity function needs to be defined to express the neighborhood of the items. Finally, an aggregation of the total ratings is calculated and the top-n recommendations are returned to the user.

In the model-based approach, Recommendations are calculated based on a probabilistic approach. Possible models to estimate the probability are Bayesian networks or clustering techniques [57, 9]. Other models can be constructed to estimate the value of the recommendation like a deep learning approach (Restricted Boltzmann Machine [42], Autoencoders [7] or Recurrent neural networks [17]) or other factorization approaches like Matrix Factorization [29] and Bayesian Personalized ranking [41]

In this project, we will focus on Factorization based models like Matrix Factorization. The aim is to show the exposure bias problem in these models and provide a correction that can improve the model.

Matrix Factorization

Matrix Factorization (MF) is a model based Collaborative Filtering approach that returns accurate results. MF first was first used as a Collaborative Filtering technique in 2009 by Koren et al. [29] who won the Netflix $1M prize. They proved that Matrix Factorization was a more accurate model than previous neighborhood style approaches. MF is efficient when dealing with sparse data and maps both users and items to a latent space of dimensionality k. Assume that $R \in \mathbb{R}^{m \times n}$ represents user's ratings to items where m is the number of users and n is the number of items. The main goal of CF recommender systems is to predict the ratings for the unknown values in the matrix R. The objective of MF is to find two latent factors matrices $P \in R^{m \times k}$ and $Q \in R^{k \times n}$ where P and Q are user and item's latent factor matrices respectively such that $R \approx PQ^T$. The entry in the $u-th$ row and the $i-th$ column of R, that is, $r_{u,i}$, is the rating user u gives to item i. The $u-th$ row vector p_u of P and the $i-th$ column vector q_i of Q are user u and item is latent vectors, respectively.

The resulting dot product, $q_i^T \ p_u$, captures the interaction between user u and item i -the user's overall interest in the item's characteristics. This approximates user u s rating of item i, which is denoted by r_{ui}, leading to the estimate

$$r_{ui} = q_i^T p_u \tag{1}$$

To learn the latent factor matrices p_u and q_i, the system minimizes the regularized squared error on the set of known ratings:

$$\min_{P,Q} = \sum_{(u,i)\in R} (r_{ui} - q_i^T p_u)^2 + \beta(\|(q_i^2\| + \|(p_u^2\|) \tag{2}$$

Here, R is the set of the (u, i) pairs for which r_{ui} is known(the training set).

The latent factors P and Q are very important. They are used to representing the users in a common space with the item and then estimate the ratings based on this representation. (Fig. 3). Items i and users u that are close to each other will yield a high rating r_{ui} since they have a high cosine similarity and therefore a maximized value of the dot product $(p_u q_i)$.

To learn the values of this latent factor representation, an optimization techniques is such as gradient descent or stochastic gradient descent. In the stochastic gradient descent approach, as described by Koren [**?**] [31], we update the value of P and Q in a way that minimizes the value of the training error.

$$e_{ui} = r_{ui} - p_u.q_i^T \tag{3}$$

$$\begin{aligned} p_u &\leftarrow p_u + \alpha(e_{ui}p_u - \lambda q_i) \\ q_i &\leftarrow q_i + \alpha(e_{ui}q_i - \lambda p_u) \end{aligned} \tag{4}$$

α is the learning rate that defines the step size of the update. A very small α can cause the algorithm to be stuck in local optima, in contrast, a large value can cause the algorithm to miss the convergence area. A hyperparameter tuning should be performed to select the right α.

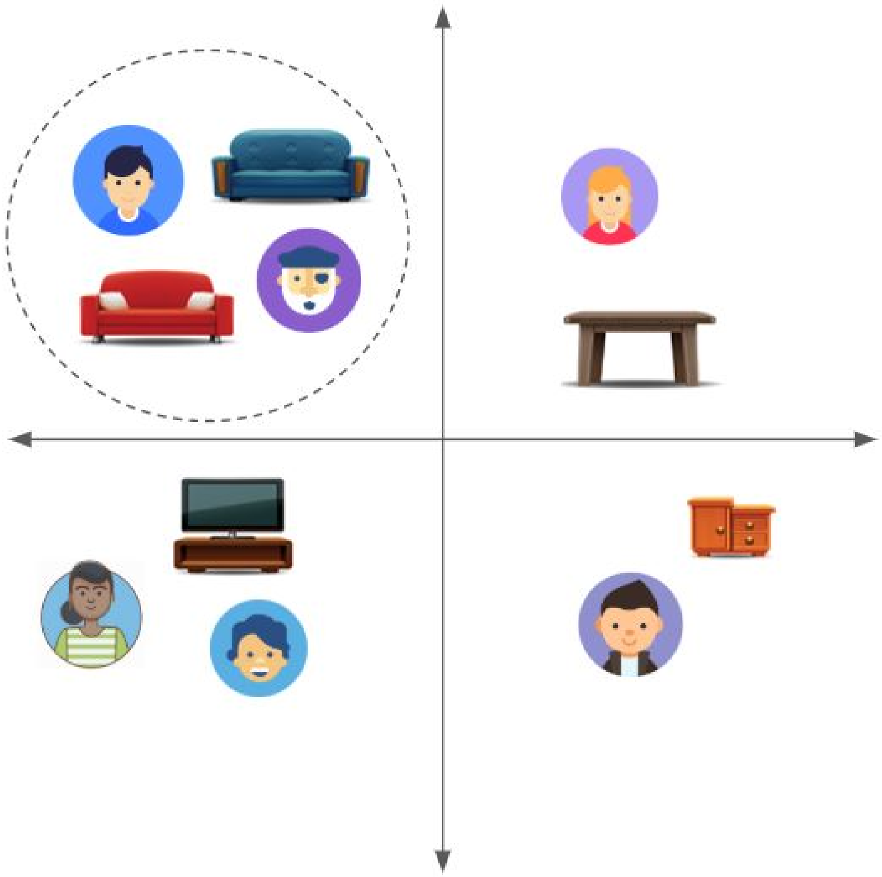


Figure 3. Representation of different users (person icons) and items (various objects) in the latent space. The dashed line encloses a neighborhood of related users and items that would yield high ratings

The problem with recommender systems models is that when they are treated as a classical machine learning algorithm, they fail to provide unbiased predictions. The main reason for this is the continuous interaction between the user and the algorithm. In the next section, we will explain how this feedback loop can affect the performance of the recommender system and affect the predictions.

2 Exposure Bias Effect

Background

When we collect the ratings in a Recommender System, we rely on the user interaction with the displayed items. Thus, the user can only provide ratings for the items that he/she has been exposed to. This leads to numerous problems such as an unbalanced rating distribution between the items. This unbalance causes the different estimators that we use to evaluate the performance or the error of the model to be biased [44]. Schnabel. et al proved that using the observational data from the available ratings will lead to a biased estimator when we use the regular Mean Absolute Error and Expected Risk Minimization to learn our models. The true evaluator for an algorithm performance would be:

$$MAE_{true} = \frac{1}{m * n} \sum_{(u,i) \in (U,I)} |R_{ui} - \hat{R}_{ui}| \tag{5}$$

Where U is the number of users, I is the number of items and $\hat{R}_{ui}$ is the predicted ratings Where MAE_{true} needs to be calculated using all the data not only the available one.

In fact, since we don't have the full data, we use estimators to estimate the true value of the performance of the recommender systems. The usual estimator in most of the Recommender System research is the naive Mean Absolute error (Equation 6)

$$MAE_{naive} = \frac{1}{|R_{ui} \neq 0|} \sum_{u,i \text{ for } R_{ui} \neq 0} |R_{ui} - \hat{R}_{ui}| \tag{6}$$

If we calculate the expected value of this estimator:

$$E(MAE_{naive}) = \sum_{u,i \text{ for } R_{ui} \neq 0} P(R_{ui} \neq 0) \times |R_{ui} - \hat{R}_{ui}| \tag{7}$$

Thus

$$E(MAE_{naive}) \neq MAE_{true} \tag{8}$$

This proves that the recommender systems are biased mathematically due to the biased estimator that we use in order to estimate the training error and testing error.

Some focused on the randomness of the missing data and considered this exposure problem as a Missing not at Random problem [37]. Researchers provided different solutions to account for this problem like new training methods [49] or a new objective function [56]. These methods have a strong mathematical justification but they lack intuition and they are specific to a given algorithm. Finally, these solutions did not provide a theoretical study that shows the effect of the feedback loop on the exposure bias.

Other approaches treated the exposure bias problem from the effect perspective. In fact, some researchers studied this problem while dealing with a specific problem of recommender systems like filter bubble [39], polarization [4] or underexposure [2].

The Filter Bubble Problem

The filter bubble is a term that was first defined by Eli Pariser [39] where he suggested that search engines and recommender systems will create some sort of bubble preventing the users from exploring diverse recommendations and new items. The filter bubble problem causes the recommendation list to be less diverse [38] and hence the user is stuck in a "bubble" where he/she keeps seeing the same type of recommendations. For instance, let us consider the behavior of a movie recommender system. Suppose that the user highly rated 10 action movies and zero comedy movies. The algorithm will then keep recommending the same type of movies repeatedly. Following this behavior, the recommended list will have low diversity.

Some have treated this problem as an exploration exploitation issue [35] [6], hence using methods like Multi armed bandits in order to increase the diversity of the recommendation list [40] , [36], [32]

The filter bubble is a serious problem that can limit the discovery of the user. For instance, for a news recommender system, this can affect the subjects that a user will read, and thus in turn can lead to another serious consequence such as polarization.

Polarization

Another important problem in Recommender systems is polarization [4]. It is closely related to the filter bubble problem [48]. Polarization can be seen generally in social networks where people interact with some published news that have a social or political dimension. This interaction can form two different groups of people with opposite opinions. If each group is trapped in his filter bubble then the recommendations will prevent them from discovering the other side of the subject and the separation will become worse. This is very important because these recommender systems may influence the results of elections and drive popular opinion. Studies [11] showed that the feedback loop in recommender systems may contribute to this polarization effect, hence, the need for effective debiasing strategies that can limit it.

Underexposure

In every recommender system, we have two main actors: Users and Items. The exposure bias is as harmful to the items as it is to the users. In fact, many items struggle to reach the user relying only on the recommender system and without using paid ads and promotions. This is due to the fact that most biased recommender systems keep promoting the highly popular items and neglect items present in the long tail [2]. This bias is called popularity bias [53]. This popularity bias is also closely related to the exposure bias. Since these popular items are being recommended more often, they get more reviews and ratings than other products (because of the different exposures). This leads to a type of biased distribution. Efforts have been done to study this effect and how state of the art recommender systems are contributing to this popularity bias problem [1]. The long tail items, however, constitute a major proportion of all the items and they should have a big share of the aggregated user interactions. Hence, dealing with the underexposure problem will be beneficial to the users since they will be able to explore newer and more diverse recommendations. It will also be beneficial to the recommender system platform since it will increase the user engagement to a much larger subset of items. This may generate additional revenue and collect more valuable information to increase the accuracy of the model. Finally, it may be beneficial to the items because they may gain a larger audience and more fair exposure.

Counteracting Exposure Bias

Many models have emerged trying to solve the different consequences of the exposure bias problem. Dealing with the different kinds of bias can occur in many steps during the recommender system's pipeline. Some methods incorporate bias correction during the training by using a regularization technique or switching the objective functions [44]. Others methods use post-learning techniques which consist of applying different ranking and selection strategies after performing the predictions. In this family, we find a big interest in the Multi-Armed Bandits techniques since they have proven to be efficient in the exploitation vs exploration problems.

Multi Armed Bandits and Ranking Techniques

As we stated, Multi-Armed Bandits techniques (MAB) are very popular approaches in the exploitation exploration problems [27]. They are used to enhance exploration. Many variations of approximations [30] are used with these techniques like and not limited to:

- $\epsilon - greedy$ approach [30]
- Thomas Sampling [30]
- Upper Confidence Bound. [30]

It is no a surprise that, since Multi-Armed Bandit techniques are good with exploration problems, they are often used in Recommender systems in order to solve the exposure bias problem. For instance, a group of DeepMind researchers [25] recently used MAB to study the effect of the feedback loop in recommender systems. They showed that the feedback loop can decrease the quality of the recommendations and they also showed that random exploration using Multi-Armed Bandit techniques can enhance and boost the quality of the predictions.

Other methods [2] use other ranking strategies like accounting for diversity while providing recommendations. In fact, Abdollahpouri et. al [2] rank the items based on two probabilities. The first one is the usual accuracy based probability that recommends the item based on its relevance. The second probability takes into account the diversity of the item. This helps in promoting long tail items. The method does not consider the feedback loop into its experiments.

Confounding techniques

Other techniques focused on eliminating the mathematical bias of the learning algorithm [44]. Schnabel et al [44] provided an inverse propensity strategy that eliminates the bias of the mathematical estimator for the training error by minimizing a modified loss function as follows

$$\min_{P,Q} = \sum_{(u,i) \in R} \frac{1}{E_{ui}} (r_{ui} - q_i^T p_u)^2 \tag{9}$$

Where E_{ui} is the probability that user u has seen the item i. The advantage of this method is that it takes into account the confounding factor resulting from the exposure bias. However, this method cannot be trained using regular Gradient Descent. Instead it should use multiplicative optimization updates to avoid divergence. Finally [44] do not consider the feedback loop effect when testing the effectiveness of the model instead, considering only considers the offline evaluation and ignoring the bias added by the " *iterated*" nature of the recommender system pipeline.

Regularization techniques

Other techniques used regularization to account for the bias in the data. Abdollahpouri et al [1] provided a regularization strategy that accounts for the popularity in a recommender system. The main idea of the model is to push recommendation in a way that balances the accuracy along with the intra-list diversity. This is achieved by using the dissimilarity matrix in the regularizer first introduced by Jacek Wasilewski and Neil Hurley [54] where the main idea is to minimize the quadratic form of the Laplacian matrix.

This method helps to control the popularity bias in regular Matrix Factorization. However, it ignores the feedback loop and the iterated effect. Furthermore, it only takes into account the diversity of the items based on their attributes. It does not consider the exposure of the user to each item.

Iterated Bias Frameworks

Some works focused on the iterated bias of Recommender Systems. Bountouridis et al. [8] designed a simulation framework to see the effect of the recommendation models on the diversity and novelty of the Recommendations. They used news data from BBC and they compared many state-of-the-art models. They did not provide a solution to mitigate this bias effect.

DeepMind also provided a simulation framework [25] where they showed that the iterative effect of the recommender systems decreases the utility of the recommendations. One limitation is that they used a simulated dataset.

Sun et al. [52] presented simulations to study the effect of the feedback loop from a machine learning perspective. They used synthetic data and hypothesis testing in order to study how the predictions shift when we model continuous interactions between the human and the model in a feedback loop. Sun et al. [55] later presented a study of several exposure bias counteraction strategies within an iterated framework.

One main challenge of studying the iterated bias of the recommender systems is availability of the right data [45]. Unless we make a user study, we cannot accurately estimate human behavior without loss of accuracy. This challenge has led scientists to come up with other methods like experimenting on synthetic data [25] and semi-synthetic data [45]. Still, this new field of research is not explored enough and needs a proper formulation in order to provide recognizable contributions.

3 Chapter Summary

In this chapter, we presented the background for our work. We first provided an introduction to Recommender Systems and presented popular methods such as Collaborative filtering and more specifically Matrix Factorization. We also presented the problem of Exposure Bias by first giving the mathematical intuition using a statistical justification, then we showed the different consequences that can result from exposure bias. Finally, we enumerated different strategies used to counteract the bias problem and discussin their limitations. In the next chapter, we will propose a new model to estimate the user exposure to all the items present in the Recommender system and then present a new strategy to counteract the exposure bias problem.

CHAPTER III

METHODOLOGY

In this chapter, we present our methodology for how to model and counteract the exposure bias. We focus on collaborative filtering strategies and specifically matrix factorization. Our method is a new regularization based technique that aims at smoothing and correcting the algorithm update rules in order to account for the exposure bias. We also present how we model the user exposure distribution since it is crucial for understanding this issue. We thus start by presenting how we define the user exposure distribution and how we plan to model it. Later, we present our regularization technique giving the mathematical intuition, the algorithm outline and a general framework.

1 Notation

First we define the notation we are going to use in the next sections.

- $U = \{\text{Set of all users}\}$
- $I = \{\text{Set of all Items}\}$
- E: Exposure matrix
- P: user latent feature representation

- Q: Item latent feature representation
- α: learning step
- β: Regularization hyperparameter
- λ: Popularity and Exposure Aware Regularization hyperparameter
- t: Learning iteration
- JSD: Jensen Shannon Divergence
- R_{ui}: Rating given by user u to item i. We set the initial $R_{ui} = 0$ if the rating is missing

2 User Exposure distribution

We define the user exposure distribution as the probability that the user u has seen item i. This distribution defines the likelihood that a given user has been exposed to different items. We define this distribution as a matrix E mapping the users to the items where:

$$E_{ui} = P(\text{user } u \text{ has seen item } i) \tag{10}$$

This distribution is also called propensity [44], and hard to estimate because it depends on many random factors such as the demographic features of the users: Users from different ages, locations etc are exposed to a different set of items. It also depends on the popularity of the item. Popular items are more likely to be seen than other items. It also depends on the websites, social networks and different platforms that the user may have used since they can promote different ads and products. An important factor that affects the user exposure distribution is the previous iteration of recommendations. In fact, the recommender system is exposing the user to a new set of items during each recommendations. Items with high likelihood of being recommended will higher chance of being seen.

The exposure distribution is thus affecting the data collection process. In fact, the user cannot rate an item if he/she has not seen it, hence the need to study this distribution and understand how we can use it in order to mitigate the resulting exposure bias.

Fair Exposure

The fair exposure distribution consists of the uniform distribution. In fact, a fair exposure across all items means that the user has equal chances of seeing all the items.

$$P(\text{user } u \text{ has seen item } i) = \frac{1}{|I|} \tag{11}$$

An example of a fair exposure is a recommender system that is starting with totally new items (the users have never seen any of the items in the recommender system). Then the recommender system will recommender a set of items randomly (based on the uniform distribution). This way, the user will have equal chances of seeing and rating these items. The resulted trained model will be unbiased since we used an unbiased data collection strategy in order to get the ratings. However, this scenario is unfeasible when dealing with a real world dataset. The main pupose of a recommender system is to provide relevant recommendations. Hence if we use random selection as a recommendation strategy, the predictions would be inaccurate. We want to develop a method that keeps providing accurate predictions and at the same time takes into account the different exposures for each user and item.

Popularity Based Exposure

Some previous work used item popularity [10] [50] as an estimator for the exposure distribution since the exposure of a user to an item. in our research, we use the popularity based exposure model as a baseline and try it with different alternatives in order to see whether we can improve this estimation.

The popularity based exposure model is defined by the following equation:

$$E_{ui} = \frac{|\{R_{ui} \neq 0\}|}{|U|} \tag{12}$$

It is important to note that the popularity based exposure model depends only on the item and not the user. It cannot provide a personalized estimation of the exposure for each user. Also popularity here captures that an item is rated by many users and not necessarily that is liked.

Poisson Factorization based model

Although a popularity based exposure model is good with approximating the user exposure, it lacks at providing personalized approximations for each user. Other work [44] suggested using learned models like Naive Bayes or logistic regression. In our research, we are going to test another probabilistic framework that is more suitable for the recommender system setting, namely Poisson Matrix Factorization [19],[21], [20].

Poisson Matrix Factorization has been used before for approximating the user exposure [33] and we will use it in this project to show how it is performing in an iterated framework. It is given by

$$E_{ui} \sim \text{Poisson}(\theta_u^T \phi_i) \tag{13}$$

Where θ_u and ϕ_i are representations of users and items in a latent space that follow the following property:

$$\theta_u, \phi_u \sim \text{Gamma} \tag{14}$$

We are going to compare the performance of the Poisson matrix factorization and the popularity exposure model using a real life dataset and taking into account the temporal relation in the ratings in order to evaluate which one results in a better performance.

Furthermore, we will use the exposure model to design a new regularization function that can help mitigate the exposure bias problem.

3 Popularity and Exposure Aware Regularization for Matrix Factorization (PEAR-MF)

We design a new regularization function based on the exposure model in order to make matrix factorization aware of the exposure bias. The main idea is to modify the objective function of the Matrix Factorization problem in order to make the update rule of gradient descent different depending on the existing bias.

The new objective function for PEAR-MF is:

$$J(P,Q) = \sum_{u \in U, i \in I} (R_{ui} - P_u Q_i^T)^2 + \beta(||P_u||^2 + ||Q_i||^2) + \lambda JSD_u(E|| \frac{1}{|I|})(||P_u||^2 + ||Q_i||^2) \tag{15}$$

JSD_u is the Jensen Shannon Divergence [16]. It is a statistical distance that measures the similarity between two distributions D_1 and D_2.

$$JSD_u(D_1, D_2) = \frac{1}{2} D_{KL}(D_1||M) + \frac{1}{2} D_{KL}(D_2||M) \tag{16}$$

Where

$$M = \frac{1}{2} D_1 + \frac{1}{2} D_2 \tag{17}$$

and D_{KL} is the Kullback–Leibler divergence given by

$$D_{KL}(D_1||D_2) = -\sum_{x \in \chi} E(x) \log(\frac{D_1(x)}{D_2(x)}) \tag{18}$$

Intuition

In our regularization function, we calculate the JSD between the exposure distribution E and the uniform distribution for each user. If the JSD is equal to zero then this means that the user had a fair exposure to all the items, which means that the use of the regular matrix factorization will not affect the predictions.

If the JSD is equal to one, then the estimated exposure distribution is maximally dissimilar from the uniform distribution. This means that the user has experienced under-exposure or over-exposure to certain items. In this case, the regularization term will contribute to penalizing the error function so that the algorithm will adjust its parameters to fit the new information.

Mathematical Analysis

In this section, we will study the convergence of our new objective function. First let us define the gradient descent updating rule for minimizing our objective function in each iteration (t).

$$P^{t+1} = P^t - 2\alpha(R - P^t Q^T)Q - 2\beta P^t - 2\lambda JSD_u(E_u || \frac{1}{|I|})P^t \tag{19}$$

$$Q^{t+1} = Q^t - 2\alpha(R - PQ^{tT})P - 2\beta Q^t - 2\lambda JSD_u(E_u || \frac{1}{|I|})Q^t \tag{20}$$

The algorithm uses Alternative Least square optimization method. It alternatively updates P and Q until we reach a fixed number of iterations or convergence.

Algorithm 1: Gradient Descent Optimization for PEAR-MF

Data: Rating Matrix R, Exposure distribution E, α, β, λ
Result: P and Q
while $t <$ *Maximum iterations* **do**

$$P^{t+1} = P^t - 2\alpha(R - P^t Q^T)Q - 2\beta P^t - 2\lambda JSD_u(E_u || \frac{1}{|I|})P^t$$

$$Q^{t+1} = Q^t - 2\alpha(R - PQ^{tT})P - 2\beta Q^t - 2\lambda JSD_u(E_u || \frac{1}{|I|})Q^t$$

end

Theorem 1. *Let J be the objective function defined by:*

$$J(P,Q) = \sum_{u,i}(R_{ui} - P_u Q_i^T)^2 + \beta(||P||^2 + ||Q||^2) + \lambda JSD_u(E||\frac{1}{|I|})(||P||^2 + ||Q||^2)$$

After fixing Q and at iteration $t+1$ during the gradient descent update, there exists $L \neq 0$ a real number where $J_Q(P^{t+1})$ is bounded by the following upper bound:

$$J_Q(P^{t+1}) \leq J_Q(P^t) - \frac{1}{2L}\left\|\nabla J_Q(P^t)\right\|^2 \tag{21}$$

for a learning step $\alpha = \frac{1}{L}$ and $\nabla J_Q(P^t) = \frac{\partial J(P,Q))}{\partial P}$

Proof. To prove the upper bound we start by showing that our objective function is lipchitz continuous. This means that for u, v as real numbers:

$$\|\nabla J_Q(u) - \nabla J_Q(v)\| \leq L\,\|u - v\| \tag{22}$$

We have:

$$\|\nabla J_Q(u) - \nabla J_Q(v)\| = \left\|2(R - uQ^T)Q - 2(R - vQ^T)Q + 2\beta(u-v) + 2\lambda JSD_u(E||\frac{1}{|I|})(u-v)\right\|$$

$$\|\nabla J_Q(u) - \nabla J_Q(v)\| = \left\|2(R - uQ^T)Q - 2(R - vQ^T)Q + (2\beta + 2\lambda JSD_u(E||\frac{1}{|I|}))(u-v)\right\|$$

$$\|\nabla J_Q(u) - \nabla J_Q(v)\| = \left\|(2\beta + 2\lambda JSD_u(E||\frac{1}{|I|}) - 2QQ^T)(u-v)\right\|$$

Then for $L = |(2\beta + 2\lambda JSD_u(E||\frac{1}{|I|}) - 2QQ^T)|$ we have from the Lipchitz continuous property and the convexity of J_Q:

$$\|\nabla J_Q(u) - \nabla J_Q(v)\| \leq L\,\|u-v\| \tag{23}$$

From the Lipchitz continuous property we can derive the following relation:

$$v^T \nabla^2 J_Q(u) v \leq ||v||^2 \tag{24}$$

After proving that J is Lipchitz continuous, we can write the Taylor expansion of J:

$$J_Q(u) \approx J_Q(v) + \nabla J_Q(v)^T(u-v) + (u-v)^T \frac{\nabla^2 J_Q(v)}{2!}(u-v) \tag{25}$$

Using the inequality in equation (23) we have:

$$J_Q(u) \leq J_Q(v) + \nabla J_Q(v)^T(u-v) + \frac{L}{2}\|u-v\|^2 \tag{26}$$

If we change u and v with P^t and P^{t+1} we get:

$$J_Q(P^{t+1}) \leq J_Q(P^t) + \nabla J_Q(P^t)^T(P^{t+1} - P^t) + \frac{L}{2}\left\|P^{t+1} - P^t\right\|^2 \tag{27}$$

Using the gradient descent rule we get the following relation:

$$P^{t+1} - P^t = -\alpha \triangledown J_Q(P^t) \tag{28}$$

if $\alpha = \frac{1}{L}$ then we get

$$J_Q(P^{t+1}) \leq J_Q(P^t) - \frac{1}{2L} \left\| \triangledown J_Q(P^t) \right\|^2$$

By symmetry of the function $J(P, Q)$, the inequality holds for $J_P(Q)$ $\square$

This theorem shows the convergence of the objective function to a local minimum since the problem is not convex (it is a bi-convex function).

This regularization function can be applied to other algorithms that are used to predict the ratings of the user and trained using gradient descent. Its main idea is to include the information of the over-exposure or under-exposure problem of a given user by increasing the error of the optimization process.

4 Chapter summary

In this chapter we presented a methodology that we are going to use to solve the exposure bias problem. First we explored the user exposure distribution. We defined its best case scenario and presented different models that we will use in order to approximate this distribution.

Then we presented our new version of Matrix Factorization called Popularity and Exposure Aware Regularization for Matrix Factorization (PEAR-MF) where we added a regularization term to the old objective function in order to penalize the error of Matrix Factorization whenever we detect an exposure bias problem using the Jensen Shannon Divergence function.

CHAPTER IV

EXPERIMENTAL RESULTS

In this section, we will present the experimental protocol that we used in order to validate our hypothesis. Our experiments focus on two main axes: Modeling the exposure bias and Counteracting the Exposure bias. In the first part, we will present an experimental approach that takes into account the temporal relation in the ratings in order to approximate the exposure of the user. We will treat the problem as a time series problem in order to evaluate the different models.

In the second part, we test the effectiveness of our counteracting strategy. We will give an experimental protocol, present our assumptions and compare our model to different state of the art models.

First, we start by presenting the data that we used for evaluation.

1 Data set

For the experimental results, we used the Movie-lens dataset [23] with 100K ratings. The data set contains 1000 users and about 1700 items. Only 5% of the ratings are available and most of the ratings are missing. The ratings in the dataset range from 1 to 5. 0 is given for a missing rating.

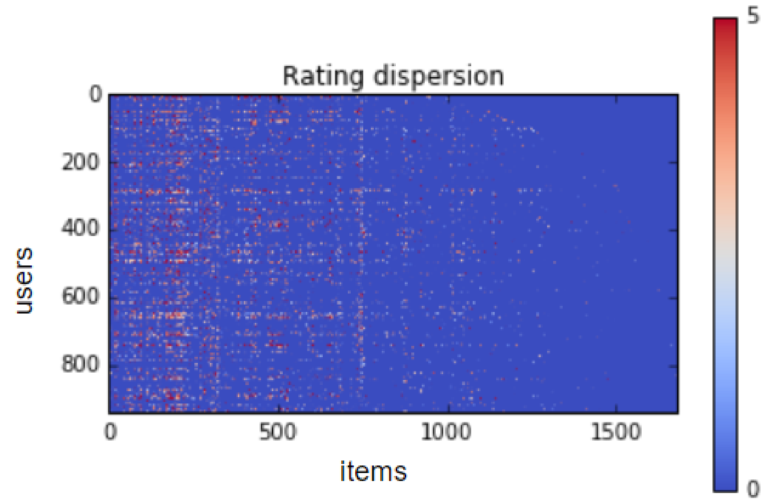


Figure 4. The dispersion of the ratings in the Movie-Lens 100K matrix, ze can see that most of the data is missing

The reason behind choosing the Movie-Lens dataset is to be able to compare it with previous results in the literature. This dataset has bees used by a large community of researchers since 1998 and it contributed to the development of the research in Recommender Systems.

2 Data pre-processing and exploratory analysis

Figure 4 shows the dispersion of the ratings in the Movie lens dataset. We can clearly see the sparsity of the matrix. Most of the ratings are missing and the existing ratings are concentrated in the first rows. We can assume that the first columns consist of popular movies.

Figure 5 and 6 confirm that most of the items are unpopular. In fact, only a small proportion of items have a high number of ratings. The popularity level is calculated

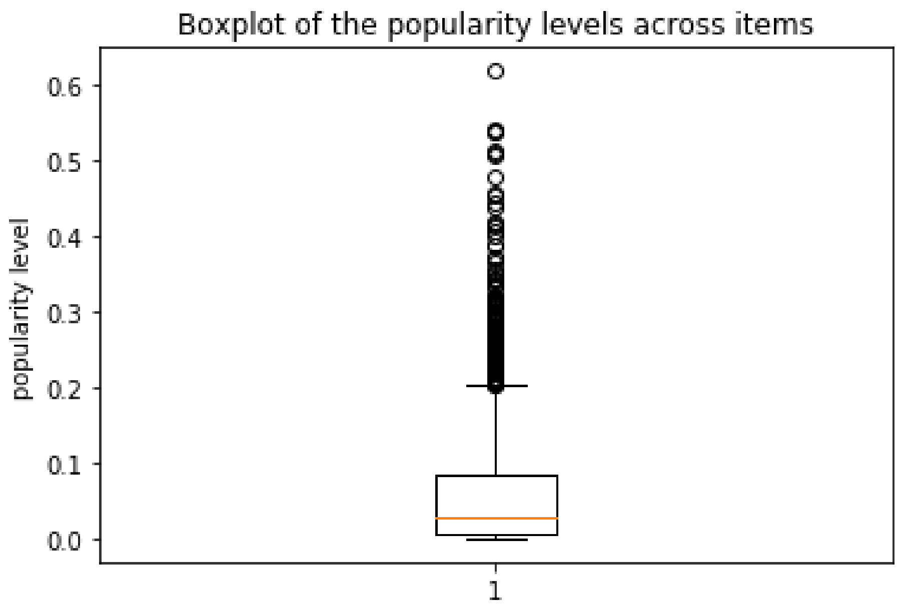


Figure 5. Box-plot of the popularity level in the MovieLens data

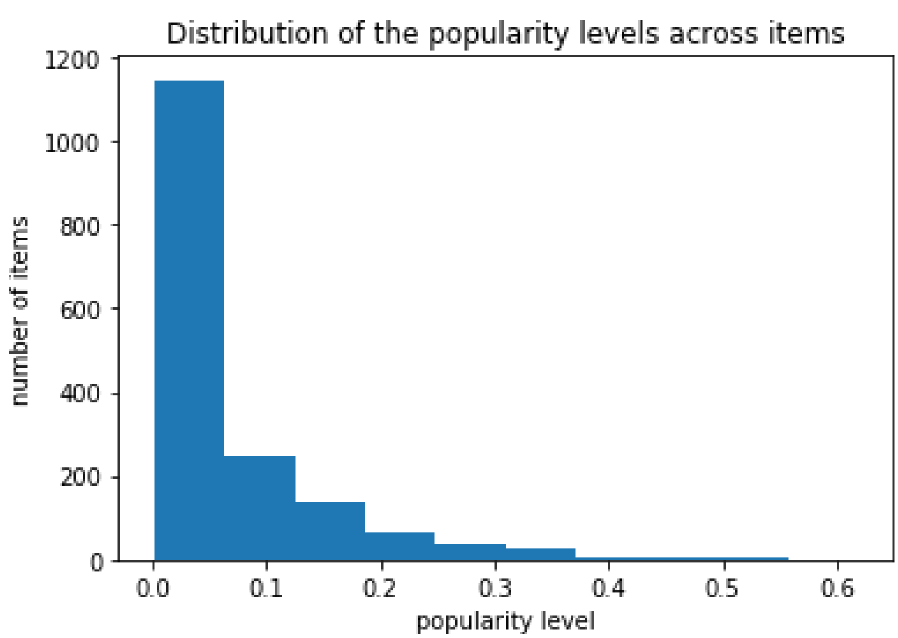


Figure 6. The distribution of the popularity in the MovieLens data

using the following equation:

$$E_{ui} = \frac{|\{R_{ui} \neq 0\}|}{|U|} \tag{29}$$

Which is the same equation used to calculate the popularity based exposure model.

3 Modeling the exposure bias

The first step of our experiments should be getting a good approximation of our exposure model. We designed an experimental protocol that relies on the temporal evolution of the ratings. In fact, the exposure of a given user relies heavily on the previous recommendations because of the closed feedback loop. For this reason, we first start by splitting the data into different parts, respecting the timestamps of the ratings.

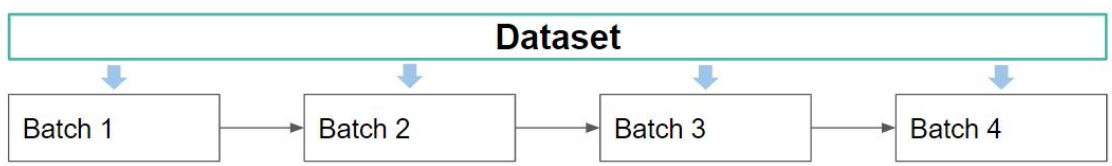


Figure 7. Data splitting example: Batch t designs the batch of the data at T=t

Figure 7 shows an example of our data splitting strategy. Contrary to the classical machine learning pipeline where we go for a random train testing split, we split the data by keeping the original order of the ratings. In Figure 7 we split the data into four splits or iterations where the first iteration contains ratings that happened before the second iteration.

Thus the evaluation process is going to be incremental like evaluating a time series prediction model. For each step, we train on one part of the data and we test on the next iteration. then we add this iteration's ratings to the training set and we repeat the process by testing on the next iteration, and so on.

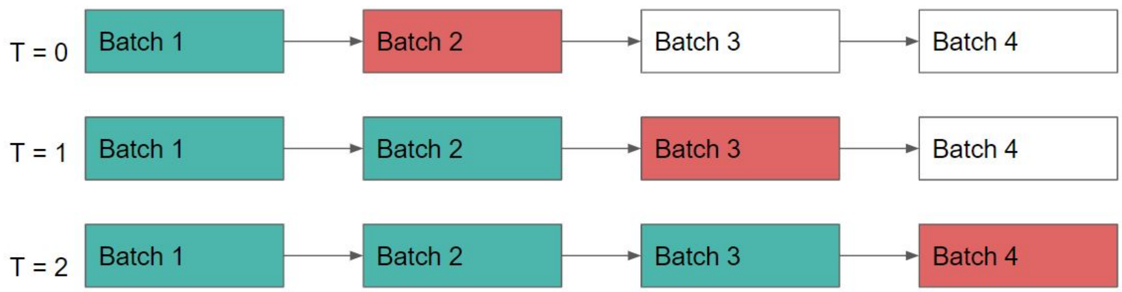


Figure 8. Training and Testing process for modeling the exposure bias. Training data is hown in green and the testing is shown in red

Figure 8 shows an illustration of this evaluation process.

- At $T = 0$ we start by training the model on the first iteration (batch) and test on the batch of iteration 2
- At $T = 1$ we merge batch 1 and 2 and retrain the model and test on batch 3
- At $T = 2$ we train the model on batch 1, batch 2 and batch 3 and we test it on batch 4

The goal of this process is to capture the temporal dependency of the rating data and see how the exposure is changing from one iteration to the next.

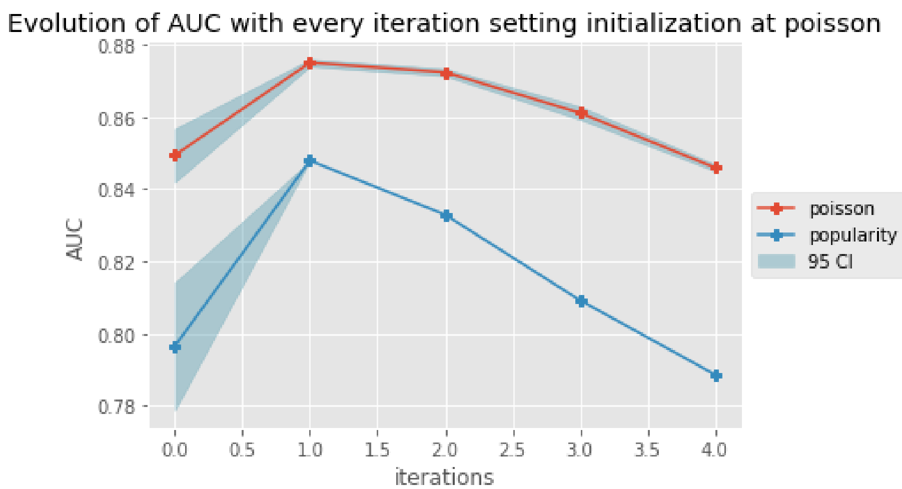


Figure 9. Evaluating Poisson based exposure model vs popularity based exposure model

Evaluation

To evaluate the performance of the models we treat the exposure estimation problem as a classification problem. For each user, we are trying to predict whether the user has seen the item or not. After predicting the scores or the probabilities of each item, we calculate the Area Under the ROC curve (AUC). In fact, the AUC is the area residing under the ROC curve that is defined by the True Positive Rate (TPR) and the False Positive Rate(FPR):

$$TPR = \frac{TP}{TP + FN} \tag{30}$$

$$FPR = \frac{FP}{FP + TN} \tag{31}$$

A perfect classifier would yield a score of 1 for AUC. It means that it correctly predicted all the true positives and has zero false positives.

Figure 9 shows that the Poisson factorization based exposure model significantly outperforms the popularity based model. This is in line with our expectation because Poisson factorization learns a more personalized model for the user exposure compared to the popularity-based model. The popularity model differs only from item to item but it does not differ from user to user. However, for instance, a user that is a sports fan will be exposed to more sport related articles than other users. For these reasons, the Poisson factorization can approximate the user exposure better than popularity model.

For the rest of our experiments, we will consider Poisson factorization as our exposure model and we will use it in our PEAR-MF and for designing our experiments.

4 PEAR-MF: Accuracy Performance and Hyperparameter Tuning

In this section, we will be interested in tuning and testing the performance of our new regularization strategy. We will use the same dataset (Movielens) and we will vary different parameters in order to see how it is affecting the performance.

The parameters that we are going to change are:

- α: The learning rate (Equation (15))
- K: number of latent factors in Matrix Factorization
- λ: Exposure coefficient (Equation (15))

To test the offline accuracy of the model we will follow a 5-fold cross-validation and compute three different accuracy metrics: Mean Absolute Error (MAE), Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG).

Mean Absolute Error

To test the accuracy of the model for each test user $u \in U_{test}$, we use MAE (Mean Absolute Error [46]) as a metric :

$$MAE_u = \frac{1}{|I^u_{test}|} \sum_{i \in I^u_{test}} |r^*_{ui} - r_{ui}| \tag{32}$$

Mean Average Precision

MAP is a known metric for evaluating recommender systems [24]. It is mainly used for document retrieval. But since the recommendation task is basically a ranking task, MAP is a valid evaluation metric for RS. MAP takes into account the ranking of the recommendation and gives us a good intuition of the precision of the top-n recommendation for test users. First let us define the general formula of Precision in Recommender Systems.

$$P = \frac{\text{Number of Recommendations that are relevant}}{\text{Total Number of Recommendations}} \tag{33}$$

From this expression we calculate the expression of the Average Precision for a set of items I:

$$AP = \frac{1}{|I_{relevant}|} \sum_{i \in I} P(i) rel(i) \tag{34}$$

when $!I_{relevant}$ is the set of relevant items $P(i)$ is the precision of the recommended item i as described in equation (33). $rel(i)$ is a binary indicator that takes the value of 1 if the item is relevant and 0 if not.

The Mean Average Precision is finally calculated based on the Average Precision for each user u in the data:

$$MAP = \frac{1}{|U|} \sum_{u \in U} AP(u) \tag{35}$$

Normalized Discounted Cumulative Gain

The nDCG metric is similar to MAP since it takes into account the ranking in the recommended list. If the high relevance items are placed at the top of the list, then we get a high nDCG.

Considereing R as the recommended list and $rel_i \in \{0, 1\}$, the Discounted Cumulative Gain is defined by

$$DCG = \sum_{i \in R} \frac{2^{rel_i} - 1}{log_2(i + 1)} \tag{36}$$

and

$$nDCG = \frac{DCG}{IDCG} \tag{37}$$

Where IDCG is the ideal DCG defined by

$$IDCG = \sum_{i}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i + 1)} \tag{38}$$

in IDCG, we rank the items by their true ranking according to their true ratings.

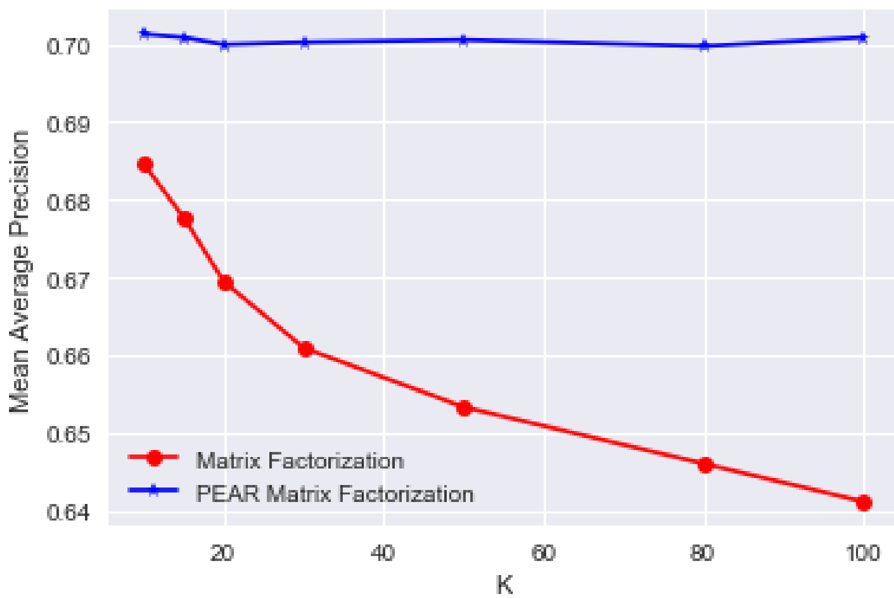


Figure 10. The variation of MAP when varying K

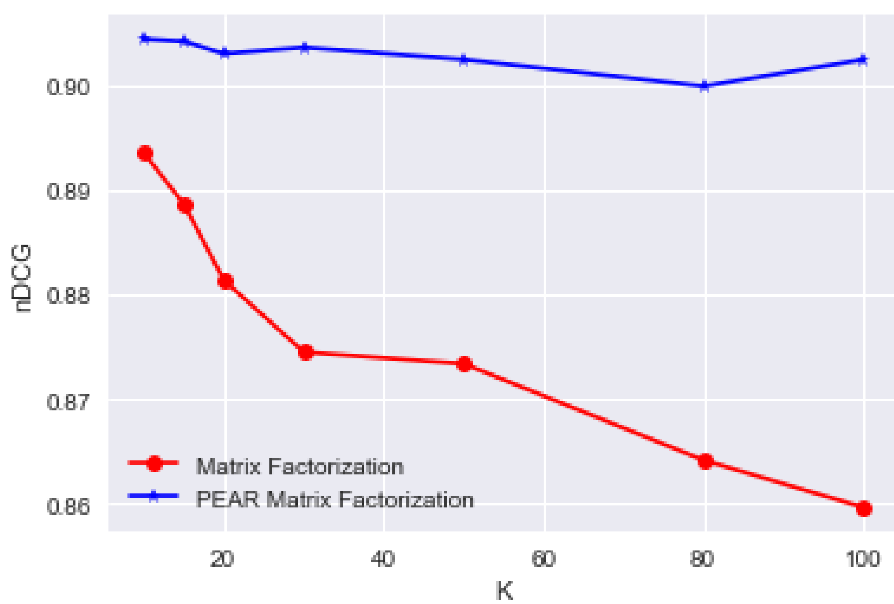


Figure 11. The variation of nDCG when varying K

Tuning the dimension of the latent space

In the first part of our tuning process, we vary the latent space dimensionality (K) and we evaluate the behavior of Matrix Factorization and PEAR-MF

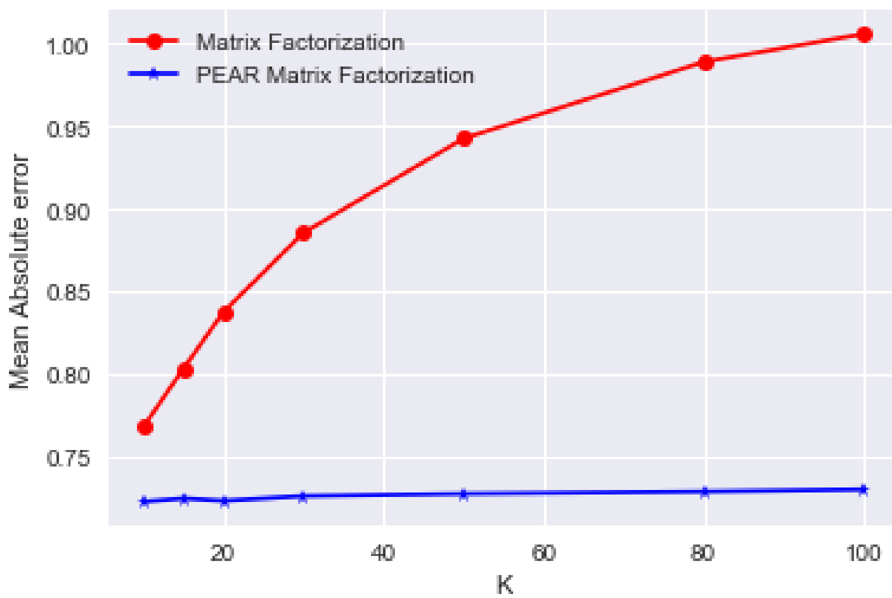


Figure 12. The variation of MAE when varying K

Figures 10, 11 and 12 show that PEAR-MF outperforms the regular MF in terms of accuracy as it is resistant to overfitting. The key reason for this is the extra regularization part added in the objective function. It helps MF generalize more on unseen data which is an advantage for trying to fix the exposure bias. Without the exposure related term in equation (15), the regularization treats all users and items when trying to reduce overfitting. With the exposure term, the amount of regularization is modulated in proportion to the extremeness of the exposure bias. This shows that the exposure bias problem and accuracy are related.

Tuning the Learning Step

The learning step is a very important hyper-parameter as it controls the convergence of the algorithm. A higher learning step can improve the convergence rate or it can make the algorithm diverge. Also, small learning steps can make the convergence

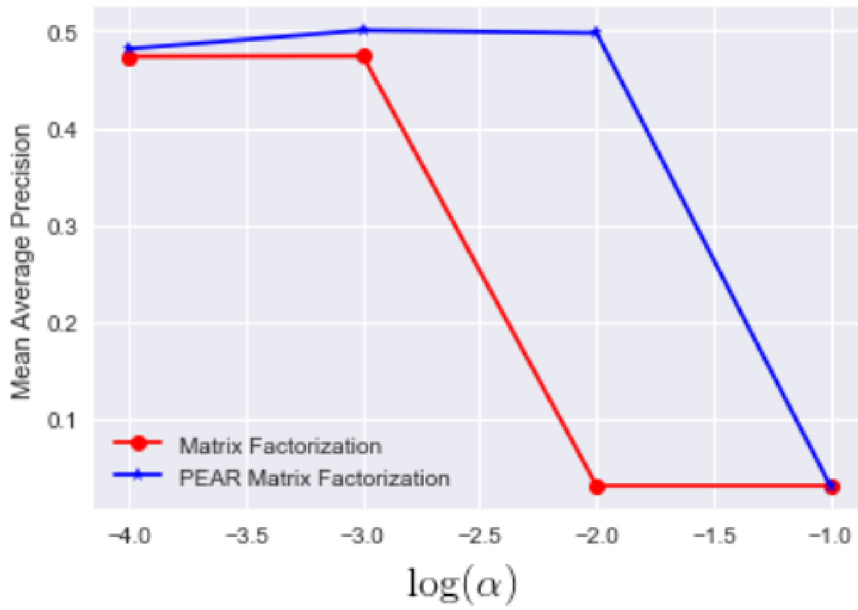


Figure 13. The variation of MAP when varying α

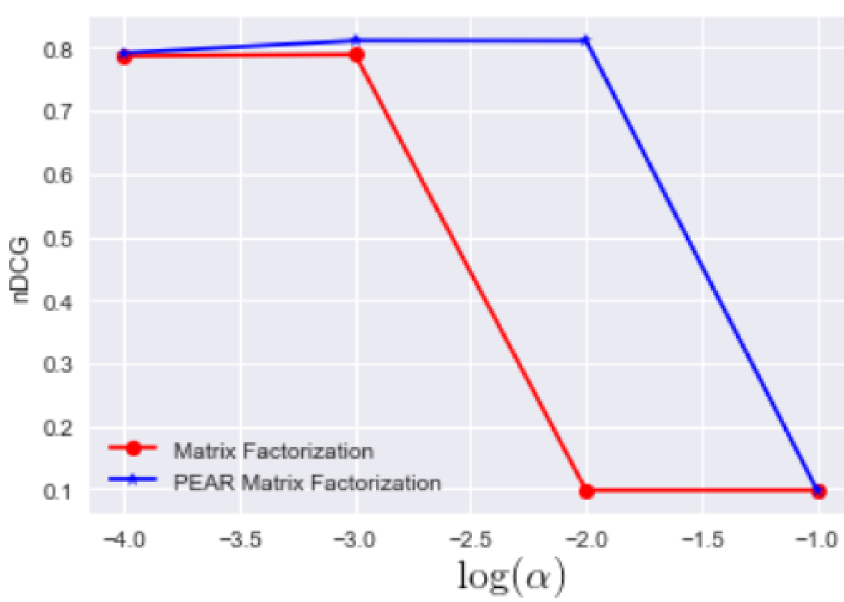


Figure 14. The variation of nDCG when varying α

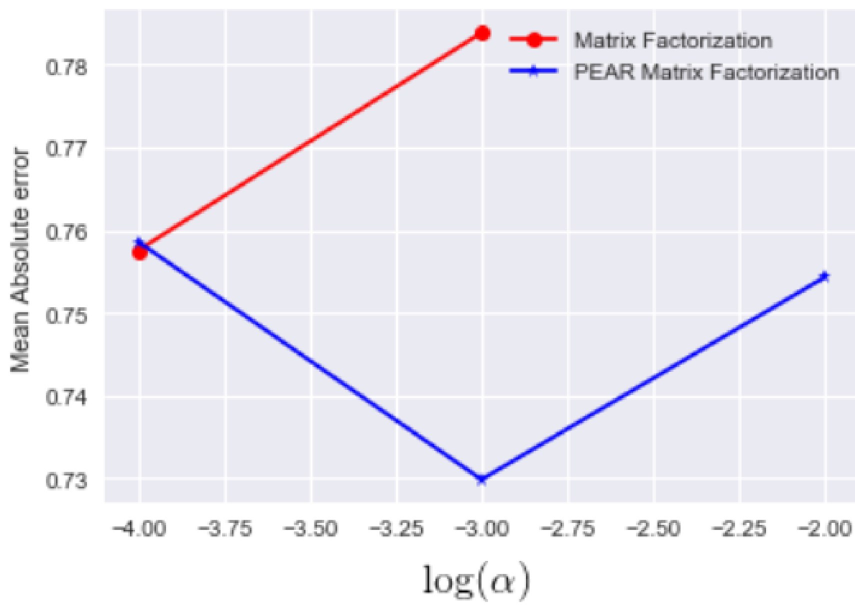


Figure 15. The variation of MAE when varying α

very slow and the algorithm can be stuck in a local minimum.

Figures 13, 14 and 15 show that PEAR-MF and MF have comparable performance for small α. Then when we increase α, the MF model diverges faster. Based on the above experiments, we will use $\alpha = 0.001$ in the remaining experiments.

Tuning the Exposure Coefficient

Another parameter that we will tune is λ which controls the Jensen Shannon regularization. $\lambda = 0$ means that we are using regular Matrix Factorization.

Figures 16, 17 and 18 show that $\lambda = 1$ gives the best results in terms of accuracy. This shows that a higher coefficient, to account for the exposure bias, improves the accuracy of the predictions. The results are also coherent for both prediction error (MAE) and ranking (nDCG and MAP).

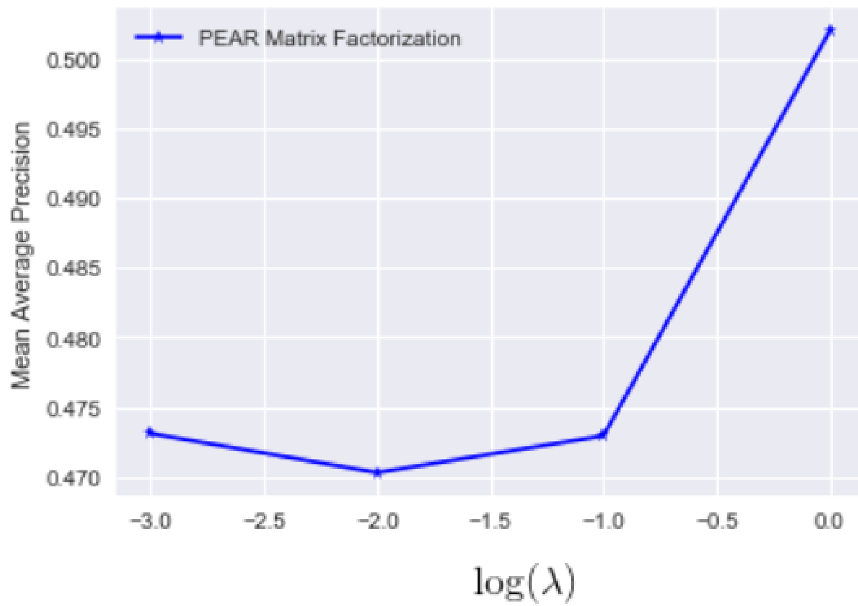


Figure 16. The variation of MAP when varying λ

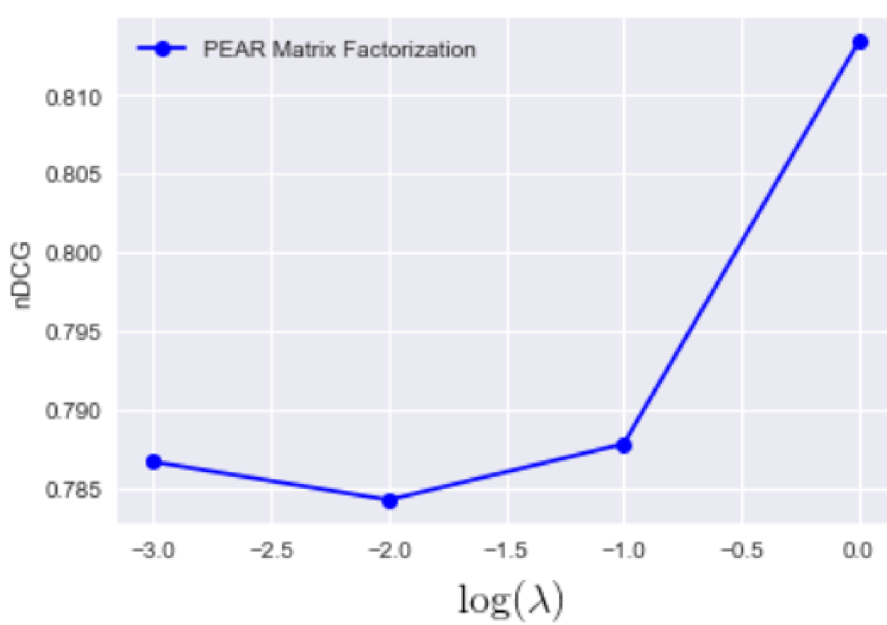


Figure 17. The variation of nDCG when varying λ

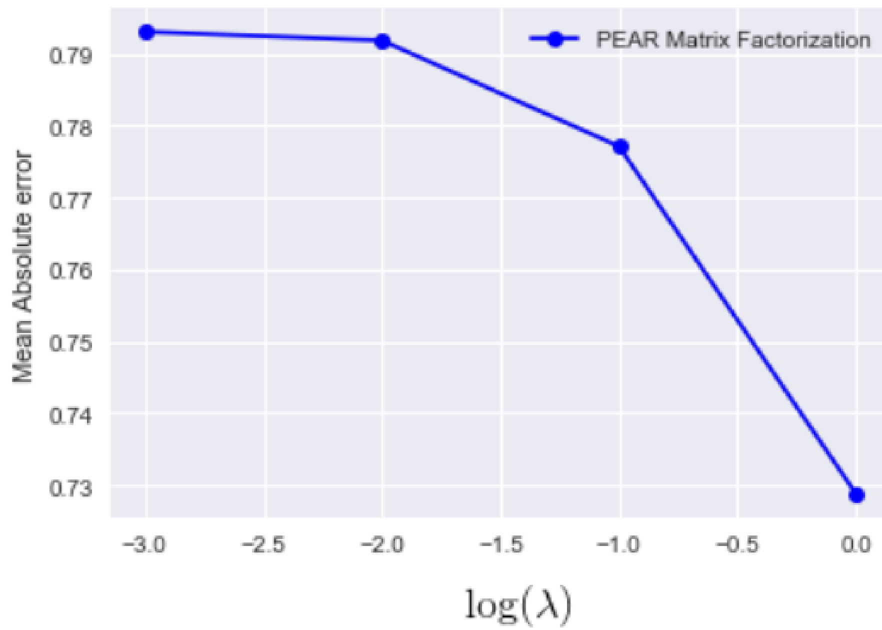


Figure 18. The variation of MAE when varying λ

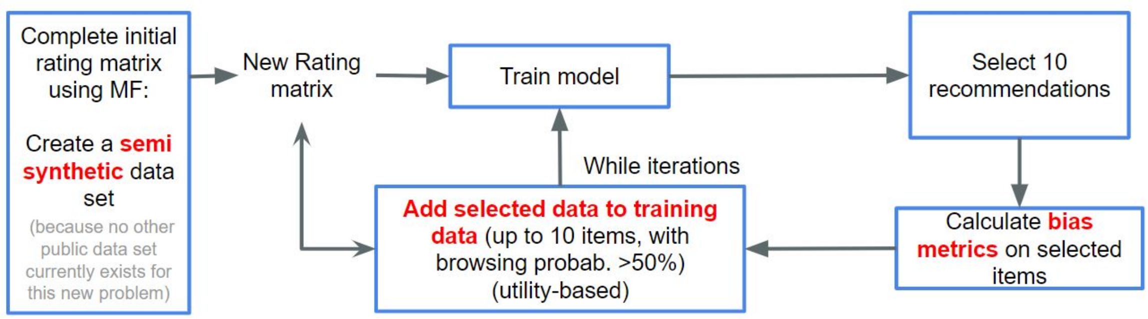


Figure 19. Experimental protocol for evaluating the exposure bias

5 PEAR-MF: Evaluation of Counteracting exposure bias

In this section, we evaluate the performance of PEAR-MF on how well it is counteracting the exposure bias. First we will define our experimental protocol, then we will define all the evaluation metrics that we are going to use and finally, we will present all of the experimental results along with the interpretation.

Figure 19 shows the methodology that we follow in order to evaluate the performance of our algorithm. First the pipeline shows the feedback loop simulation in our experiment. This is a very crucial part because we do not want to only evaluate the offline performance of our algorithm, but also to evaluate its behavior in an iterative framework which is closer to the real world performance.

However, this approach comes with limitations in terms of the required dataset. In order to get an accurate performance of the model in an iterated way, we need to know the complete preferences of the user including the missing ratings. This is not feasible in the real world because we cannot fill all the missing ratings in a recommender system (A user cannot rate all the items on Amazon for example). For this reason, we adopt a simple trick consisting of generating a semi-synthetic dataset as was done in Schnabel et al. [44]. The idea is to start from an incomplete dataset and use a Matrix completion algorithm to complete it (such as Matrix Factorization). Then we tweak the ratings in a way to make it match the real world ratings. For instance, for each user we cluster the predicted ratings into five percentiles $p1$,$p2$,$p3$,$p4$,$p5$. The highest percentile will then be assigned the rating of five, the next one will be assigned the rating of four, etc. This method helps eliminate user bias and creating a complete matrix that can be used in our online experiment.

After creating the complete matrix, we proceed to simulating the feedback loop of our recommender system. We train our model using the initial set of training data, then we select ten recommendations to present to the user. At this step, we will use these recommendations for calculating the performance metrics. Then we will select a subset from these recommendations in order to add it to the training set. This subset is selected based on the relevance of each item from the complete matrix. This relevance is called the browsing probability and was introduced in [10].

$$p(choose|u,i,R) \sim p(seen|u,i,R)p(rel|u,i,R) \tag{39}$$

After selecting the items that we are going to add to the training data, we use the complete matrix in order to add these ratings. We repeat this process for a fixed number of iterations. This protocol summarized in Algorithm 2, will enable us to evaluate the performance of the recommender system while taking into account the feedback loop.

Algorithm 2: Iterated framework for evaluating Recommender systems

Data: Recommender System Model M
Result: Evaluation Metrics E
Create a semi synthetic dataset ;
while $t <$ *Maximum iterations* **do**
    Train the Model M;
    Select Recommendations;
    Calculate Evaluation Metric E;
    Add Relevant Recommendations to the training set ;
end

Evaluating Exposure Bias

In this section we use two different datasets: Movielens 100K and Movielens 1M. The reason for working with a larger dataset is that we want to see the effect of scaling the algorithm to a larger and more sparse dataset.

To evaluate the exposure bias of the recommender system we will use the following metrics.

Expected Novelty

Expected Novelty or the Expected Popularity Complement (EPC), as defined by Castells et al [10], is a measure that evaluates the expected number of relevant items not previously seen by the user.

$$EPC = C \sum_{i_k \in Rc} disc(k) p(rel|i_k, u)(1 - p(seen|i_k)) \tag{40}$$

Where the notation is as follows

- C: Normalizing factor
- Rc: Recommended list
- disc(k): The discovery model that depends on the rank k of the items (items with higher ranks will have more discovery potential)
- $p(rel|i_k, u)$: Relevance model
- $p(seen|i_k)$: Exposure model

From the equation, we can see that the metric takes into account the rank, relevance and the exposure of the user to the item into consideration. It is called Expected Popularity Complement because the author [10] used the popularity based model as the exposure model but since we have shon that the Poisson model can provide more accurate estimation we will use the Poisson model as our exposure model. High EPC means that the Recommender System has recommended relevant items that the user has not seen before

Expected Diversity

The expected diversity is defined by Castells et al. [10] as the Expected Profile Distance (EPD). This metric measures the amount of diversity in each recommendation list based on the pairwise distance between the items in the recommendation and the items that the user has already interacted with.

$$EPD = C' \sum_{i_k \in Rc, j \in I_u \theta} disc(k) p(rel|i_k, u) p(rel|j, u) d(i_k, j) \tag{41}$$

- C': Normalizing factor
- θ : The set of items that the user has interacted with
- $d(i_k, j)$: pairwise distance between item i_k and item j
- I_u The set of items previously rated by the user

A high EPD means that the recommended items are different from the items that the user has already interacted with and they are relevant. EPD is a good estimate for the diversity potential for a recommender system as it links to rank, relevance, and diversity in one estimate.

Expected Free Discovery

Another metric introduced by Castells et al. [10] is the Expected Free Discovery (EFD). Contrary to EPC and EPD, it does not take into account the relevance of the model. However, it focuses on the discovery potential of the recommender system.

$$EFD = \frac{-1}{R} \sum_{i \in R} -log_2(p(i|seen)) \tag{42}$$

If the Recommender System recommends items with low estimated exposure then the EFD will be close to one. As this metric does not take into account the relevance, it can provide biased results favoring pure exploration strategies like random selection. However, it can provide insight into the exploration potential of the model.

Gini Coefficient

Another metric that we will use to track the balance in our rating data is the Gini Coefficient. The Gini Coefficient is used to calculate the balance within a given distribution.

$$G = \frac{\sum_i (2i - n - 1)x_i}{n \sum_i x_i} \tag{43}$$

Where

- i: item i
- x_i: number of ratings for item i
- n: number of items used in the population

The Gini coefficient [18] of a uniform distribution is equal to zero. A High Gini coefficient means that the distribution is skewed or imbalanced. The metric helps to track the effect of the feedback loop on the popularity of the items. In a skewed distribution of ratings, only a few items achieve high popularity and the rest are stuck in the long tail. If after each iteration the Gini coefficient keeps increasing, this means that the recommender system is still biasing the recommendations by promoting only popular items.

Hit Rate

Another important evaluation of the performance of our Recommender Systems is the accuracy of the predictions. If the recommendations are diverse and unbiased but they are not relevant, then we cannot consider the designed strategy as successful. For this reason, we will calculate the Hit Rate in each recommendation list. The Hit Rate defines the percentage of items that the user will interact with from all the items in the recommended list. A hit rate of 100% means that the user has interacted with all the items.

$$HR = \frac{|\{i, Hit_i \neq 0\}|}{|R|} \tag{44}$$

Experimental Results

After presenting all the different metrics that we are going to use to evaluate the performance, we will present the results. All the experiments focus on the iterated framework that we presented in the first section.

Models evaluated

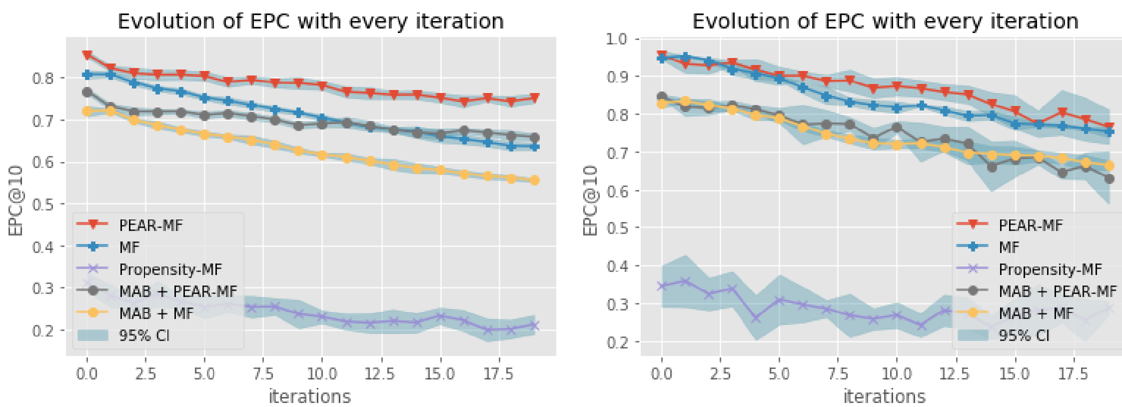
Since our focus is on Matrix factorization models, we will compare the performance of the following models:

- Basic Matrix Factorization [29]
- PEAR-MF
- Propensity-MF [44]
- Naive Multi Armed Bandits Strategy + MF
- Naive Multi Armed Bandits Strategy + PEAR-MF

The main objective is to see how the new regularization strategy is enhancing the performance of the basic matrix factorization. Also, we want to see if the MAB strategy is reducing the exposure bias while maintaining good accuracy.

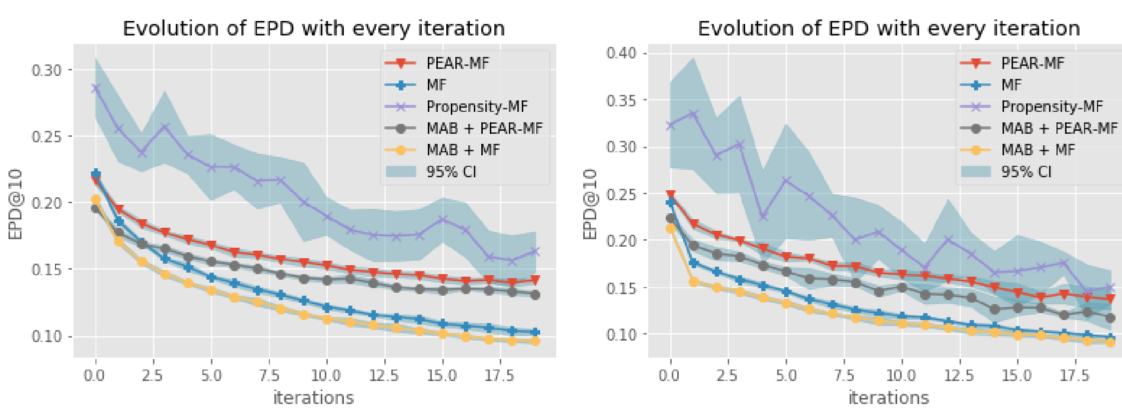
Exposure Bias Results

Figure 20 shows that PEAR-MF significantly (p-value $< 10^{-6}$) outperforms the other strategies in terms of Expected Novelty. It proves that the recommendations provided by PEAR-MF are relevant and also have a low probability of being seen by the user. It is also important to note that EPC is decreasing with every iteration. This confirms the feedback loop effect of recommender systems and how it contributes to creating filter bubbles. We also see that MAB strategies do not help at improving the results because they decrease the relevance of the recommendations. Propensity-MF has a



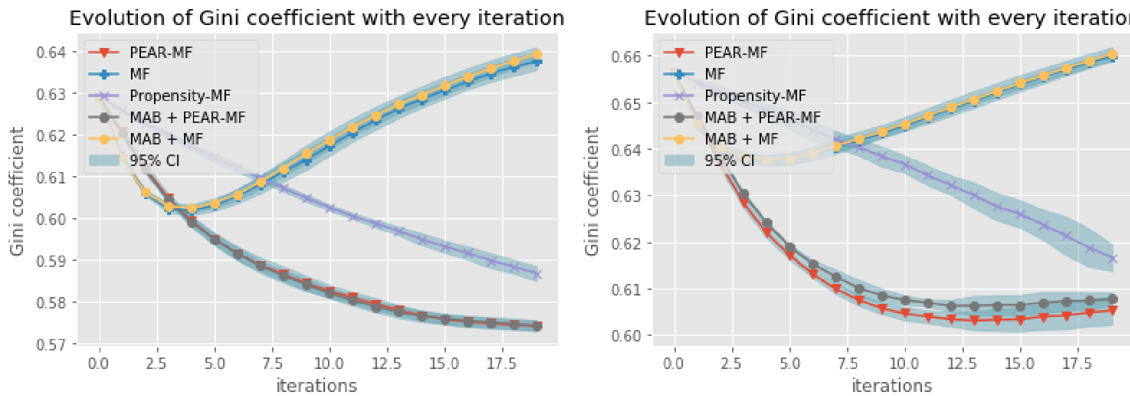
(a) Movie-lens 100K Dataset **(b)** Movie-lens 1M Dataset

Figure 20. The Expected Novelty captures the proportion of new items present in the recommendation list (the higher the better). PEAR-MF significantly outperforms the other models



(a) Movie-lens 100K Dataset **(b)** Movie-lens 1M Dataset

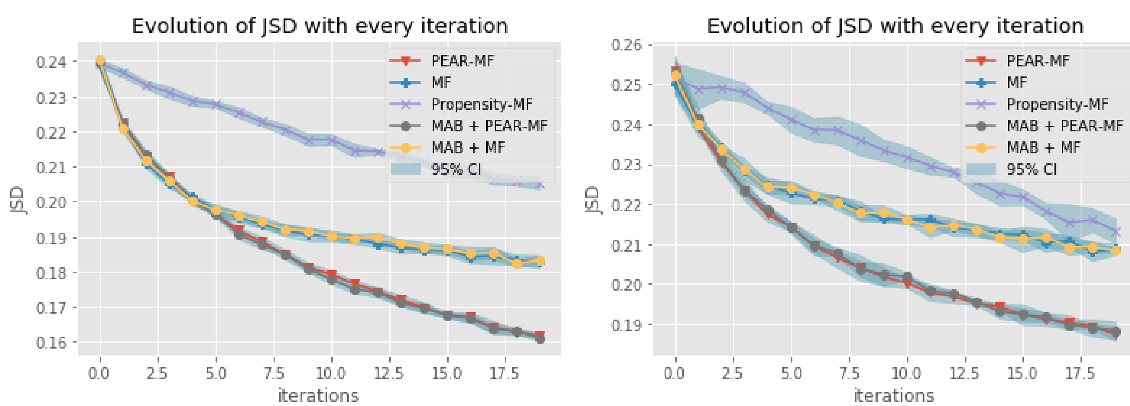
Figure 21. EPD captures the diversity within a recommendation list. The iterated EPD shows that Propensity-MF is performing better by providing more diverse recommendations



(a) Movie-lens 100K Dataset

(b) Movie-lens 1M Dataset

Figure 22. The evolution of the Gini Coefficient is tracked after each iteration. PEAR-MF is decreasing the Gini coefficient after each iteration which proves that it is contributing to decreasing the imbalance in the data



(a) Movie-lens 100K Dataset

(b) Movie-lens 1M Dataset

Figure 23. JSD captures the over-exposure or under-exposure amount for each user. The tracked metric shows that PEAR-MF is decreasing the amount of the exposure bias due to the penalization term in the objective function

lower performance than other methods which is probably due to a low relevance of the recommendations, as confirmed by Figure 24.

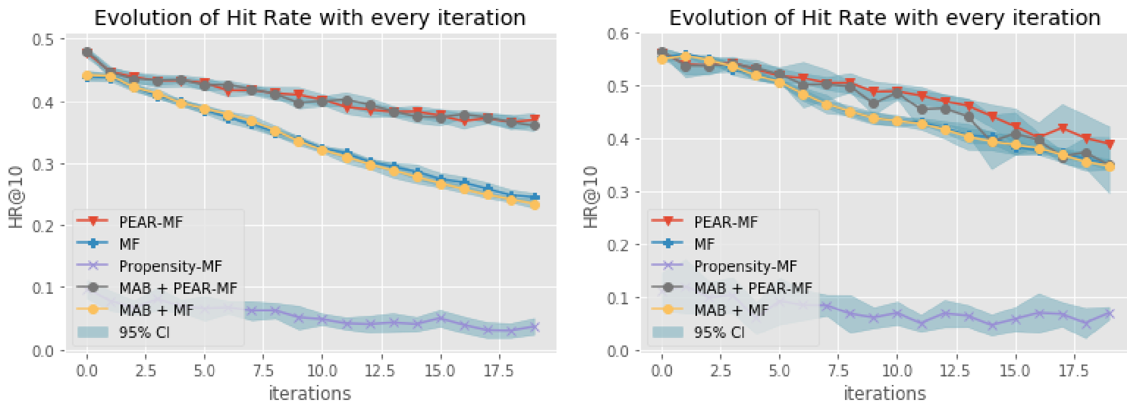
Figure 21 shows that Propensity-MF is significantly better than MF and PEAR-MF at providing diverse recommendations. This is due to the fact that Propensity-MF favors items with very low exposure, hence the items will be more diverse However, the results seem to have a large variance and decreasing with every iteration. PEAR-MF is still outperforming MF by providing more diversified predictions.

Figure 22 shows how PEAR-MF is improving the balance in the rating distribution. After each iteration, the Gini coefficient is decreasing which proves that this method is recommending more items from the long tail. It is also outperforming the regular Matrix Factorization model. The Figure also shows that MF is making the data more imbalanced due to the feedback loop. After each iteration, MF is promoting the popular items and ignoring the long tail items. Propensity-MF is also contributing to balancing the rating distribution by decreasing the Gini coefficient after each iteration.

Figure 23 shows that PEAR-MF is better at decreasing the gap between the user exposure and the uniform distribution. This is expected because the model is optimizing that specific goal (we are penalizing ratings that are not contributing to having a more fair exposure)

Accuracy Results

To test the predictive accuracy of our recommendation strategy, we calculate the hit rate on the provided recommendations and we track its evolution through several



(a) Movie-lens 100K Dataset **(b)** Movie-lens 1M Dataset

Figure 24. The Hit Rate shows the quality of the recommendations at each iteration (the higher the better) PEAR-MF shows that by trying to solve the exposure bias problem we improve the quality of the recommendations

iterations of the feedback loop

Figure 24 shows that PEAR-MF and MAB + PEAR-MF are performing better than MF. This proves that fixing the exposure bias helps to improve the quality of the recommendations. In fact, in an iterated framework, the accuracy may be considered good (inline with the user's taste) but the quality can be low. For instance, a user that keeps seeing the same type of recommendation through several iterations can lose interest and stop interacting with these recommendations. In other words, the interest of the user to these recommendations can fade because of the lack of exploration from the recommender system.

6 Chapter summary

In this chapter, we presented the experimental protocol and results of the project. We defined two experimental approaches. The first set of experiments helped to select the most accurate exposure model. Then we used this model to perform the second set of experiments where we tested the performance of our model in mitigating the exposure bias and the accuracy in an iterated framework.

CHAPTER V

CONCLUSIONS

In this thesis, we achieved three different research objectives. We provided a personalized model to model the user exposure along with the experimental protocol used to confirm its effectiveness. We designed a new regularization model that can mitigate the exposure bias problem. Finally we showed the effect of the feedback loop by running simulations that mimic the life cycle of a Recommender System and showing how to evaluate the extent of the exposure bias problem.

Our work still has limitations. Our main challenge is the data used to evaluate our method. We used semi synthetic data in order to be able to run simulations. A user study will be more accurate, and is planned for the future. Also we worked only with movie data. Other types of data should be investigated like news and social media interactions where the effect of polarization may be more clear.

Another limitation is the relevance of the model that is decreasing with every iteration of the feedback loop. An alternative approach that can include Active Learning or other strategies to enhance the accuracy can be used for the future.

Because recommender systems increasingly control what humans discover, unbiased recommendations improve fairness and expand human discovery potential. For this reason, working on unbiased filtering algorithms is crucial for better quality recommendations.

REFERENCES

[1] 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*, RecSys '17, pages 42–46, New York, NY, USA, 2017. ACM.

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

[3] G. Adomavicius and E Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng*, 7:734–74, 2005.

[4] M. Badami, O. Nasraoui, W. Sun, and P. Shafto. Detecting polarization in ratings: An automated pipeline and a preliminary quantification on several benchmark data sets. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2682–2690, Dec 2017.

[5] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[6] Andrea Barraza-Urbina, Benjamin Heitmann, Conor Hayes, and Angela Carrillo Ramos. Xplodiv: An exploitation-exploration aware diversification approach for recommender systems. In *FLAIRS Conference*, 2015.

[7] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 153–160, Cambridge, MA, USA, 2006. MIT Press.

[8] Dimitrios Bountouridis, Jaron Harambam, Mykola Makhortykh, Mónica Marrero, Nava Tintarev, and Claudia Hauff. Siren: A simulation framework for understanding the effects of recommender systems in online news environments. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 150–159, New York, NY, USA, 2019. ACM.

[9] Heckerman D. Kadie C Breese, J.S. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence*, page 43–52, 1998.

[10] Pablo Castells, Neil J. Hurley, and Saul Vargas. *Novelty and Diversity in Recommender Systems*, pages 881–918. Springer US, Boston, MA, 2015.

[11] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 224–232, New York, NY, USA, 2018. ACM.

[12] Brian M. Oki David Goldberg, David Nichols and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 1992.

[13] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. *Semantics-Aware Content-Based Recommender Systems*, pages 119–159. Springer US, Boston, MA, 2015.

[14] Christian Desrosiers and George Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, Boston, MA, 2011.

[15] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Comput. Sci. Rev.*, 20(C):29–50, May 2016.

[16] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. page 31, 01 2004.

[17] F.A. Gers. Learning to forget: continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN 99*. IEE, 1999.

[18] C. Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. [—.]*. Number pt. 1 in Studi economico-giuridici pubblicati per cura della facoltà di Giurisprudenza della R. Università di Cagliari. Tipogr. di P. Cuppini, 1912.

[19] 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*, NIPS'14, pages 3176–3184, Cambridge, MA, USA, 2014. MIT Press.

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

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

[22] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10:2935–2962, December 2009.

[23] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[24] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[25] Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. Degenerate Feedback Loops in Recommender Systems. *arXiv e-prints*, page arXiv:1902.10730, Feb 2019.

[26] D. Gupta K. Goldberg, T. Roeder and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 2001.

[27] Michael N. Katehakis and Arthur F. Veinott. The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268, may 1987.

[28] Yehuda Koren and Robert Bell. *Advances in Collaborative Filtering*, pages 145–186. Springer US, Boston, MA, 2011.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[30] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *CoRR*, abs/1402.6028, 2014.

[31] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA, 2000. MIT Press.

[32] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010.

[33] 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*, WWW '16, pages 951–961, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[34] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[35] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3085–3094, New York, NY, USA, 2014. ACM.

[36] Jonathan Louedec, Max Chevalier, Josiane Mothe, Aurélien Garivier, and Sébastien Gerchinovitz. A multiple-play bandit algorithm applied to recommender systems. 05 2015.

[37] Benjamin M. Marlin, Richard S. Zemel, Sam T. Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. *CoRR*, abs/1206.5267, 2012.

[38] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. Exploring the filter bubble: The effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 677–686, New York, NY, USA, 2014. ACM.

[39] E. Pariser. *The Filter Bubble: How the New Personalized Web is Changing what We Read and how We Think*. Penguin Books, 2012.

[40] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 784–791, New York, NY, USA, 2008. ACM.

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

[42] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[43] 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*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

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

[45] Patrick Shafto and Olfa Nasraoui. Human-recommender systems: From benchmark data to benchmark cognitive models. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 127–130, New York, NY, USA, 2016. ACM.

[46] Guy Shani and Asela Gunawardana. *Evaluating Recommendation Systems*, pages 257–297. Springer US, Boston, MA, 2011.

[47] Meenakshi Sharma and Sandeep Mann. A survey of recommender systems: Approaches and limitations. *International Journal of Innovations in Engineering and Technology*, 2013.

[48] Dominic Spohr. Fake news and ideological polarization: Filter bubbles and selective exposure on social media. *Business Information Review*, 34(3):150–160, 2017.

[49] 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*, KDD '10, pages 713–722, New York, NY, USA, 2010. ACM.

[50] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 125–132, New York, NY, USA, 2011. ACM.

[51] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[52] Wenlong Sun, Olfa Nasraoui, and Patrick Shafto. Iterated algorithmic bias in the interactive machine learning process of information filtering. 09 2018.

[53] Virginia Tsintzou, Evaggelia Pitoura, and Panayiotis Tsaparas. Bias disparity in recommendation systems. *CoRR*, abs/1811.01461, 2018.

[54] Jacek Wasilewski and Neil Hurley. Incorporating diversity in a learning to rank recommender system. In *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida, USA, May 16-18, 2016.*, pages 572–578, 2016.

[55] Olfa Nasraoui Wenlong Sun, Sami Khenissi and Patrick Shafto. Debiasing the human-recommender system feedback loop in collaborative filtering. Humbl2019.

[56] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 279–287, New York, NY, USA, 2018. ACM.

[57] Koren J Zhang, Y. Efficient bayesian hierarchical user modeling for recommendation system. *Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 47–54, 2007.

CURRICULUM VITAE

NAME: Sami Khenissi

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.Sc., Computer Science & Engineering
January 2018 - May 2019
GPA: 4.0
University of Louisville, *Louisville, Kentucky*

B.Eng., Applied Mathematics
September 2014 - June 2017
Ecole Polytechnique de Tunisie, *Tunis, Tunisia*

Employment:

Research Assistant
April 2017 - Present
University of Louisville, *Louisville, Kentucky*

Data Scientist Intern
June 2016 - August 2016
Digisponsor, *Paris, France*

Software Engineer Intern
June 2015 - August 2015
Orange, *Tunis, Tunisia*

Publications

Wenlong Sun, Sami Khenissi, Olfa Nasraoui, and Patrick Shafto. 2019.
Debiasing the Human-Recommender System Feedback Loop in Collaborative Filtering. In Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.
ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308560.3317303

Achievements and Awards

- Dean Citation for academic excellence: April 2019
- Master of Science Award: April 2019
- Best Poster Award in the Speed School E-Expo April 2019
- Top 1% student in the National Exam for Admittance to Engineering Schools Tunisia: September 2014