REVEALING AND ADDRESSING BIAS IN RECOMMENDER SYSTEMS

A Dissertation

by

XING ZHAO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2021

Major Subject: Computer Science

ABSTRACT

In the past decades, recommenders have achieved outstanding success in delivering personalized and accurate recommendations. Highly customized recommendations bring individuals great convenience, while helping content providers to connect with interested consumers accurately. However, they may also lead to undesirable outcomes, often through the introduction of bias in the training, deployment, and maintenance of recommender systems. For example, recommenders may impose unfair burdens on certain user groups, disadvantaging their prominence on job-based recommenders. And they may narrow down a user's interest areas, raising concerns of echo chambers, fairness, and diversity. In this dissertation, we ground our work in identifying gaps in the literature for identifying and addressing bias in recommender systems, then introduce four approaches for mitigating biases from multiple perspectives, listed as below:

- First, we identify an inherent bias in many recommendation algorithms which optimize for the head (or popular portion) of the rating distribution, thus lead to large estimation errors for tail ratings. We conduct a data-driven investigation and theoretical analysis of the challenges posed by traditional latent factor models for estimating such tail ratings. With these challenges in mind, we propose a new multi-latent representation method designed specifically to estimate these tail ratings better, to reduce the bias associated with tail ratings in recommender systems.

- Second, we address another unexplored bias – the target customer distribution distortion. Traditional recommender systems typically aim to optimize an engagement metric without considering the overall distribution of target customers, thereby leading to serious distortion problems. Through a data-driven study, we reveal several distortions that arise from conventional recommenders. Toward overcoming these issues, we propose a target customer re-ranking algorithm to adjust the population distribution and composition in the Top-k target customers of an item while maintaining recommendation quality.

- Third, we focus on mitigating the next unexplored bias – user's taste distortion. We show how existing approaches assume a static view of user's tastes, and so previously proposed calibrated recommenders result in poor modeling of the shift of a user's evolution. Thus, we empirically identify the taste distortion problem through a data-driven study over multiple datasets. We propose a taste-enhanced calibrated recommender system designed with the shifts and trends of user's taste preferences in mind, which results in improved taste distribution estimation and recommendation results.

- Last but not least, we study the distribution bias in a dynamic recommendation environment. Previous studies of such distribution-aware recommendation have focused exclusively on static scenarios, ignoring important challenges that manifest in real-world dynamics and leading to poor performance in practice. Hence, we present the first study of distribution bias in dynamic recommendations, and propose new methods to mitigate this bias even in the presence of feedback loops and other dynamics.

DEDICATION

To my family.

ACKNOWLEDGMENTS

I would like to express my thanks and gratitude to the people who have helped and supported me during my Ph.D. life.

First and foremost, I would like to express my deepest gratitude to my advisor Dr. James Caverlee, for his outstanding mentorship, supports, and guidance. Time back in 2016, I chose him as my advisor, which became one of my best choices I have ever made in my life. During these years, he continuously fuels me with impetus and passion for research in daily life, making me from a starter to be a better researcher. I have benefited much from his insightful research suggestions and professionalism. He taught me how to find problems from existing research works with critical thinking, bring up research ideas, effectively arrange the research processes, and continuously improve the research results. His guidance and support to me ran through the entire process. This dissertation would have never been done without his great mentorship. Moreover, his influence on me is by no means limited to the research works. It is hard to forget that every time I face difficulties and disappointments, he always encourages me and helps me build confidence again and again. His respectable personality, including his patience, kindness, and generousness, made him a permanent role model and life mentor for me and shaped me into a better person.

I would like to acknowledge with gratitude the love and caring from my family. A special feeling of gratitude to my loving parents whose words of encouragement and push for tenacity ring in my ears. They provided me an excellent education and environment to grow up in. This dissertation has never been done without their unconditional support and selfless devotion. I would also like to thank my brother Chen and his family, who have never left my side and are very special. I give special thanks to my wife, Lei, and my daughter, Amie, for being on this journey with me together. I always appreciate all their love and support throughout this process.

Many thanks to my lab-mates for their support and helpful discussions. I enjoyed all the time we spent together in the office, making jokes, exchanging ideas, etc. We shared the beautiful moments of paper acceptance as well as the sad moments of being rejected. All these became an unforgettable memory in my life. It is still fresh in my memory when I joined the lab. Our senior lab members, Hancheng Ge, Haokai Lu, Wei Niu, Cheng Cao, and Majid Alfifi, welcomed me and helped me understand how to become a successful researcher. They also support me greatly in my follower research projects with their excellent research experience and technical expertise. Special thanks to Ziwei Zhu for the collaboration on several projects. It is my honor to have his name as the co-author for many of my published papers. Many thanks to my colleagues, Yun He, Parisa Kaghazgaran, Jianling Wang, Zhuoer Wang, and Yin Zhang, for discussing and exchange research ideas. We worked together through these years and made the whole lab full of warmth and joy. Additionally, I would like to thank my former lab-mate, Zhiyuan Cheng, for his excellent advice and guidance for my future career.

Finally, I'm grateful to Dr. Xia Hu, Dr. Thomas Ioerger, and Dr. Huiyan Sang for generously being my committee members. Thank them for the constructive comments and feedback that makes my research work more polished and valuable.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor James Caverlee [advisor], Professor Xia (Ben) Hu, and Professor Thomas Ioerger of the Department of Computer Science & Engineering and Professor Huiyan Sang of the Department of Statistics.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported in part by DARPA and NSF grants.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background and Motivation

In the past decades, recommenders have achieved outstanding success in delivering personalized and accurate recommendations. Recommender systems have flooded every corner of our lives. By analyzing the past behaviors of users and finding patterns between users and items, recommenders connect us to movies on streaming media platforms [1, 2, 3], jobs to apply for [4, 5, 6], items to purchase on e-commerce sites [7, 8, 9], new stories to read [10, 11, 12], and even itineraries for upcoming trips [13, 14, 15]. Furthermore, recommenders are driving new developments in personalized health and emerging applications in machine learning [16, 17]. Indeed, these highly customized recommendations bring individuals great convenience, while helping content providers to more accurately connect with interested consumers.

Research in recommendation systems has historically emphasized the accuracy of recommendations. Such traditional recommenders often focus on accuracy-like metrics that emphasize whether a user will interact with the recommended content. For example, will the user view a particular movie or purchase a particular product? Naturally, this emphasis on accuracy-like metrics may ignore vital bias concerns.

Accuracy-driven recommenders may also lead to undesirable (and often unforeseen) outcomes, often through the introduction of *bias in the training, deployment, and maintenance of recommender systems*. For example, recommenders may impose unfair burdens on certain user groups, disadvantaging their prominence on job-based recommenders through gender bias.[*] They may augment racial bias in crime prediction.[†] Recommenders may narrow down a user's interest areas, raising concerns of echo chambers [18], fairness [19, 20] and diversity [21, 22, 23]. This "down the rabbit hole effect" may lead an algorithm to continually recommend the same type of content (since it maximizes engagement metrics) even if the content is not societally optimal. Recommenders may

[*]https://factorialhr.com/blog/gender-bias-recruitment/

[†]https://medium.com/thoughts-and-reflections/racial-bias-and-gender-bias-examples-in-ai-systems-7211e4c166a1

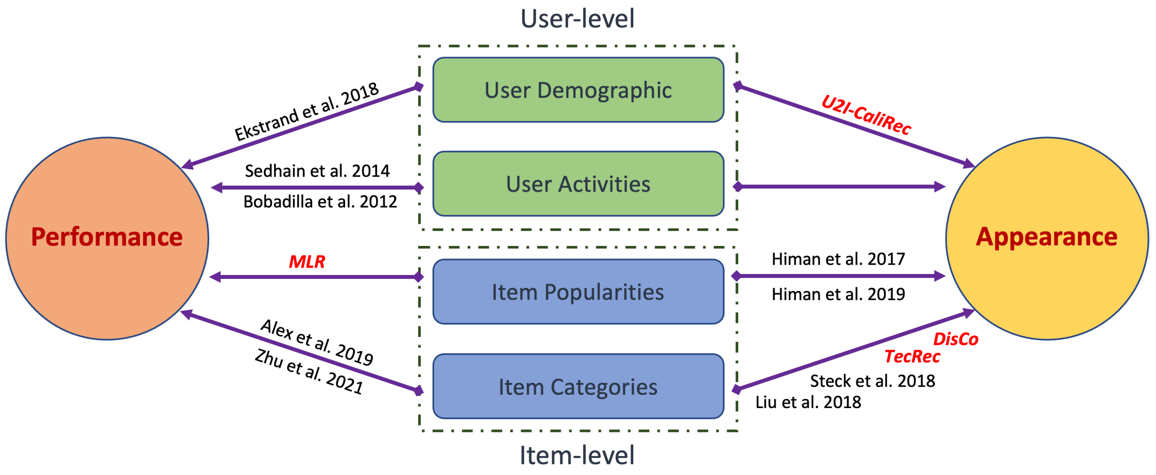


Figure (1.1) The conceptual framework of recommendation bias. On each arrow, there are also related representative de-biasing research. The methods in red – *MLR*, *U2I-CaliRec*, *TecRec*, and *DisCo* – are the contributions introduced in this dissertation research.

propagate unexpected demographic distributions; for example, a biased prediction of the demographics of target customers (e.g., by gender or age) could lead to the loss of potential consumer groups by mis-targeting the recommendations [24]. In one extreme, there are reports of recommenders facilitating the exploitation of minors[‡].

Recently, many efforts have begun investigating bias and fairness issues in recommendation systems, and focused on building bias-aware recommender systems for certain types of bias concerns, including studies of selection bias [25, 26, 27, 28], exposure bias [29, 30, 31, 32], popularity bias [33, 34, 35, 36], and position bias [37, 38, 39, 40], among many others. While there is increasing recognition of the challenges of bias in recommendation, many existing works focus on individual instances of bias. Hence, this dissertation research is grounded in a comprehensive conceptual framework for identifying bias in recommender systems from multiple perspectives. Figure 1.1 shows the main structure of our conceptual framework. Briefly speaking, it views bias from the perspective of: (1) two types of bias: appearance and performance; and (2) two affected objects: users and items. We now introduce each of the concepts and perspectives as follows.

Bias Types: Appearance vs. Performance. *Appearance* focuses on the constituents of the recom-

[‡]https://techcrunch.com/2019/02/18/youtube-under-fire-for-recommending-videos-of-kids-with-inappropriate-comments/

mendation list returned from a recommendation system. For example, for a job recommendation platform such as LinkedIn or Glassdoor, the appearance is the predicted male-female ratio for a given job position. In this context, we compare the predicted gender ratio with a target distribution. We may want the target gender ratio as 1:1 for fairness concerns, or as an unbalanced but reasonable one for some positions. In this case, we expect an unbiased recommender should return a gender-ratio close to the desired one.

On the other hand, *performance* focuses on the prediction quality of the recommendation list for the recommender system. For example, for males and females, will a system equally treat them by providing recommendations with the same quality? If not, it will raise bias or fairness concerns. Unlike the appearance, the performance is a pure fairness consideration which keeps the principle of “everyone created equal” in mind. Generally, we expect an unbiased recommender should return an equal-quality prediction for every user or item group.

Objects: User-level vs. Item-level. Next, we consider bias problems from the user and item perspective, respectively. We have the *user-level* consideration, such as their demographics and activities; and we have the *item-level* consideration, such as items’ categories and popularity. All these factors could raise bias issues in recommendation results, though the relative importance are application dependent. For example, from an advertisement display platform’s perspective, we may want to give a prominent position for an item that is not popular yet but has great potential. Therefore, a cold-item-friendly recommender should be the priority. On the other hand, from the perspective of a user, we care more about the quality (or accuracy) of personalized recommendations. For example, the user’s activity is an indicator of how long or how active a user interacted with the platform. Here, in this context, bias is the difference in performance among users with different activities, e.g., new or loyal users.

Even framing the bias in recommender systems, there are still many research opportunities and challenges remain. For example, how do we identify bias in different settings of recommendation, and how do we mitigate these biases? With this conceptual framework, we show in Figure 1.1 representative de-biasing research efforts that align with these two perspectives: both from the *performance* and *appearance* perspective and from the *user-level* and *item-level* perspective.

1.2 Contributions

Through this conceptual framework, this dissertation identifies gaps in the literature for identifying and addressing bias in recommender system. It then couples this conceptual framework with four new methods for mitigating bias in recommender system (highlighted in red in Figure 1.1). In sum, this dissertation is structured around the following unique contributions:

- **Mitigating Bias of Item's Rating Estimation:** In our first contribution, we identify an inherent bias in many recommendation algorithms to optimize for the head (or popular portion) of the ratings distribution, leading to large estimation errors for tail ratings. We conduct a data-driven investigation and theoretical analysis of the challenges posed by traditional latent factor models for estimating such tail ratings. These approaches assume a single latent representation, leading to over- and under-estimations of tail ratings, with particularly pronounced errors on controversial items. With these challenges in mind, we propose a new multi-latent representation method designed specifically to estimate these tail ratings better, to reduce the bias associated with tail ratings in recommender systems. Experimental results show the estimation improvement is especially great for those items far from the ratings mean. Furthermore, the proposed model is generalizable and can be easily extended to take advantage of other single-latent-representation-based models.

- **Mitigating Bias of Item's Target Customer Distribution:** In our second contribution, we address another unexplored bias problem – the target customer distribution distortion issue. Traditional recommender systems typically aim to optimize an engagement metric without considering the overall distribution of target customers, thereby leading to serious distortion problems. Through a data-driven study, we reveal several distortions that arise from conventional recommenders. Toward overcoming these issues, we propose a target customer re-ranking algorithm to adjust the population distribution and composition in the Top-k target customers of an item while maintaining recommendation quality. By applying this proposed algorithm onto a real-world dataset, we find the proposed method can effectively make the class distribution of items' target customers close to the desired distribution, thereby mitigating distortion.

- **Mitigating Bias of User's Dynamic Taste Distribution:** In our third contribution, we focus on mitigating a related unexplored bias problem – user's dynamic taste distortion issue. We show how existing approaches assume a static view of user's tastes, and so previously proposed calibrated recommenders result in poor modeling of the dynamics of a user's evolution. Thus, we empirically identify the taste distortion problem through a data-driven study over multiple datasets. We show how taste preferences dynamically shift and how a calibration mechanism should be designed with these shifts in mind. We further demonstrate how to incorporate these preference shifts into a taste enhanced calibrated recommender system, which results in improved taste distribution estimation and recommendation results.

- **Mitigating Bias of Distribution in Dynamic Recommendation:** In our fourth contribution, we present the first study of distribution bias in dynamic recommendations, and propose new methods to mitigate this bias even in the presence of feedback loops and other dynamics. We first conduct a data-driven study of distribution bias in a dynamic environment, where we find conventional distribution-aware recommenders may not only raise more serious distribution bias in the dynamic environment, but also cause a significant fall in recommendation performance. We then propose a distribution correction method to better model the shift of a user's preferences in the dynamic recommendation loop and better match the preference distribution at each step of the dynamic process.

1.3 Dissertation Overview

The remainder of this dissertation is organized as follows:

- **Chapter 2: Related Work.** In this chapter, we discuss related work, specifically latent factor recommendation model, rating distribution, calibration, and several different type of recommendation systems.

- **Chapter 3: Improving the Estimation of Tail Ratings in Recommender System with Multi-Latent Representations.** In this chapter, we introduce MLR for better improving the estimation of tail ratings by extending traditional traditional methods with new multi-latent representations for better modeling these tail ratings.

- **Chapter 4: Addressing the Target Customer Distortion Problem in Recommender Systems.** In this chapter, we introduce U2I-CaliRec – a target customer re-ranking algorithm – to adjust the population distribution and composition in the Top-k target customers of an item while maintaining recommendation quality.

- **Chapter 5: Rabbit Holes and Taste Distortion: Distribution-Aware Recommendation with Evolving Interests.** In this chapter, we introduce TecRec – a post-ranking framework incorporating a time-series neural network model to predict users' preference shifts – that can strongly improve both taste distribution estimation (i.e., mitigating both the rabbit hole effect and the taste distortion problem) and recommendation quality in the static recommendation environment.

- **Chapter 6: Mitigating Distribution Bias in Dynamic Recommendation.** In this chapter, we introduce DisCo – a Distribution Correction Recommender that dynamically corrects preference distribution in interactive recommendations. Results show that DisCo improves the closeness of the preference distribution in the recommendation with users' real future preference, and increases users' agreement with the preference distribution of the recommendations. Results also indicate that the recommendation quality can be improved compared with accuracy-driven recommenders and conventional distribution-aware recommenders, giving more evidence that there is not always a trade-off between calibration and recommendation.

- **Chapter 7: Conclusions and Future Work.** We conclude the dissertation with a summary of contributions, and discuss potential research extensions to the results presented here.

2. RELATED WORK

This chapter highlights related works in conventional accuracy-driven recommendation systems and de-biasing approaches for bias that may arise in recommendation results.

2.1 Accuracy-driven Recommendation

Traditional accuracy-driven recommenders aim to improve the accuracy of predicted user-item relevance. In this section, we introduce related works of latent factor models, sequential recommendations, and dynamic recommendations.

2.1.1 Latent factor models

The latent factor model is one of the cornerstones of modern recommender systems, critical for traditional approaches [41, 42] as well as recent neural variants like NCF [43] and others [44, 43, 45, 46, 47, 48]. Furthermore, these latent factor models have been adapted in a number of directions, including location-aware recommendation systems [49, 50, 51], aspect-aware latent factor models [52], and bio-inspired approaches [53, 54], among many others. Latent factor models typically depend on an assumption of a *single latent representation*. That is, every item and user has only a single latent representation. We refer to such approaches as *Single Latent Representation* (SLR)-based methods.

At the core of these latent factor models, it is assumed that both items and users live in a low-dimensional latent space, where the latent factors typically capture user preferences and item characteristics. For instance, Matrix Factorization finds the optimal low-rank matrix $P^{m\times r}$ and $Q^{n\times r}$, representing user latent factors and item latent factors respectively, such that $P \cdot Q^T$ is close to the original rating matrix $M^{m\times n}$, where r is the predefined low rank, m is the number of users, and n is the number of items.

$$arg \min_{P,Q} (M - P \cdot Q^T)^2 \tag{2.1}$$

If M is the user-item rating matrix, each row of P and Q corresponds to a user latent factor and an item latent factor, respectively. Since each user and item corresponds to a single row in P or Q, we say this is an SLR-based method. In practice, many latent factor models incorporate bias terms for users and items and a global offset into the prediction model. For clarity in the discussion, consider the classic matrix factorization model (MF) with bias:

$$\hat{r} = p_u \cdot q_i^T + b_u + b_i + \mu \tag{2.2}$$

where $\hat{r}$ is the estimated rating. In this case, p_u and q_i are the latent representations for user u and item i, respectively. The bias terms capture user bias (b_u), item bias (b_i), and a global offset (μ).

More recently, neural variants like Neural Collaborative Filtering (NCF) have been proposed to combine deep learning architectures with traditional matrix factorization [43]. In particular, NCF is structured with two sub-models: Generalized Matrix Factorization (GMF) and a Multi-Layer Perceptron. The GMF submodel corresponds to a neural version of MF, and so also relies on a single latent vector for representing a user's preference or an item's characteristics.

2.1.2 Sequential Recommendation

Sequential recommendation is designed specifically to track a user's changing interests to improve recommendation accuracy, e.g., [55, 56, 57, 58, 59]. Sequential recommenders typically combine personalized models of user behavior with a context defined by a user's recent activities. For example, Hidasi *et al.* introduced GRU4Rec which employs Gated Recurrent Units (GRU) to model users' click sequences for session-based recommendation [60]. Soon after, they improved their original work with a version that further boosts Top-N recommendation performance [61]. Kang *et al.* proposed a self-attention based sequential recommender system, SASRec, which models the entire user sequence, and adaptively considers consumed items for prediction [56]. Other sequential recommender systems also achieve impressive performance, e.g., [57, 62, 63]. These methods use sequential activities to model a user's latent preferences in the next stage, however, they are not fundamentally designed with recommendation distribution in mind. Therefore, the resulting recommendations could lead to biases such as rabbit holes and echo chambers.

2.1.3 Dynamic Recommendation

Dynamic recommendations can be viewed as a closed-loop where users interact with the system through a set of actions (e.g., clicks, views, ratings); this user-feedback data is then used to train a recommendation model; the trained model is used to recommend new items to users; then the loop continues. In this dynamic recommendation setting, there have been many efforts to identify sources of bias [64, 65, 66, 67]. Chaney *et al.* explored the impact of algorithmic confounding on a range of simulated recommendation systems [64]. They stated that algorithmic confounding amplifies the homogenization of user behavior without corresponding gains in recommendation utility and amplifies the impact of recommendation systems on item consumption distribution. Khenissi *et al.* theoretically proved that in the dynamic environment, recommender system tends to limit the discovery of users through iterative recommending [65]. Recently, Morik *et al.* studied how biased feedback and uncontrolled exposure in a dynamic learning-to-rank system could lead to unfairness and undesirable recommendation behaviors [66].

2.2 De-biasing Approaches

We already introduced different types of bias in recommendation systems in Chapter 1. In this section, we introduce de-biasing approaches designed for explicit rating distribution, preference distribution, and diversification.

2.2.1 Studies of Explicit Ratings distributions

Gediminas *et al.* investigated the impact of rating characteristics like rating density, rating frequency distribution, and value distribution, on the accuracy of popular collaborative filtering techniques [68]. Hu *et al.* observed that product ratings tend to fit a 'J-shaped' distribution [69] since users provide reviews are more likely to "brag or moan" compared to all purchasers. As an extreme case of the 'J-shaped' distribution is the 'U-shape' of *controversial items* with many extreme ratings on both sides of the distribution. Victor *et al.* in [70] formalized the concept of controversial items in recommendation systems and then compared the performance of several trust-enhanced techniques for personalized recommendations for controversial items with polarized ratings (bi-modal distribution) versus other items. Similar to our observations, they showed that predicting ratings for controversial items is much worse than for other items. Badami surveyed state-of-the-art research on the polarization [71], finding that many trust-based recommender system attempts to improve recommendation for controversial items by defining a trusted network for each user, e.g., [72, 73, 74, 75]. Recently, Beutel *et al.* proposed a focused learning model to improve the recommendation quality for a specified subset of items, through hyper-parameter optimization and a customized matrix factorization objective [76].

2.2.2 Preference Distribution-aware Recommendation

Recently, Steck proposed a distribution-aware recommendation (what we refer to as CaliRec), which focuses on the distribution of genres in a user's recommendation list [18]. This work aims to reflect the user's interest areas in the recommendation result to ameliorate the rabbit hole effect through re-processing the output of other recommender systems. Soon after, Kaya and Bridge compared Steck's work with intent-aware recommendations, and proposed a new version of distribution-aware approaches and new evaluation metrics [77]. These studies in distribution-aware recommendation make the strong assumption that the target distribution for recommended results should fit the overall distribution in the training data. This assumption essentially asserts that a user's preferences are mature and fixed, and so post-processing should aim to match these fixed preferences. Besides, existing works have focused primarily on one-shot static settings, ignoring the distorting effects present in dynamic environments. Furthermore, these and related studies have shown that the distribution adjustment and accurate recommendation tend to trade-off with each other [78, 18, 24].

Distribution analysis is a long-standing problem in the statistics field, especially regarding sample data from a population. Stratified sampling [79, 80, 81] is a sampling method from a population that can be partitioned into sub-populations to better understand and represent the population. It achieved great success in the statistical field but has not been fully adopted into the research of personalized recommendation. Specifically, stratified sampling focuses on the representativeness of strata to the whole population. And stratification is the process of dividing members of the population into homogeneous subgroups before sampling. Therefore, it is worth considering stratified sampling to analyze the distribution without the dynamic changing, e.g., demographic distribution. However, it would be challenging to use such a traditional statistical method to analyze the temporal-based distribution, e.g., a dynamic preference distribution in recommendations. That is, stratification may ignore the trends and shifts of the change among each subgroup, which would bring the poor estimation of the distribution in the next stage.

2.2.3 Diversity-focused Recommendation

Rather than aiming to match a user's taste distribution as distribution-aware recommenders, some methods aim to introduce diversity into the list of ranked results [82, 83, 84, 22, 85, 86]. For example, a diversity-focused recommender may use the category of an item or the genre of a movie as an "aspect," with the aim of covering many diverse aspects [87, 88]. Some methods eschew such explicit aspects in favor of identifying latent aspects as the basis of diversification [89].

These diversity-focused approaches can ameliorate the *rabbit hole effect*, but do so in a different fashion from distribution-aware recommendations. For example, if a user watched 70 romance and 30 action movies, a diversity-focused recommender system would avoid recommending 100% romance movies; however, it may recommend other genres to achieve diversity, like horror, documentaries, or other genres that are not reflected in the user's taste distribution. Therefore, compared with distribution-aware recommendations, diversity-focused recommendations cannot provide the interpretable diversification and provide possibilities to control the diversification to avoid undesirable recommendation[24].

Furthermore, diversity-focused and distribution-aware recommender systems also differ in terms of evaluation. Traditional diversity-focused recommenders use feature-based metrics, such as $\alpha\text{-}NDCG$ [90] or subtopic recall ($S\text{-}recall$) [91]. Such metrics evaluate diversity (e.g., coverage of aspects) of the recommended results. However, distribution-aware recommender systems evaluate the taste distribution's closeness in the recommended results and a target one.

3. IMPROVING THE ESTIMATION OF TAIL RATINGS IN RECOMMENDER SYSTEM WITH MULTI-LATENT REPRESENTATIONS*

Based on the conceptual framework of bias in recommendations (recall Figure 1.1), we begin our study with the estimation bias for items with different ratings. For example, how can we predict which popular movies a user will **not** like? Or which users will like an app that has received mixed reviews? And for controversial items with polarized ratings (e.g., political books), how can we ensure that we recommend items to the right subset of users? While recommender systems have made great strides in connecting users to the right items – be it on YouTube, Yelp, Netflix, or Amazon – there are still great challenges in estimating these *tail ratings* that are far from the mean rating for many items.

We say that the *tail ratings* are ratings from a user to a specific item that are significantly lower or significantly higher than an item's average rating, typically accounting for a smaller fraction of all ratings on an item. For example, Figure 3.1(a) shows the rating distribution for six different data sets, all of which have a majority of ratings in the upper ranges (with a mean rating of 3.3 to 4.3). The tail ratings for Amazon Books could be defined as the ratings of 1 or 2 (significantly below the average 4.09) that account for a small fraction of all ratings. The tail ratings for MovieLens could be defined as the 1-2 ratings, as well as the 5 ratings (well below and above the average 3.35 rating). In contrast, the *head ratings* are those ratings that are close to the average rating, typically accounting for the majority of all ratings on an item.

While the importance of the distribution of ratings on recommender system has been long recognized, e.g., [68, 71, 69, 70], many popular methods based on latent factor models and recently introduced neural variants [92, 93, 94, 43, 45, 46] optimize for the head of these distributions, potentially leading to large estimation errors for tail ratings. As we will show in Section 3.1,

*Reprinted with permission from "Improving the Estimation of Tail Ratings in Recommender System with Multi-Latent Representations" by Xing Zhao, Ziwei Zhu, Yin Zhang, and James Caverlee, 2020. Proceedings of The Thirteenth ACM International Conference on Web Search and Data Mining. Copyright 2020 by ACM. DOI: https://doi.org/10.1145/3336191.3371810 these tail estimation errors are common across multiple domains and datasets, leading to large over-estimations of the ratings of items with very low ratings, and large under-estimations of the ratings of items with very high ratings. For example, Figure 3.1(c) shows large RMSE prediction errors for these tail ratings when using two popular latent factor models. These errors can lead to bad recommendations, degrade trust in the recommender, and for controversial items, potentially expose users to items they are diametrically opposed to.

In this chapter, we study the problem of estimating tail ratings with an emphasis on improving the quality of these estimates within latent factor models that drive many modern recommenders. We show how many existing methods rely on an assumption of a *single latent representation* (what we refer to as SLR; a discussion can be found in Chapter 2) that leads to large errors in tail rating estimation. We conduct a data-driven investigation of the limitations of such an assumption underlying these models whereby ratings are assumed to fit a uni-modal distribution. Paired with this investigation, we formally analyze the limitations of these single latent representation methods with respect to tail ratings. With these limitations in mind, we propose a new method which is designed to learn *multiple latent representations* for better modeling these tail ratings. In this way, the estimation of tail ratings can escape the constraint of a uni-modal distribution. We show how to incorporate these multi-latent representations in an end-to-end neural prediction model that is designed to better reflect the underlying rating distributions of items. Through experiments over six datasets from Amazon, Goodreads, and MovieLens, we find the proposed model leads to a significant improvement in RMSE versus a suite of benchmark methods. We also find that the predictions for the most polarized items are improved by more than 15%.

This chapter is structured as follows. Section 3.1 introduces the datasets used in this chapter, followed by a data-driven investigation and theoretical analysis of the limitations of traditional latent factor models for dealing with tail ratings. We introduce our proposed multi-latent representation approach in Section 3.2, and then evaluate it over multiple datasets in Section 3.3. Finally, we summarize this work in Section 3.4.

	# Users	# Items	# Ratings	# Avg
Amazon Books	3,824	9,640	172,018	4.09
Amazon Digital Music	5,541	3,568	64,706	4.22
Amazon Kindle	68,223	61,934	982,619	4.35
Amazon CD & Vinyl	75,258	64,443	1,097,592	4.29
GoodReads	2,671	7,702	195,174	4.00
MovieLens	610	9,724	100,836	3.35

Table (3.1) All datasets have a global mean around 3.35 to 4.35, with tail ratings in the lower (1-2) and upper (5) portions of the distribution.*

3.1 A Data-driven Study of Tail Ratings

In this section, we conduct a two-part investigation of the *single latent representation* assumption underlying latent factor models like MF and NCF and how this impacts tail ratings. Firstly (Section 3.1.1), we demonstrate the challenges in estimating tail ratings across six datasets that are due to the fundamental uni-modal latent factor distribution. Secondly (Section 3.1.2), we formally analyze the limitations of SLR-based methods with respect to tail ratings.

3.1.1 Data-Driven Study

We use six ratings-based datasets: Amazon Books, Amazon Digital Music, Amazon Kindle, and Amazon CDs & Vinyl [95], MovieLens [96], and GoodReads [97]. For each, we adopt the N-core selection criteria which have been shown to lead to more robust training and evaluation: that is, each user gives at least N ratings, and each item receives at least N ratings. Specifically, we use 12-core for Amazon Books and 5-core for the others. For MovieLens, we consider users who have rated at least 20 movies. For the following analyses and experiments, we randomly split the ratings of each user into training, validation, and test sets using a random 60%, 20%, 20% split. Details of these datasets are shown in Table 3.1.

Observations: All Ratings. For each dataset, we estimate ratings of the test set using the standard latent factor model in Equation 2.2 (MF) and a neural variant based on NCF, since these two are foundational for both traditional and neural recommenders.

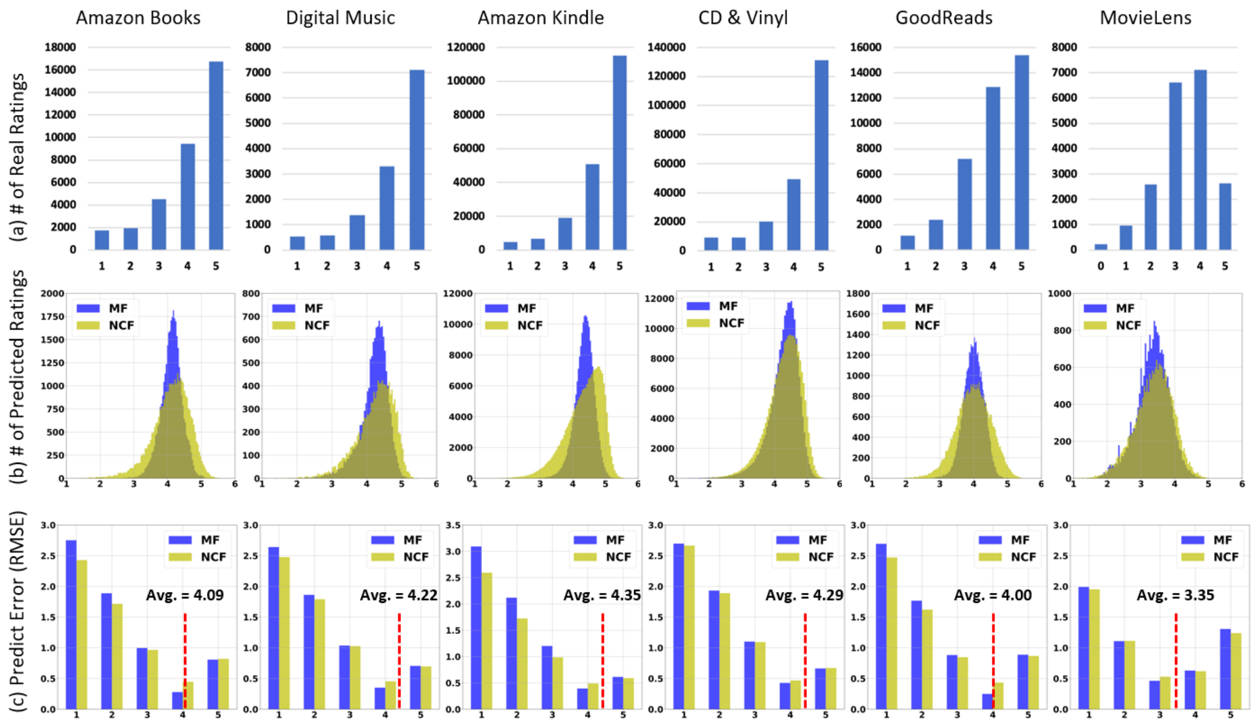


Figure (3.1) The count of ratings for six different datasets (a); the predicted ratings by MF and NCF are uni-modal as shown in (b); meaning that errors are concentrated in the tails of the predicted distributions (c).*

Figure 3.1 (a) shows the original rating distribution in the test set for each of our six datasets. As we can see, from the perspective of all ratings, the count of the original ratings fits a uni-modal distribution. The tail ratings (e.g., rating '1' and '2' for many datasets, but also rating '5' for MovieLens) only take a small portion of the overall ratings, and most ratings concentrate around the global mean (what we refer to as the *head* of the rating distribution).

Figure 3.1 (b) shows the distributions of predicted ratings by Matrix Factorization (blue) and Neural Collaborative Filtering (yellow). Predicted ratings by MF are normally distributed with a mean which is close to the global means in the training dataset. Regardless of the number of tail ratings in the ground truth, very few ratings are predicted around the tails. Similarly, results using the NCF are slightly better with respect to the data distribution, and the mean is slightly off-centered of the global mean. However, similar to what we observed for MF, the predicted ratings by NCF are mostly concentrated around the head of the distribution, with very few ratings predicted around the tails. These observations show how tail ratings are more likely to be under-served by traditional

	Controversial Items	Polarized Ratings	Rating Percentage
Amazon Books	128	1,393	0.810%
Amazon Digital Music	11	67	0.104%
Amazon Kindle	233	1,589	0.162%
Amazon CD & Vinyl	362	4,650	0.424%
GoodReads	54	752	0.385%
MovieLens	38	99	0.098%

Table (3.2) Ratings of controversial items in six datasets.*

methods that rely on a single latent representation. Due to the high global mean, low tail ratings are most likely over-estimated by such current methods.

Figure 3.1 (c) demonstrates the prediction errors (RMSE) for the ground truth ratings in our test set. As we can see, the predictions are extremely poor for tail ratings for both MF and NCF. For instance, in the Amazon Books dataset, all ratings of '1', which are far from the global mean 4.09, have much worse prediction error (by both MF and NCF) than ratings '3', '4', and '5', which are closer to the global mean. Similar situations are evident for the other datasets as well. Since SLR-based models primarily under-serve these tail ratings, our goal in the following section is to improve this estimation by relaxing the single latent representation assumption.

Observations: Polarized Ratings. We further focus on an extreme case of tail ratings: polarized ratings. Polarized ratings can indicate controversial items; a recommender that mistakenly estimates a high rating for what a user would perceive as a low rating (or vice versa) can be a serious error particularly in domains like politics [98, 99]. Following previous work [100], we adopt a variance threshold, $VAR(R_i) >= 3$, to identify items with polarized ratings. We also ensure that the items have at least a minimum number of ratings, $|R_i| >= 5$, leading to the smaller dataset in Table 3.2.

Focusing on two of the datasets in Figure 3.2, we see the original polarized distribution (green bars) of books in the test set. These distributions are bi-modal, with peaks near to the lowest rating (1) and the highest rating (5). As before, we estimate the ratings of the test set using the standard latent factor model in Equation 2.2 (ignoring NCF for now; the results are similar). The yellow bars

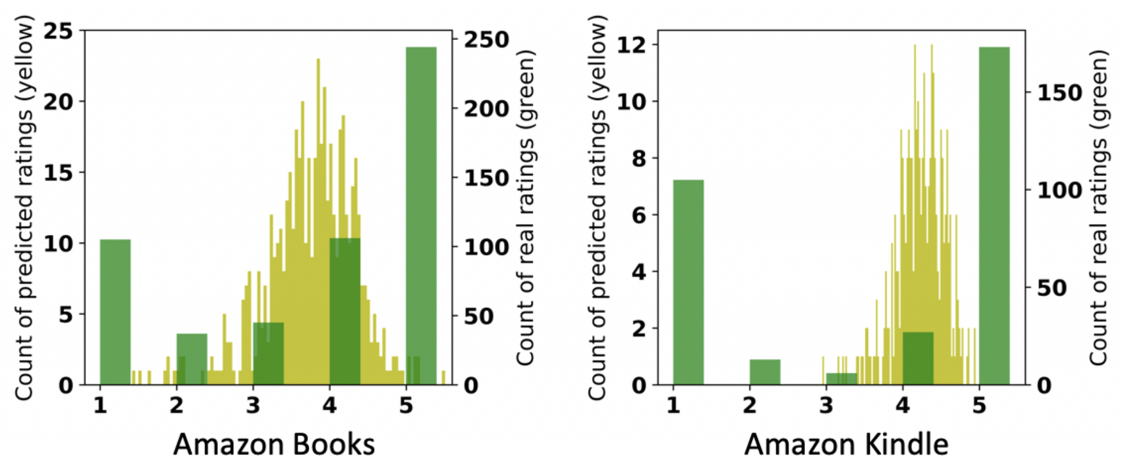


Figure (3.2) Rating distribution for controversial items in Amazon Books and Kindle. The actual ratings (green) for these items fit a U-shape distribution. However, the predicted ratings (yellow) for these items fit a uni-modal distribution, leading to high prediction errors.*

in Figure 3.2 show the predicted ratings for these polarized books in the test set. As expected, the predicted ratings fit the uni-modal distribution and are quite distant from the original ground truth ratings. Most of the ratings on the lower end have been over-predicted into the range near to the global mean.

3.1.2 Limitations of SLR Methods

This data-driven investigation has shown that latent factor models with a single latent representation assumption perform poorly on estimating tail ratings, and especially so for items with polarized ratings. But what is the underlying cause of these errors?

Loss Function Assumes Uni-Modal Data. From a probabilistic perspective, the prediction model found by latent factor models like the ones underlying Matrix Factorization and Neural Collaborative Filtering encourages the predicted value $F(x|\theta)$ to be close to the truth value y, where x is the given feature, and θ denotes the parameters used in the prediction function. These models typically use an L2 norm loss function, which is defined as:

$$\mathcal{L}_{SLR} = \sum_{i=1}^{N} \|y - F(x|\theta)\|_F^2 \tag{3.1}$$

Recall that the task of the model is to find the optimal parameter set θ using the following function:

$$\hat{\theta} = arg \min_{\theta} \mathcal{L}_{SLR} \tag{3.2}$$

Inside the loss function, $\|y - F(x|\theta)\|_F^2$ is the L2 norm between ground truth y and its predicted value $F(x|\theta)$. Similarly, the widely used evaluation metric, Mean Square Error (MSE), is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\|y - F(x|\theta)\|_F^2 = \frac{1}{N}\mathcal{L}_{SLR} \tag{3.3}$$

Next, we consider a Gaussian distribution, defined as:

$$p(z|\mu, \sigma^2) = \exp\left(-\frac{\|\mu - z\|^2}{2\sigma^2}\right) \tag{3.4}$$

where μ is the mean and σ^2 is the variance. If we let $z = F(x|\theta)$, where $F(x|\theta)$ is the predicted value for y, let $\sigma^2 = 1$, and let $\mu = \bar{y}$ ($\bar{y}$ is the mean of data sample), then further apply the log to both sides, we arrive at:

$$\log p(F(x|\theta)|\mu, \sigma^2) \propto -\frac{1}{2}\|F(x|\theta) - \bar{y}\|_F^2 \tag{3.5}$$

which is exactly the negative of the L2 norm loss. In other words, minimizing the L2 loss (or MSE) is equivalent to maximizing the log-likelihood of a Gaussian. A similar derivation can be used to show that minimizing the L1 loss – as in mean absolute error (MAE) – is equivalent to maximizing the log-likelihood of a Laplacian.

Therefore, these traditional loss functions assume that the values y (the rating data of the original matrix M) come from a *uni-modal distribution*. So the observations in the previous section are driven by this underlying uni-modal distribution assumption. While regularization terms can be added to the loss function to help encourage the predicted values to deviate from a strict Gaussian distribution, it is still fundamentally constrained by this uni-modal distribution assumption. Indeed, any method using the L2 (or L1) norm loss – as in MF and NCF but also others – will face the same limitation. Furthermore, since the loss function forces the predicted ratings to fit a uni-modal distribution, it causes the learned latent factors to also fit a uni-modal distribution as well.

Predictions are Blurry. As we've seen in our previous data-driven investigation, the rating distribution for particular items is not necessarily uni-modal. As a result, SLR-based prediction models can give rise to a "blurry" problem, which occurs when the distribution of data y follows a complex distribution, e.g., a bi-modal distribution as in the polarized rating case for controversial items. In these cases, the distribution of polarized ratings p_y may consist of two Gaussian distributions, d_1 and d_2. But the distribution of the optimized predicted ratings, $p_{F(x|\theta)}$, will only fit a single uni-modal Gaussian $(d_1 + d_2)/2$. In other words, the predicted ratings tend to be blurry, which means they are forced to follow the uni-modal Gaussian, due to the average of d_1 and d_2 [101]. Hence, the predicted ratings, $p_{F(x|\theta)}$, using the single-latent-representation-based models have a uni-modal distribution, even for those items which have complex distributions, e.g., bi-modal distribution, in the training dataset.

In the context of a latent factor model like MF or NCF, we have seen that the tail ratings have worse predicted accuracy compared to ratings near to the global center (recall Figure 3.1). Now we have theoretically analyzed why this occurs since SLR-based models are not distribution sensitive. Regardless of how the ground truth is distributed, most predicted ratings will be in the range near to the center, and the count for each predicted range fits the uni-modal distribution. Most tail ratings will be either over-predicted or under-predicted, depending on the particular global means.

3.2 Our Approach: Multi-Latent Representation Recommender

We have experimentally and theoretically analyzed the phenomenon that ratings which are far from the center have a worse predicted error using SLR-based methods, especially for items with polarized rating distributions. In this section, we aim to overcome the limitation of *single latent representations* by proposing a new method which is aware of multi-modal rating distributions. In essence, we aim to learn *multiple latent representations* for each item and user. Our proposed method – MLR – is a neural method with two main components: a multiple latent representation factorization model (MLR-MF, refer to Section 3.2.1) and a gating mechanism to decide which latent representation is most appropriate (MLP-Gate, refer to Section 3.2.2). Together, the entire model is illustrated in Figure 3.3.

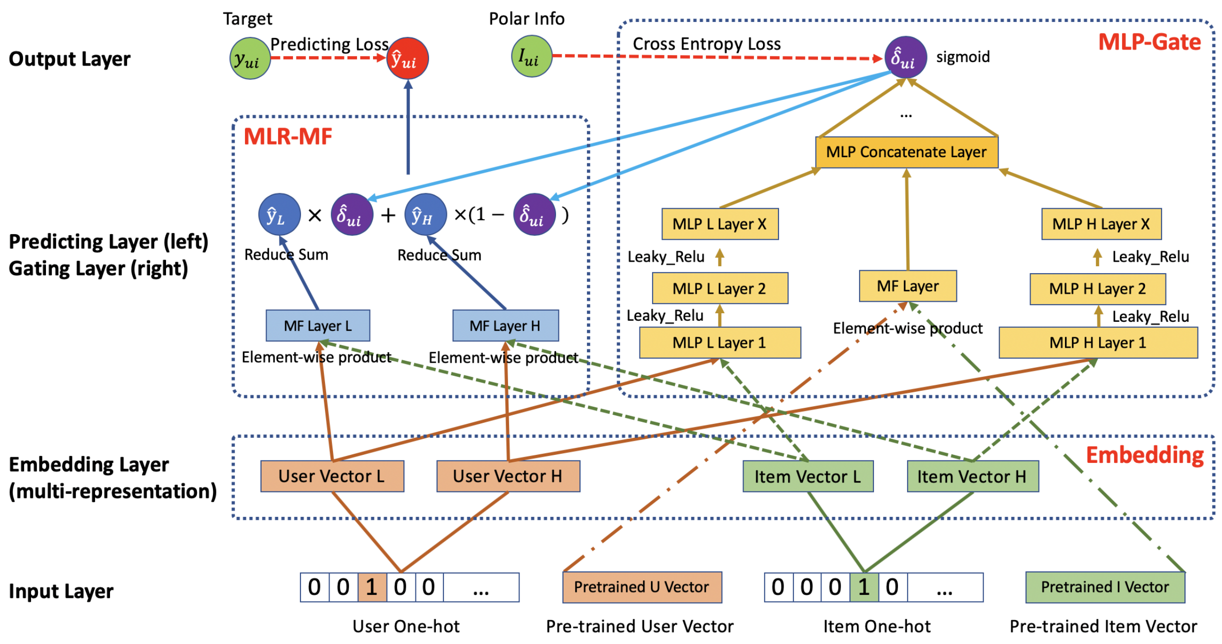


Figure (3.3) Overview of MLR: (1) the bottom box shows the embedding layer, where each item and user is represented as two latent vectors, L and H; (2) the top-left box shows the MLR-MF process to dot product the learned two representations for each side, L and H, respectively; and (3) the top-right box shows the MLP-Gate process which outputs the probability δ_{ui} as the gate to control which pair of representations, $p_L \cdot q_L^T$ or $p_H \cdot q_H^T$, would be used for the final prediction.*

Setup. For a given user-item-rating dataset, our goal is to learn from the training dataset and well predict the missing ratings whose ground truth value is far from the global mean. These tail ratings are typically under-served by traditional methods. Formally speaking, in a training dataset M with potential rating range $[r_{min}, r_{max}]$ and global mean μ, we define the threshold variable $\beta = \min(|\mu - r_{min}|, |r_{max} - \mu|)$. Then we define the tail rating range, $\mathbb{T}$, is $[r_{min}, \mu - \beta)$ if $\beta = |r_{max} - \mu|$, otherwise $\mathbb{T}$ is $(\mu + \beta, r_{max}]$. Our objective is to improve the prediction performance in the tail rating range $\mathbb{T}$, which are primarily under-served by traditional SLR-based models.

3.2.1 MLR Factorization Model (MLR-MF)

The fundamental principle of our approach is to model each user and item with *multiple* representations. For ease of presentation, we focus on items with bi-modal distributions, so we have two “avatars”: I_{Low} and I_{High}, and each one has its latent vector as its representation. The threshold for splitting could be set as the global mean or the expectation value of all ratings for

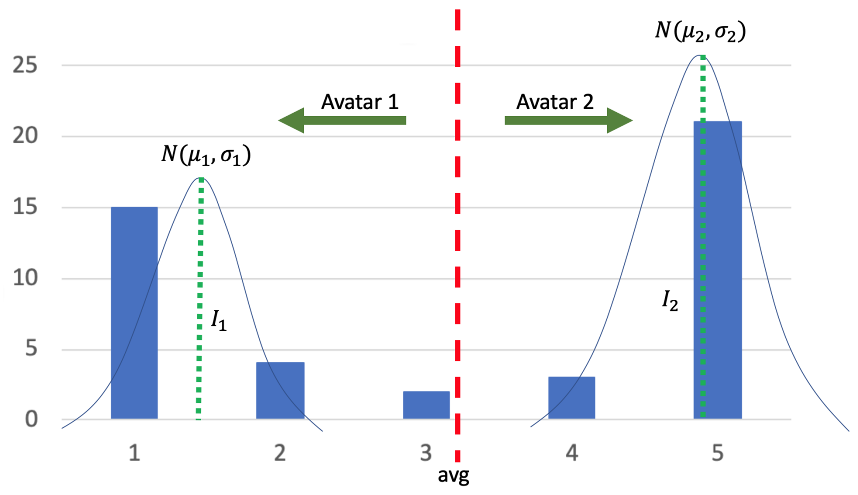


Figure (3.4) An example showing how to split ratings, using two "avatars" per item. In this case, two Gaussian distributions, $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$, capture the bi-modal distribution for this item, where μ_1 is close to the mean of the lower ratings, whereas μ_2 is close to the mean of the higher ratings.*

each item (or from each user) in the training dataset. Figure 3.4 shows the schematic of this idea. Specifically, rather than assuming the distribution of this item's ratings fits a uni-modal distribution with $\mu \approx avg$, we can instead take advantage of a mixture of Gaussian distributions $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$, which would more accurately reflect the distribution of the original ratings. The absolute value of two peaks, l_1 and l_2, can systematically adjust the importance of the two representations to describe one item. That is, when an item has ratings with a uni-modal distribution (as in most cases), the splitting method would lead to $l_1 \ll l_2$ such that $N(\mu_1, \sigma_1)$ could have little influence for the majority of ratings but would still be helpful for tail ratings.

Of course, this bi-modal approach can be extended to consider 3, 4, or more "avatars", leading to a mixture of multiple Gaussians. In practice, however, we do find advantages to using two representations, rather than more. One of the drawbacks of the splitting process is the split matrix could be sparser than the original one; thus, there would be a sharp decline of the learning performance for each split matrix. Another reason to limit the number of mixtures is to relax the strain on the gating process (introduced in the following section) to decide which avatar is the best representation. Besides, since all our datasets have a quite small rating window (e.g., 1 to 5), few cases have visible multi-modal distribution.

In ideal conditions, the original matrix M should be automatically split to matrix M_{Low} and matrix M_{High}, where each contains only higher (or lower) ratings, with a given binary label, I, for distinguishing whether a given pair (u, i) of the original M should belong to M_{Low} or M_{High}. Next, by applying a standard SLR-based method on both new matrices, we could learn two user latent vectors P_{low} and P_{high}, and two item latent vectors Q_{Low} and Q_{High}, respectively. The ideal predicted rating, $\widetilde{r}$, for a pair (u, i) could be calculated by:

$$\widetilde{r_{ui}} = I_{ui} \times (p_{uL} \cdot q_{iL}^T) + (1 - I_{ui}) \times (p_{uH} \cdot q_{iH}^T) \tag{3.6}$$

meaning that, ideally, for a given user u and item i, **if we know** I_{ui}, the side (low or high) the predicted rating belongs on, then we could easily choose the $r_{ui} = p_{uL} \cdot q_{iL}^T$ or $r_{ui} = p_{uH} \cdot q_{iH}^T$. Experimental results (see Figure 3.5) show this ideal case can lead to an improvement in 55% of RMSE in average over the traditional latent factor model on all datasets.

3.2.2 Gating Mechanism (MLP-Gate)

However, this information, I, is not known in the test dataset. Therefore, we propose to build a Gated Multi-Layer-Perceptron (MLP-Gate) to learn the gate variable δ and let it control which side (low or high) the predicted ratings should belong to. Similar to Equation 2.2, the final predicted rating matrix, $\hat{R}$, is calculated as follows:

$$\begin{aligned} \hat{R} = &\delta \times (P_L \cdot Q_L^T + b_{IL} + b_{UL} + \mu_L) + \\ &(1 - \delta) \times (P_H \cdot Q_H^T + b_{IH} + b_{UH} + \mu_H) \end{aligned} \tag{3.7}$$

where b_{UL} & b_{UH} are user bias, b_{IL} & b_{IH} are item bias, and μ_L & μ_H are global means for M_{Low} and M_{High}, respectively.

The ground truth label of δ is given when we split the original matrix to two, i.e. I. Here we build a sub-model, using the learned latent representations of users (P_L and P_H) and items (I_L and I_H), to learn the δ for each given user-item pair.

From the bottom of Figure 3.3, in the MLR-MF learning part, each user (or item) has two latent factors, i.e. p_L and p_H (or q_L and q_H for an item). It is intuitive to combine the user feature and item feature in the same side together by concatenating MLP_L_1 and MLP_H_1. A similar design has been widely used in other neural recommenders [43, 102, 103]. For these two concatenated layers, we add the hidden layers and use a traditional Multi-layer-perceptron (MLP) to learn the interaction between user and item vectors in each side. We choose a $Leaky\_Relu$ as the activation function in each hidden layer. The last layer of the MLP process is named MLP_L_X and MLP_H_X, respectively.

Here, we also adopt the pre-trained latent factor, P_{MF} and Q_{MF}, by traditional SLR models. We employ the element-wise product into these two pre-trained latent factors, and concatenate MLP_L_X and MLP_H_X together, to constitute the combined layer ***MLP Concatenate***. Then, we use the $sigmoid$ as the activation function on ***MLP Concatenate*** layer and output the gate probability δ.

3.2.3 Learning Process

To design the loss function for the MLR, we consider two parts to monitor the performance. Firstly, we use I to check the predicted performance for δ, monitored by cross-entropy loss. Secondly, we use the ground truth rating R to check the predicted performance for the final predicted rating $\hat{R}$, by Mean Squared Error (MSE).

Specifically, the loss function $\mathcal{L}_{MLR-MF}$ for this MLR-MF sub-model is defined below:

$$\begin{aligned}\mathcal{L}_{MLR-MF} =& \|R - \hat{R}\|_F^2 + \lambda_1 \times (\|U\|_F^2 + \|I_L\|_F^2 + \|I_H\|_F^2 \\ & + \|b_u\|_F^2 + \|b_{iL}\|_F^2 + \|b_{iH}\|_F^2)\end{aligned} \tag{3.8}$$

where:

- R: ground truth rating matrix;
- $\hat{R}$: predicted rating matrix by Equation 3.7;
- λ_1: hyper-parameter controlling regularization terms for MF.

The loss function $\mathcal{L}_{MLR-MF}$ is identical to the traditional SLR-based methods. To avoid the blurry problem, we consider adding the loss for MLP-Gate together. For the part of MLP-Gate, we choose binary cross-entropy as the loss function. The loss function $\mathcal{L}_{MLR-MLP}$ is defined below:

$$\begin{aligned}\mathcal{L}_{MLR-MLP} = &- (I \times \log(\delta) + (1 - I) \times \log(1 - \delta)) \\ &+ \lambda_2 \times (\|W\|_F^2 + \|b\|_F^2)\end{aligned} \tag{3.9}$$

where:

- I: ground truth label of input pair (u, i);

- δ: output value of MLP-Gate, the predicted probability of class of input pair (u, i);

- λ_2: hyper-parameter controlling the regularization terms;

- W: All weights in MLP-Gate;

- b: All bias in MLP-Gate.

Up to now, since we learn the MLR-MF and MLP-Gate simultaneously, we can combine $\mathcal{L}_{MLR-MF}$ and $\mathcal{L}_{MLR-MLP}$ as the final loss, defined as follows:

$$\mathcal{L}_{MLR} = \mathcal{L}_{MLR-MF} + \alpha \times \mathcal{L}_{MLR-MLP} \tag{3.10}$$

where α is used for adjusting the contribution of the loss of MLP-Gate to the total loss. As we demonstrated in Section 3.1.2, the L_2 norm in $\mathcal{L}_{MLR-MF}$ forces the generated ratings to be uni-modal, however, $\alpha \times \mathcal{L}_{MLR-MLP}$ could help the model to learn which pair of the latent factors, $p_L \cdot q_L^T$ or $p_H \cdot q_H^T$ could be finally adopted to calculate the predicted value, then adjust the predicted ratings to escape the uni-modal distribution.

In the learning process, we adopt the Adaptive Moment Estimation (Adam) [104] method as the optimizer to train both MLR-MF and MLP-Gate, since it yields faster convergence for both sub-models compared to SGD.

3.3 Experiments and Results

In this section, we evaluate the proposed MLR model over the six datasets listed in Table 3.1. We consider four scenarios: (i) an ideal case to measure the ceiling potential improvement of MLR versus SLR-based methods; (ii) a comparative study versus eight benchmark methods for all ratings (both head and tail ratings); (iii) a focused comparative study on tail ratings only; and (iv) a case

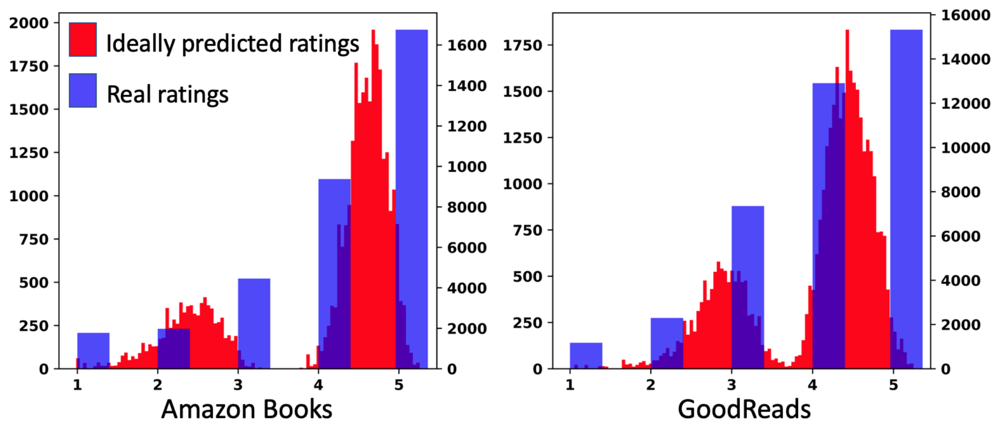


Figure (3.5) Ideal predicted ratings using two latent representations for both users and items, adopting the ground truth I_{ui} instead of the learned probability δ_{ui} for final prediction.*

study on items with extreme polarized ratings, as a special case of tail ratings. We randomly split the ratings of each user into training, validation, and test sets. 5-fold cross-validation is used in all experiments. All experimental results shown below are evaluated on the test dataset.

3.3.1 Ideal Results

First, we evaluate the quality of the multiple-latent-representation underlying the MLR approach in an idealized scenario. That is, we assume we have access to the ground truth label, I, from our validation dataset that determines whether a given pair (u, i) of the original M should belong to M_{Low} or M_{High}. In practice, of course, this label is unavailable to the model, but will give insights into the ceiling potential of MLR. This scenario corresponds to having a cross-entropy loss of 0, which is unlikely in practice.

Figure 3.5 shows the ideal predicted ratings using the multi-latent-factors representations in two testing datasets as examples. As we can see, the distributions are better fits for the original distribution than traditional methods (recall Figure 3.1), and it can capture the tail ratings far from the global centers. Overall, the RMSE is 0.5147, which is a 49.22% improvement from the best benchmark method (NCF) on the Amazon Books dataset (with an original RMSE = 1.0136). This result encourages us that MLR has a stronger and more robust representative ability than SLR. From the other perspective, the impressive ideal performance indicates that the actual performance of

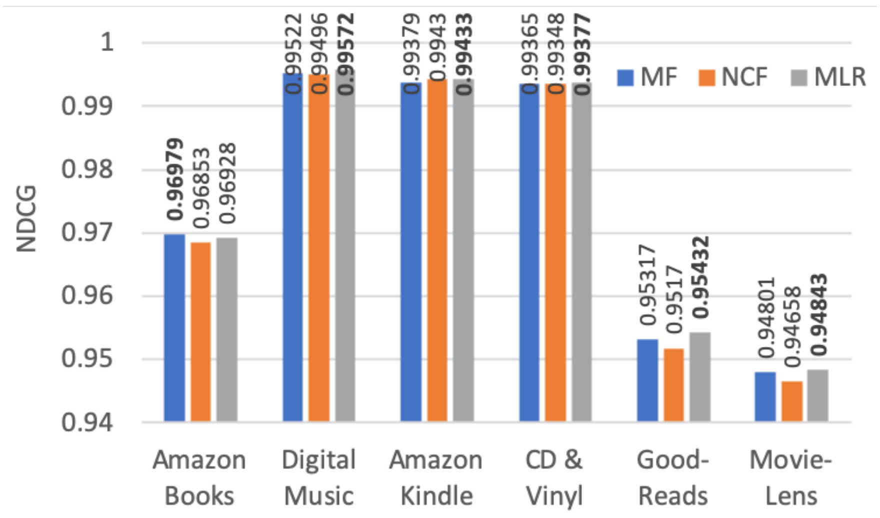


Figure (3.6) NDCG Comparison for **all** ratings on six datasets. MLR is slightly better than MF and NCF on most datasets.*

MLR will strongly depend on the quality of the gating mechanism in the MLP-Gate portion of the approach. Thus, the limitation now becomes how the gate δ controls which latent factors should be used to represent an item or a user.

3.3.2 Prediction for All Ratings

First of all, we evaluate the performance of the MLR model on all datasets comparing with other state-of-the-art methods. This evaluation considers all ratings, so improvements on tail ratings may be overshadowed by predictions on a large number of head ratings closer to the mean rating.

For comparison, we choose Normal Predictor (NP) that guesses a random rating based on the distribution of the training data, Co-clustering (CoC) [105], KNN, SlopeOne [106], SVD, SVD++ [42], Neural Collaborative Filtering (NCF), and Matrix Factorization (MF) with bias as benchmark methods. For our method, we tune the hyper-parameters in terms of number of hidden layers, number of neurons of each layer, dimensions of a representative factor, activation functions, and so on.

We first focus on the comparison of ranking quality. NDCG is the most popular measure for evaluating the ranking quality in recommender systems [107]. Figure 3.6 shows the ranking quality comparison for all ratings on six datasets. We observed that the ranking performances are very close among MLR, MF, and NCF, where MLR is slightly better than MF and NCF on most datasets except Amazon Books. This result indicates that all these three methods perform well at ranking the recommended items to users. Again, our objective is to improve the explicit estimation of ratings. When the ranking qualities are similar, how good is MLR on accurately predicting user-item ratings?

	NP	CoC	KNN	SlopeOne	SVD	SVD++	MF	NCF	MLR
Amazon Books	1.4779	1.0487	1.0424	1.0910	1.0787	1.0639	1.0318	1.0136	**0.9893**
Digital Music	1.4115	1.0016	0.9936	1.0585	0.9429	0.9418	0.9292	0.9369	**0.9039**
Amazon Kindle	1.2265	0.8702	0.8728	0.9106	0.8253	0.8125	0.8084	**0.7874**	0.7879
CD & Vinyl	1.3743	1.0142	1.0079	1.0659	0.9810	0.9743	0.9681	0.9597	**0.9355**
GoodReads	1.3723	0.9734	0.9732	0.9616	0.9469	0.9424	0.9386	0.9353	**0.9070**
MovieLens	1.4915	0.9864	0.9345	0.9578	0.9397	0.9355	0.9264	0.9346	**0.9155**

Table (3.3) Comparing MLR versus eight benchmark methods for all ratings (including both tail and head ratings). Overall, MLR shows a slight improvement in most cases, but with greater gains specifically among tail ratings (see Table 3.4).*

Table 3.3 shows the overall results for all datasets using benchmark models and MLR. As we can see, MLR results in the best RMSE for five of the datasets, with a small loss to NCF on one dataset. Overall, the improvement is fairly small for all ratings, with a maximum of 3.36% improvement versus MF and 3.51% improvement for NCF. Since the non-tail ratings dominate in the aggregate, the overall improvements are small, indicating that MLR at least does not degrade rating prediction performance relative to baselines.

3.3.3 Prediction for Tail Ratings

Hence, we next focus solely on the improvement to tail ratings. Table 3.4 shows a detailed comparison between the proposed MLR method versus NCF and MF. For each dataset, we break the predictions into buckets according to the predicted rating by MF – so there are predicted ratings from 1-2, 2-3, and so on. The first three columns show the RMSE for MF, NCF, and MLR. The columns $\Uparrow_{MLR-MF}$ and $\Uparrow_{MLR-NCF}$ show the prediction improvements for MLR versus MF and

	Predicted Range	MF (RMSE)	NCF (RMSE)	MLR (RMSE)	$\Uparrow_{\text{MLR-MF}}$ (%)	$\Uparrow_{\text{MLR-NCF}}$ (%)	Covered $\mathbb{T}$ (%)	$\mathbb{T}$ $\Uparrow_{\text{MLR-MF}}$ (%)	$\mathbb{T}$ $\Uparrow_{\text{MLR-NCF}}$ (%)
Amazon Books	[1,2)	0.8717	1.0625	0.3366	61.38	68.32	0.01	61.38	68.32
	[2,3)	1.5132	1.4256	1.1990	20.76	15.89	1.66	54.76	49.65
	[3,4)	1.3294	1.2752	1.2428	6.51	2.54	70.44	23.46	15.51
	[4,5)	0.8595	0.8659	0.8480	1.33	2.06	27.77	0.57	-2.59
	>5	0.1818	0.2216	0.1418	22.00	36.0	-	-	-
	All	1.0318	1.0136	0.9893	**4.11**	**2.39**	100	**15.34**	**8.74**
MovieLens	[1,2)	1.0927	1.1561	0.9936	9.07	14.05	3.29	17.52	26.39
	[2,3)	1.0372	1.0532	1.0135	2.28	3.76	50.18	8.96	15.53
	[3,4)	0.9053	0.9083	0.8996	0.63	0.95	45.74	0.74	-0.03
	[4,5)	0.7612	0.7907	0.7515	1.28	4.95	0.79	2.73	2.54
	>5	0.8928	0.9737	0.7830	12.29	19.58	-	-	-
	All	0.9264	0.9346	0.9155	**1.17**	**2.04**	100	**3.04**	**6.16**
Amazon Kindle	[1,2)	1.6149	1.3267	1.1774	27.08	11.25	0.49	86.31	25.83
	[2,3)	1.3148	1.200	1.1962	9.01	0.34	8.81	18.36	28.14
	[3,4)	1.1055	1.068	1.0734	2.91	-0.50	59.52	13.60	11.54
	[4,5)	0.7116	0.7029	0.7024	1.29	0.06	30.70	3.57	1.69
	>5	0.4368	0.3764	0.3598	17.61	4.40	0.47	18.33	6.00
	All	0.8084	0.7874	0.7879	**2.53**	-0.01	100	**8.93**	**7.43**
CD & Vinyl	[1,2)	0.7149	1.3280	0.6660	6.84	49.85	0.03	47.70	45.29
	[2,3)	1.5414	1.6074	1.1281	26.81	29.81	2.34	48.05	50.76
	[3,4)	1.3702	1.3464	1.2804	6.54	4.90	47.14	22.12	19.78
	[4,5)	0.8755	0.8708	0.8597	1.80	1.27	50.46	5.22	2.88
	>5	0.2337	0.2592	0.2166	7.31	16.42	0.01	3.74	2.10
	All	0.9681	0.9597	0.9355	**3.36**	**2.52**	100	**11.79**	**9.50**
GoodReads	[1,2)	-	-	-	-	-	-	-	-
	[2,3)	1.4382	1.4377	1.2134	15.62	15.59	1.90	39.62	39.35
	[3,4)	1.0570	1.0503	1.0253	3.00	2.38	82.10	10.81	8.96
	[4,5)	0.7992	0.8043	0.7729	3.90	3.30	15.99	-2.98	-4.33
	>5	0.3079	0.2911	0.3098	0.61	6.42	-	-	-
	All	0.9368	0.9353	0.9070	**3.18**	**3.02**	100	**7.88**	**6.14**
Digital Music	[1,2)	0.7059	1.7798	0.7468	-5.8	58.03	0.46	-5.8	58.03
	[2,3)	1.4220	1.5920	1.2855	9.59	19.25	10.14	31.42	44.18
	[3,4)	1.2610	1.2709	1.2134	3.77	4.52	56.68	17.19	14.88
	[4,5)	0.7974	0.7981	0.7845	1.62	1.70	32.71	1.73	1.19
	>5	0.2712	0.2915	0.2766	-1.99	5.11	-	-	-
	All	0.9292	0.9369	0.9039	**2.72**	**3.51**	100	**9.40**	**11.51**

Table (3.4) Comparing MLR with single latent factor models MF and NCF for different rating ranges as predicted by MF. For ratings predicted by MF and NCF, those farther from the global mean, the improvement (in the columns $\Uparrow_{\text{MLR-MF}}$ and $\Uparrow_{\text{MLR-NCF}}$) by MLR is more pronounced, especially for tail ratings (in the columns $\mathbb{T}$ $\Uparrow_{\text{MLR-MF}}$ and $\mathbb{T}$ $\Uparrow_{\text{MLR-NCF}}$). '-' indicates there is no predicted data in the current range by MF.*

NCF for each bucket of ratings, respectively. The column Covered $\mathbb{T}$ shows the **tail ratings** that are covered by the current rating bucket. Finally, the columns $\mathbb{T} \Uparrow_{MLR-MF}$ and $\mathbb{T} \Uparrow_{MLR-NCF}$ show the improvement for MLR versus MF and NCF for each bucket of tail ratings.

We observe that for ratings far from the center, e.g., ratings less than 3, the overall improvement by MLR is substantial. For example, for Amazon Books with ratings from 1-2, MLR results in a 60%+ improvement versus both MF and NCF. For ratings from 2-3, MLR results in a 15%+ improvement versus both alternatives. Of course, these tail ratings cover a small portion of all ratings, but the improvements are large. And even for buckets with high coverage, there are still improvements (e.g., 2.54% and 6.51% for Amazon Books for rating bucket 3-4). Considering MovieLens, the tail ratings occur on both sides of the mean rating: we see large improvements for the low rating bucket 1-2 (e.g., 9% and 14% versus MF and NCF) and for the high ratings bucket 4-5 (e.g., 1.28% and 4.95% versus MF and NCF).

In some cases, there are some decreases in the predicted tail rating range near to the global mean in some datasets, e.g., a decrease of 2.59% on Amazon Books in the predicted range 4 to 5, and a decrease of 0.03% on MovieLens data in the predicted range 3 to 4, compared with NCF. In these ranges near the center, the proposed MLR approach may perform worse than traditional methods since these methods predict most ratings in this range, but MLR does not. As a result, the gating mechanism that guides δ may lead to mispredictions.

For a more detailed look, Figure 3.7 shows the distributions of predicted ratings by MLR versus MF, as well as the ground truth ratings for Amazon Digital Music and Kindle datasets. As we can see, rather than predicting most ratings near to the global mean, the MLR approach better fits the original distribution of the ground truth data, and overcomes the uni-modal distribution limitation. Overall, we observe that our prediction model performs better on the tail ratings.

3.3.4 Prediction for Controversial Items

Finally, we consider a special case of tail ratings for controversial items with low polarized ratings. Specifically, we consider ratings of 1 or 2, which are over-estimated by traditional methods. Therefore, we apply the MLR model on the controversial items in three of the datasets with

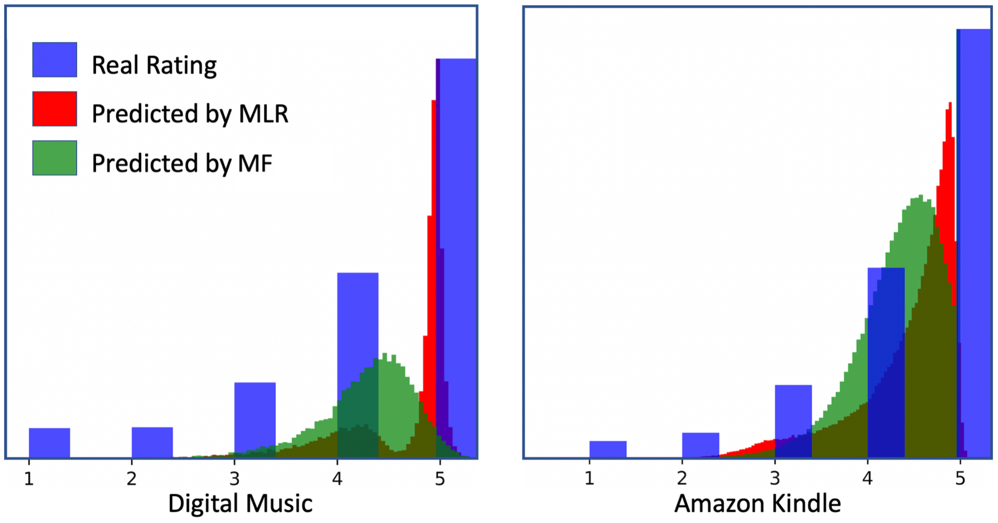


Figure (3.7) Distribution of real (blue) and predicted (green by MF and red by MLR) ratings for two datasets. Compared with the predicted ratings by MF (green), the predicted ratings by MLR (red) fit more accurately to the actual ratings.*

	Amazon Books	Amazon Kindle	Amazon CD & Vinyl
MF (RMSE)	1.9711	1.7630	1.8816
NCF (RMSE)	1.8923	1.6524	1.7934
MLR (RMSE)	1.6355	1.4479	1.5903
$\Uparrow_{MLR-MF}$	**17.02%**	**17.87%**	**15.48%**
$\Uparrow_{MLR-NCF}$	**13.57%**	**12.38%**	**11.33%**

Table (3.5) Comparing MLR versus baselines for controversial items.*

sufficient items. Table 3.5 shows the predicted performance on the low-polarized ratings (1 or 2) of the controversial items. The results are strong, where MLR improves the prediction of polarized ratings by 17.02%, 17.87%, and 15.48% on Amazon Books, Amazon Kindle, and Amazon CD & Vinyl dataset, respectively, compared with the MF method.

3.4 Summary

In this chapter, we studied the estimation bias for items with different ratings. We conducted a data-driven investigation and theoretical analysis of the challenges posed by traditional latent factor models for estimating tail ratings. These approaches assume a single latent representation, which can lead to over- and under-estimations of tail ratings, with particularly pronounced errors on controversial items. With these challenges in mind, we proposed a new multi-latent representation method designed specifically to estimate these tail ratings better. Experimental results show the estimation improvement is especially strong for those items far from the ratings mean. Furthermore, the proposed model is generalizable and can be easily extended to take advantage of other SLR-based models.

4. ADDRESSING THE TARGET CUSTOMER DISTORTION PROBLEM IN RECOMMENDER SYSTEMS[†]

In essence, the worse estimation of tail ratings is due to the unequal distribution of all ratings. Our proposed approach MLR introduced in last chapter improves the estimation of tail ratings in explicit recommender systems. However, there is still bias and distortion caused by the inequitable demographic distribution in implicit recommendation methods. Next, we analyze the distribution bias of predicted target customers.

Predicting the potential target customers for a product is essential. Accurate analysis and prediction of the population distribution of target customers for an item (e.g., a product or an advertisement) could directly improve the item's business prospects. One key factor in targeting is the *class distribution* of target customers. For example, an advertising campaign may wish to guarantee at least a certain demographic (e.g., 18-35) sees its ads. Or a job posting may wish to guarantee that an equal number of women and men are targeted.

This focus on the overall distribution and composition of an item's target customers is often in opposition to the criteria driving *personalized* recommenders, which typically aim to optimize an engagement metric without considering the overall distribution of target customers. For example, recommendation algorithms such as Probabilistic Matrix Factorization (PMF) [108], Bayesian Personalized Ranking (BPR) [109], and Neural Collaborative Filtering (NCF) [43], are designed to optimize for metrics based on user-item interactions like $mean average precision$, $NDCG$, $precision@k$, or $recall@k$. However, these approaches typically do not consider the distribution of target customers.

To illustrate, consider a media-service provider that recommends movies to their users. From the perspective of each movie, what customers will be considered first as the targets to receive the recommendation? One intuitive answer is that the target customers should be the ones who are most

[†]Reprinted with permission from "Addressing the Target Customer Distortion Problem in Recommender Systems" by Xing Zhao, Ziwei Zhu, Majid Alfifi, and James Caverlee, 2020. Proceedings of The Web Conference 2020. Copyright 2020 by ACM. DOI: https://doi.org/10.1145/3366423.3380065

potentially interested in this item. There are many approaches to recommend items to users based on past user-item interactions. Suppose at the end, based on the predicted recommendation results, we have a target customer set that receives the recommendation of item i. We use q as the class (such as gender, age, or occupation) distribution of target customers for item i. By relying on a recommender that does not consider the distribution of target customers, consider the following distortions that may arise (and that we verify do arise in our data-driven analysis in this chapter):

Distortion 1: the target distribution q may be dominated by the overall class distribution. For example, suppose the male-female ratio of users in the entire customer base is 70%:30%. For each movie, the set of recommended users always contains more males than females, even though some movies may be individually preferred by females.

Distortion 2: the target distribution q for minority classes may be under-recommended for majority-preferred items. For example, suppose movie i is a male-preferred movie, and the male-female ratio of users who have already watched movie i is 90%:10%. There will be fewer females (even much less than 10%) in i's predicted target customers.

Distortion 3: the target distribution q may unexpectedly differ from the appropriate distribution. For example, if i is an R-rated movie, the predicted target customer set may include a great number of children using conventional recommenders.

In this chapter, we focus on these types of *target customer distortion problems*, where the distribution and composition of target customers differ much from the desired ones, that arise in recommenders. Recent research has examined related issues in the distribution challenge from the user's perspective through *calibrated recommendations*, to ensure that users are exposed to a diverse recommended list of items [18]. However, there is a gap in viewing this distribution challenge from the item's perspective. That is, what is the distribution of customers that are targeted for each item?

Concretely, we first introduce an approach to identify the target customers for each item (Section 4.1), which is challenging since recommenders are typically structured to reveal what each user prefers (rather than what users are targeted by each item). We conduct a data-driven study in Section 4.2 to reveal several distortions that arise from conventional recommenders. Toward overcoming these issues, in Section 4.3, we propose a target customer re-ranking algorithm – U2I-Calibration – to adjust the population distribution and composition in the Top-k target customers of an item while maintaining recommendation quality. Next, in Section 4.4, we apply this proposed algorithm onto the MovieLens 1M dataset [96], and evaluate the distribution of users and the quality of recommendation, then discuss its merits and drawbacks. Last but not least, we summarize this work in Section 4.5.

4.1 Preliminary

To address the target customer distortion problem, we first need to obtain the target customers of an item predicted by a conventional recommender system. With these target customers, we could then identify the distortions that occur.

Given a user set $\mathcal{U}$, an item set $\mathcal{I}$, and a binary *user-item interaction matrix* $\mathbf{H} \in \mathbb{N}^{|\mathcal{U}|\times|\mathcal{I}|}$ (where $\mathbf{H}_{u,i} = 1$ indicates that user $u \in \mathcal{U}$ has watched movie $i \in \mathcal{I}$ for example), traditional recommenders output predicted recommendation results which we represent as a *score matrix* $\mathbf{D} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$. Each value $\mathbf{D}_{i,j}$ expresses the predicted score from user $u \in \mathcal{U}$ to item $i \in \mathcal{I}$.

To obtain the Top-k recommended items for user u, we can return the first k items with the largest predicted score in row $\mathbf{D}_{u,\_}$, calculated as follows ($\searrow$ symbolizes descending sort):

$$top^{\mathcal{U}}(u,k) = \underset{\searrow,k}{\arg\operatorname{sort}}\langle \mathbf{D}_{u,1}, \mathbf{D}_{u,2}, ..., \mathbf{D}_{u,|I|}, \rangle \tag{4.1}$$

A straightforward idea is using the same way to obtain the Top-k predicted target customers of an item, directly from the score matrix $\mathbf{D}$. However, by leveraging the conventional personalized recommenders, e.g., [43, 48, 109, 110, 111, 112], we cannot directly compare the predicted scores in a column (for one item) in the score matrix, due to the different users' bias. That is, for example, a higher predicted score for user-item pair (u_1, i) does not necessarily indicate that u_1 likes item i more than another user u_2 who has a lower predicted score for i, because u_1 and u_2 may have different scoring scales.

Therefore, to obtain the Top-k predicted target customers of an item, we should first normalize $\mathbf{D}$ column-wise. Instead of using the predicted value, we normalize $\mathbf{D}$ using the rank information from a user to an item. We map each value $\mathbf{D}_{u,i}$ to its **descending rank order** in the row $\mathbf{D}_{u,\_}$, and define a *ranking matrix* $\mathbf{R} \in \mathbb{N}^{|\mathcal{U}| \times |\mathcal{I}|}$ mapping scores to ranks from $\mathbf{D}$. For example, if a row in $\mathbf{D}$ is $\langle 0.9, 0.3, 0.6 \rangle$, the mapped row in $\mathbf{R}$ should be $\langle 1, 3, 2 \rangle$. Values of $\mathbf{R}$ are integers between 1 to $|\mathcal{I}|$.

On the one hand, from the user's perspective, each value of the ranking matrix, $\mathbf{R}_{u,i}$ represents u's preference rank to i among all items. Therefore, the Top-k recommended items for a user u could also be represented as follows ($\nearrow$ symbolizes ascending sort):

$$top^{\mathcal{U}}(u, k) = \underset{\nearrow, k}{\arg\,\mathrm{sort}} \langle \mathbf{R}_{u,1}, \mathbf{R}_{u,2}, ..., \mathbf{R}_{u,|I|} \rangle \tag{4.2}$$

On the other hand, from the item's perspective, each value of the ranking matrix, $\mathbf{R}_{u,i}$, represents the preference degree from user u to this item i among all users. Target customers are the users who are most potentially interested in an item (with highest preference degree). Now, the Top-k target customers for an item i could be represented as:

$$top^{\mathcal{I}}(i, k) = \underset{\nearrow, k}{\arg\,\mathrm{sort}} \langle \mathbf{R}_{1,i}, \mathbf{R}_{2,i}, ..., \mathbf{R}_{|U|,i} \rangle \tag{4.3}$$

Furthermore, we know each row, $\mathbf{R}_{u,\_}$, represents the preference ranking given from a user u to each item $i \in \mathcal{I}$. Suppose values in the originally predicted recommendation matrix $\mathbf{D}$ are different from one another, and the values in $\mathbf{R}_{u,\_}$ must be from 1 to $|\mathcal{I}|$. However, if we vertically observe $\mathbf{R}$, we can find the values in column $\mathbf{R}_{\_,i}$ may be identical to one another, and the value range is **not** necessarily from 1 to user size $|\mathcal{U}|$. For instance, assuming there is a trendy item i_{hot} which has been set as many users' first priority, so that the column $\mathbf{R}_{\_,i_{hot}}$ should contain many "1"s and the average (or median) of $\mathbf{R}_{\_,i_{hot}}$ could be much less than a column for an unpopular item. Keeping this observation in mind, we will focus not only on all item's class distribution but also on popular items.

4.2 A Data-driven Study of Target Customer Distortion

In this section, we focus on three examples from a real-world dataset, and observe the distortion between the desired class distribution (denoted as p) and the class distribution of predicted Top-k target customers (denoted as q) for an item.

4.2.1 Dataset

We adopt the MovieLens 1M dataset [96], which contains 1 million user-movie interactions collected from 6,040 users ($\mathcal{U}$) and 3,706 movies ($\mathcal{I}$). We only consider user-item interactions rather than the explicit ratings, i.e. all interacted user-movie pairs will be considered as 1. In addition, for each user, this dataset contains user profile information, including age, gender, and occupation, which could help us to analyze the class distribution of the audience. We now start from gender and age as the concerned demographic classes in the beginning to identify the three distortions discussed before.

For clarity in presentation, we adopt Bayesian Personalized Ranking (BPR) [109], one of the most influential and foundational personalized recommender algorithms. Experiments with other tested algorithms show similar results; our emphasis here is on the general problem of target customer distortion that can manifest in recommenders that optimize for engagement metrics without consideration of the overall item target distribution. We apply BPR and obtain the predicted user-item interaction matrix $\mathbf{D}$. Through the transformations introduced in Section 4.1, we find the predicted target customers for each item. In the following analysis, we would like to observe the class distribution of the Top-k target customers, $top^{\mathcal{I}}(i, k)$, of certain items. Intuitively, we expect the predicted class distribution q of target customers to match the distribution p of existing users in the training set of an item. To this end, we sort all users who interacted with an item by their timestamps. Next, for each item, we select the first 60% interactions as the training set and randomly select half of the rest data as validation set (20%), then, use the remaining 20% as the test dataset.

4.2.2 Examples of Target Customer Distortion

Intuitively, we expect that the desired class distribution p and the predicted class distribution q should be as similar as possible. However, following the three distortions defined before, we find there is a strong disagreement between the two distributions in many cases:

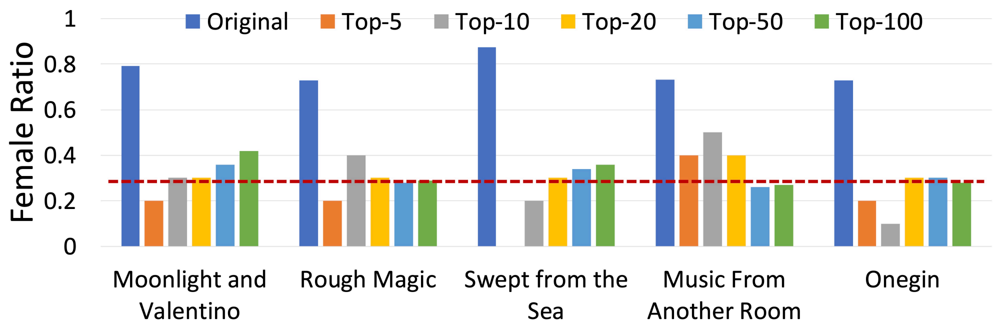


Figure (4.1) The female ratio on some female-preferred movies. The dotted red line is the female ratio of the entire dataset.†

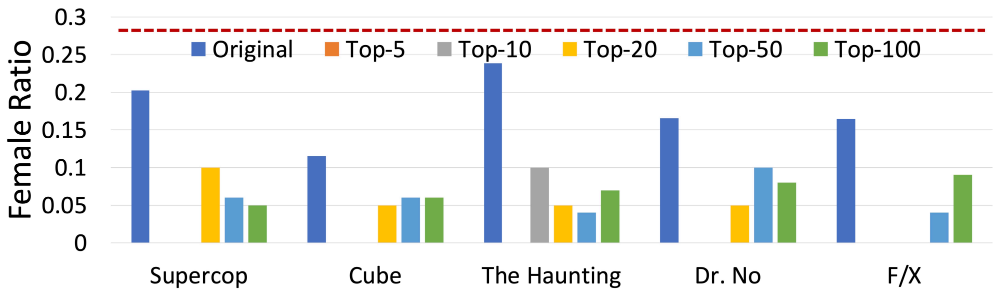


Figure (4.2) The female ratio on some male-preferred movies. The dotted red line is the female ratio of the entire dataset.†

Case 1: Fewer females are the target customers of female-preferred movies.

Figure 4.1 shows the gender distribution of 5 female-preferred movies in their training set (p, blue bar), and corresponding distribution in their Top-k predicted target customer set (q, bars with other colors). We observe that even though in the training set, females have much more interactions than males with these movies, the predicted target customers still contain more males than females, and for all settings of the Top-5, Top-10, Top-20, Top-50, and Top-100 target customers. One of the reasons is due to the overall smaller ratio (29%) of females in the entire training set (refer to the dotted red line). In this example of distortion, the recommender under-serves a large target customer group (in this case, females).

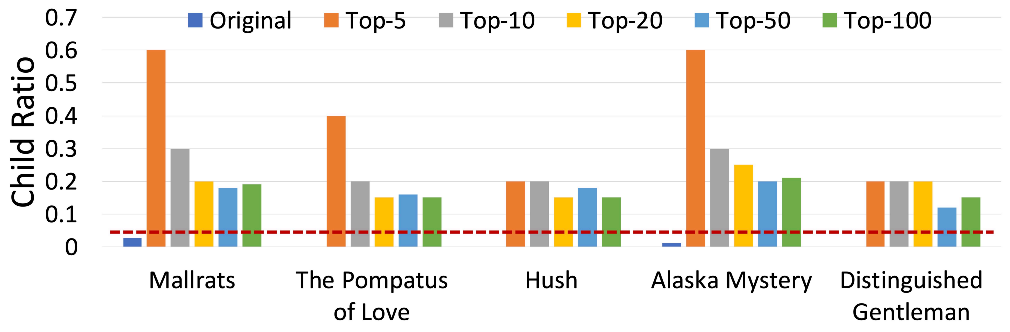


Figure (4.3) The children ratio on some R-rated movies. The dotted red line is the child ratio of the entire dataset.†

Case 2: Few females are the target customers of male-preferred movies.

Figure 4.2 shows the gender distribution of five male-preferred movies in the training set and Top-k target customer set. The historical training data show more males interacted with these movies than females. In this case, it is expected that fewer females may be included in the target customers set. But, this does not mean the recommender should ignore females in their Top-K target customer set. In fact, even though females do watch these movies, however, in the predicted result, there are few females in the Top-5 and Top-10. Although this situation is relaxed when we select more candidates as target customers, the ratio of females is still much lower than the desired one. In this example of distortion, the recommender under-serves a small-size class (in this case, female), or sometimes ignores them completely. A similar phenomenon has also been analyzed in Steck's research from the user perspective [18]: some genres with a small portion will be less recommended to a user.

Case 3: Children are target customers of R-rated movies.

Figure 4.3 shows the ratio of children (age under 18) for five R-rated movies. Although the R-rated movie should not be targeted to children under 18-years-old, we still observe some cases in our training set. As shown in Figure 4.3, the recommender includes a substantial portion of children as target customers of R-rated movies. One of the hypotheses of the phenomena shown in Case 1 and 2 is due to the female ratio (29%) over the entire dataset being lower than males

(71%). However, the phenomenon of predicting more children as target customers of R-rated movies surprisingly violates the aforementioned hypothesis. That is, in the entire dataset the ratio of children (age under 18) is only around 4%, but the predicted ratio of children in target customers of R-rated movies are much more than that. From this case, we can observe that in some cases conventional recommenders may over-serve a tiny class (i.e., children).

4.3 Our Approach: U2I-Calibration

In Section 4.2.2, we have shown the predicted class distribution of target customers (q) strongly disagrees with the expected one (p) recommended by using a conventional recommender. Intuitively, one potential solution for these issues is to re-generate the Top-k predicted target customers set to make the class distribution of target customers for an item fit the desired distribution. In many cases, it may be reasonable to set the desired distribution $p(o|i)$ as the class distribution in the training set. Furthermore, there may be some special cases where we wish to manually control the distribution $p(o|i)$, e.g., setting $p(o = child|i_{R-rated}) = 0$ to limit children from being recommended R-rated movies.

Problem Statement: Given a predicted item-user interaction ranking matrix $\mathbf{R}$ by a conventional recommender, a concerned class o (e.g., gender or age), and the desired class distribution $p(o|i)$, we aim to make the class distribution of predicted target customers, $q(o|i)$, be as similar as $p(o|i)$ through a re-ranking process, while maintaining the original recommendation performance.

As many recommenders are trained in a pairwise manner, many studies state that one might not be able to include calibration into the training [18]. Therefore, a common solution is re-ranking the predicted list in a post-processing step, which has been widely used in calibrated machine learning approaches [113, 114, 18]. In this section, we propose a post-processing approach for target customer re-ranking – U2I-Calibration – to make the class distribution of target customers of each item as close as the desired class distribution.

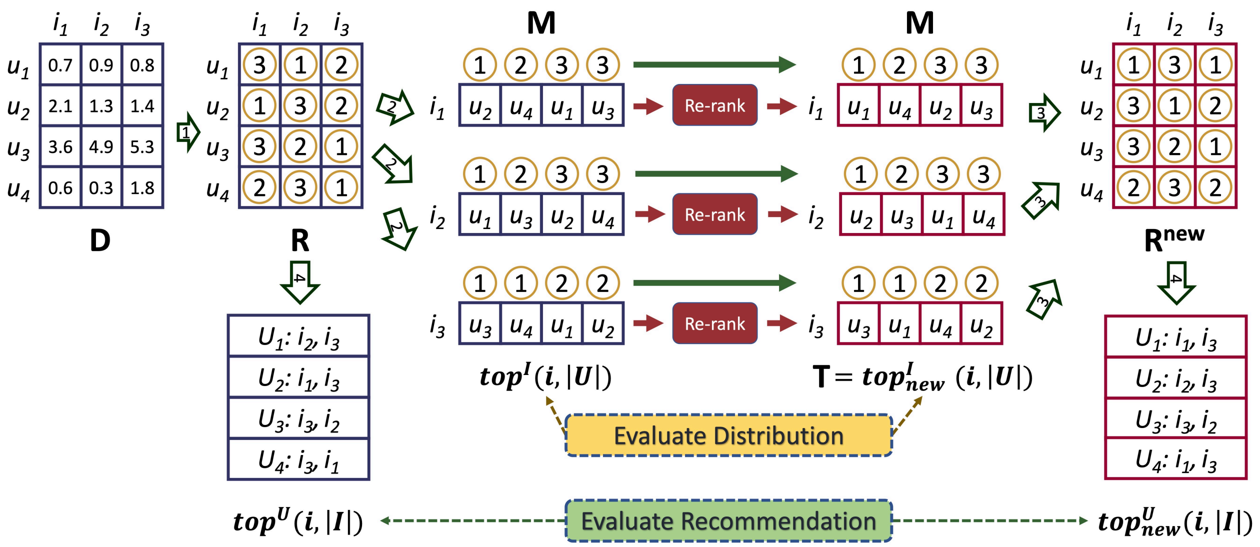


Figure (4.4) U2I-Calibration: a target customer re-ranking algorithm: (1) from *score matrix* $\mathbf{D}$ predicted by a conventional recommender, generate the original *ranking matrix* $\mathbf{R}$; (2) from $\mathbf{R}$ generate the Top-k recommended item set $top^{\mathcal{U}}(u,k)$ for each user for later evaluation; (3) from $\mathbf{R}$ generate the Top-k target customer set $top^{\mathcal{I}}(i,k)$ for each item as well as the *memory matrix* $\mathbf{M}$; (4) re-rank the Top-k target customer set and generate a new *target customer matrix* $\mathbf{T}_i = top^{\mathcal{I}}_{new}(i,|\mathcal{U}|)$ and evaluate the distribution $KL(p||q)$ through comparing $top^{\mathcal{I}}_{new}(i,k)$ and $top^{\mathcal{I}}(i,k)$; (5) from $\mathbf{T}$ and $\mathbf{M}$ generate the *new ranking matrix* $\mathbf{R}^{new}$; (6) from $\mathbf{R}^{new}$ generate the new Top-k recommended item set $top^{\mathcal{U}}_{new}(u,k)$; and (7) evaluate the recommendation by comparing $top^{\mathcal{U}}_{new}(u,k)$ and $top^{\mathcal{U}}(u,k)$.†

4.3.1 Class Distribution

We have introduced how to get the Top-k predicted target customers for each item i, $top^I(i,k)$, from the ranking matrix $\mathbf{R}$ (refer to Section 4.1). From the user set $top^I(i,k)$, we can now analyze the Top-k target customer's class distribution, such as gender and age. Given o as the class of interest, where o could represent gender or age range (or other domain-specific class of interest), we denote for each valid value c for o, the *desired class distribution* $p(o=c|i)$ for item i as:

$$p(o=c|i) = \frac{\sum_{u\in\mathcal{U}} \omega_{u,i} \times p(o=c|u)}{\sum_{u\in\mathcal{U}} \omega_{u,i}} \tag{4.4}$$

where $p(o=c|u)$ and $\omega_{u,i}$ are two binary variables: $p(o=c|u)$ is 1 if u belongs to c, and $\omega_{u,i}$ is 1 if user u watched movie i in the training dataset, respectively. When o represents gender, we assume given a movie i, the probability $p(o=male|i)$ is the ratio of males to all people who watched this movie, and $p(o=female|i)$ is the ratio of females, supposing for simplicity in presentation here that $male$ and $female$ are mutually exclusive. For a given historical matrix $\mathbf{H}$, the desired class distribution $p(o|i)$ could be a fixed number (ratio) based on the historic interaction record. In some special occasions, $p(o|i)$ could also be manually set as a desired number, for example, we could set $p(o = child)|i_{R-rated}) = 0$ to expect that all children (age under 18) should not be recommended R-rated movies.

Similarly, we could calculate the *predicted class distribution*, $q(o|i)$, of predicted Top-k target customers $top^{I}(i, k)$ for item i as follows:

$$q(o|i) = \frac{\sum_{u \in top^{I}(i,k)} p(o|u)}{k}. \tag{4.5}$$

Ideally, we expect the *predicted class distribution* $q(o|i)$ to be as similar as the *desired class distribution* $p(o|i)$. Otherwise, if $q(o|i)$ is quite different from $p(o|i)$, then we will have identified a *target customer distortion*. In most of the cases, we expect the desired distribution $p(o|i)$ is the historical distribution of existing users for an item i, as calculated in Eq. 4.4. We also allow manually setting p in some cases. For example, as in the third case in Section 4.2, some children (age under 18) watched the R-rated in our training dataset so that the distribution of $p(o = child|i) \geq 0$ for R-rated movie i. Even though, we still could manually force $p(o = child|i) = 0$ to avoid recommending R-rated movie i to children by ranking all children in the very back of the potential relative user list of i.

To compare the similarity/distance between two distributions $p(o|i)$ and $q(o|i)$, we use the Kullback-Leibler (KL) divergence [115] as the metric, where $KL(p||q) = 0$ indicates the distributions $p(o|i)$ and $q(o|i)$ are exactly the same; and a larger $KL(p||q)$ indicates they are opposite of each other.

4.3.2 KL-weighted Top-k Target Customers

In Section 4.1, we showed how to get the Top-k predicted target customers $top^{\mathcal{I}}(i, k)$ from the ranking matrix $\mathbf{R}$. To memorize the preference priority from each user $u \in top^{\mathcal{I}}(i, |\mathcal{U}|)$ to item i, we introduce a *memory matrix* $\mathbf{M} \in \mathbb{N}^{|\mathcal{I}| \times |\mathcal{U}|}$:

$$\mathbf{M}_i = \underset{\nearrow}{\mathrm{sort}} \langle \mathbf{R}_{1,i}, \mathbf{R}_{2,i}, ..., \mathbf{R}_{|\mathcal{U}|,i}, \rangle \tag{4.6}$$

recalling that $\mathbf{R}_{(u,i)}$ is the **rank** of item i in user u's priority list.

To re-rank the Top-k most likely target customers and let the class distribution of target customers q to fit our desired distribution p, we leverage *maximum marginal relevance (MMR)*, which can provide precise re-ranking results [116]. We store these re-ranked results into the new *target customers matrix* $\mathbf{T} \in \mathbb{N}^{|\mathcal{U}|\times|\mathcal{I}|}$, so that the new Top-k predicted target customers for item i, i.e., $top^{\mathcal{I}}_{new}(i,k)$, is the first k elements (users) in i^{th} column of $\mathbf{T}$. We could obtain the optimized new target customer set, $\mathbf{T}_i$, for item i as follows:

$$\mathbf{T}_i = \underset{\mathcal{C}_i \subseteq top^{\mathcal{I}}(i,|\mathcal{U}|)}{\arg\max} \; (1-\lambda) \times r(\mathcal{C}_i) - \lambda \times KL(p||q(\mathcal{C}_i)) \tag{4.7}$$

where $\lambda \in [0,1]$ is the trade-off between the original recommendation results and the distribution metric, $\mathcal{C}$ is the current optimal subset of re-ranked target customers, and recommendation score $r(\mathcal{C})$ is calculated from the ranking (priority) of user $u \in \mathcal{C}$ in item i's original target customers list:

$$r(\mathcal{C}_i) = \frac{1}{|\mathcal{C}_i|}\left(\sum_{u\in\mathcal{C}_i} \frac{1}{\mathbf{R}_{u,i}+1}\right) \tag{4.8}$$

Through the re-ranking process, a user's ID can be stored in a column $\mathbf{T}_i$ per step, from top to bottom.

4.3.3 Top-Z Selection Mechanism

To add each user into $\mathbf{T}_i$, a traditional re-ranking method would go through the entire original target customer list, $top^{\mathcal{I}}(i,|\mathcal{U}|)$ with size of $|\mathcal{U}|$), then select the one with the most optimal KL-weighted score. To save running time and maintain prediction quality, instead of going through the entire list of $top^{\mathcal{I}}(i,|\mathcal{U}|)$, we only consider the Top-Z users in $top^{\mathcal{I}}(i,|\mathcal{U}|)$; in our experiments, we set Z as 30 times the number of valid values for o (e.g., $Z = 60$ for $o \in Gender$).

The benefits of only selecting from Top-Z users in the *current* $top^{\mathcal{I}}(i,Z)$ rather than the entire user set are not only significantly speeding processing time ($Z \ll |\mathcal{U}|$), but also further ensuring recommendation quality. That is, we need not engage our re-ranking algorithm to choose the user in the bottom of $top^{\mathcal{I}}(i,|\mathcal{U}|)$, although it may slightly improve $KL(p||q)$.

4.3.4 Rebuild the Rank Matrix for Users

In Section 4.1, we introduced how to transform $\mathbf{D} \rightarrow \mathbf{R} \rightarrow top^{\mathcal{I}}(i,k)$. As every step is a linear transformation, the entire process can be reversed. That means from the re-ranked target customers matrix $\mathbf{T}$ where $\mathbf{T}_i = top^{\mathcal{I}}_{new}(i, |\mathcal{U}|)$, we could reverse the process through $\mathbf{T} \rightarrow \mathbf{R}^{new}$ and generate the new version of the ranking matrix $\mathbf{R}^{new}$. Specifically, leveraging the re-ranked target customer matrix $\mathbf{T}$ and the original memory matrix $\mathbf{M}$, we could build the new ranking matrix $\mathbf{R}^{new}$ by:

$$\mathbf{R}^{new}_{u,i} = \mathbf{M}_{v(\text{where } \mathbf{T}_{i,v}=u),i} \tag{4.9}$$

In this way, the new Top-K recommended items for a user u could be easily calculated by:

$$top^{\mathcal{U}}_{new}(u,k) = \underset{\nearrow,k}{\arg\operatorname{sort}} \langle \mathbf{R}^{new}_{u,1}, \mathbf{R}^{new}_{u,2}, ..., \mathbf{R}^{new}_{u,|I|} \rangle \tag{4.10}$$

Also, to check the recommendation performance of the re-ranked matrix $\mathbf{R}^{new}$, we leverage the widely-used evaluation metric, $F-1@K$. Through comparing with the original Top-k recommended items to a user (in $\mathbf{R}$) and the new Top-k recommended items for the same user (in $\mathbf{R}^{new}$) after using the proposed target customer re-ranking algorithm, we can measure the impact on the recommendation results after considering the class distribution of target customers. To illustrate the workflow of the proposed re-ranking algorithm, Figure 4.4 walks step-by-step of our proposed U2I-Calibration through a simple example.

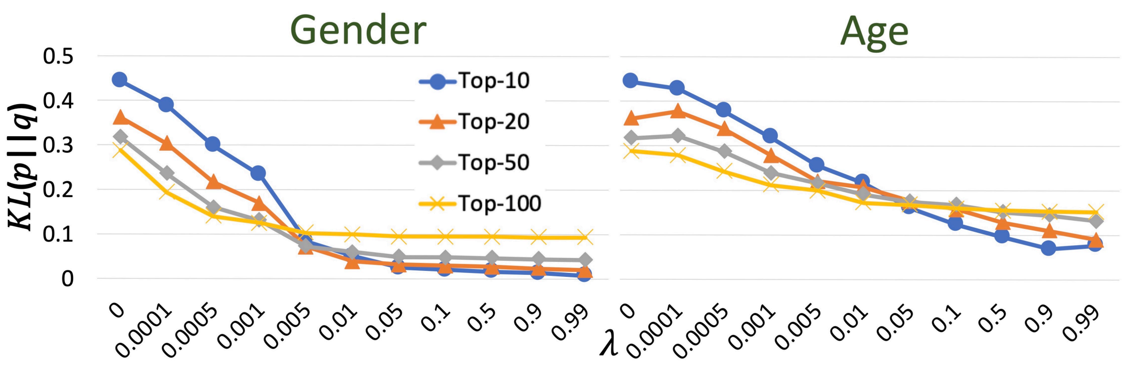
4.4 Experiments and Results

In the previous sections, we have identified how the class distribution of target customers of an item (q) and it's desired distribution (p) can be distorted. To address this problem, we proposed a target customer re-ranking algorithm – U2I-Calibration. In this section, we apply U2I-Calibration onto the MovieLens dataset introduced in Section 4.2, and evaluate the results from both perspectives of distribution bias and recommendation accuracy.

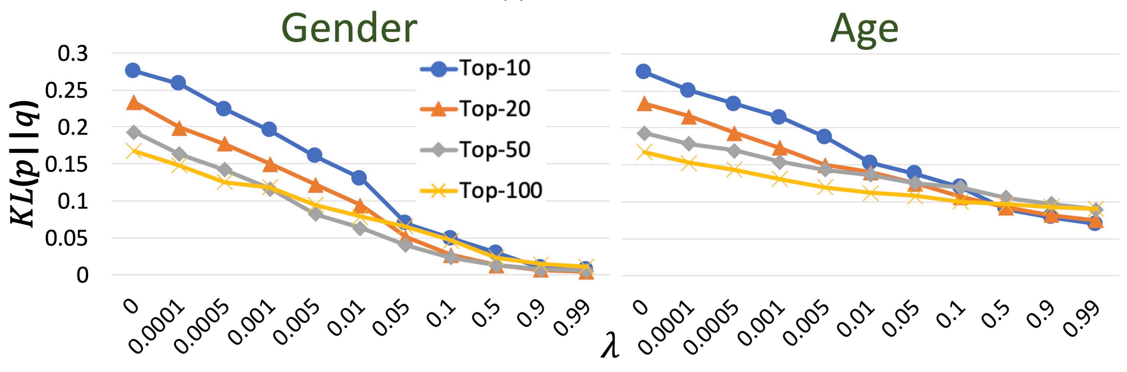
To match the result analysis in Section 4.2, we use BPR as the base of our target customer re-ranking algorithm. It is important to note that the proposed algorithm is a post-processing solution which could be applied upon any conventional recommenders. Here we use BPR as a representative.

We first apply BPR onto our training dataset, and through the transformation introduced in Section 4.1 we obtain the Top-k predicted target customers $top^{\mathcal{I}}(i,k)$ for each user i. Through applying the proposed re-ranking approach onto $top^{\mathcal{I}}(i,k)$ we now have the re-ranked new Top-k predicted target customers $top^{\mathcal{I}}_{new}(i,k)$ for i. Furthermore, we can also obtain users' Top-k recommended items before and after applying the re-ranking approach, i.e. $top^{\mathcal{U}}(u,k)$ and $top^{\mathcal{U}}_{new}(u,k)$, respectively.

4.4.1 Bias of Class Distributions of Target Customers



(a) All items



(b) Popular items

Figure (4.5) $KL(p||q)$ of class distributions of target customers after applying U2I-Calibration.†

First of all, we compare the desired class distribution p with $top^{\mathcal{I}}(i,k)$ and $top^{\mathcal{I}}_{new}(i,k)$, respectively. Figure 4.5a shows the score of the distribution metric $KL(p||q)$ with different settings of λ, in two cases, i.e., $o \in Gender$ and $o \in Age$, for all items. In both cases, we observe that $KL(p||q)$ decreases with the increase of λ. The difference among these two cases are: comparing with the

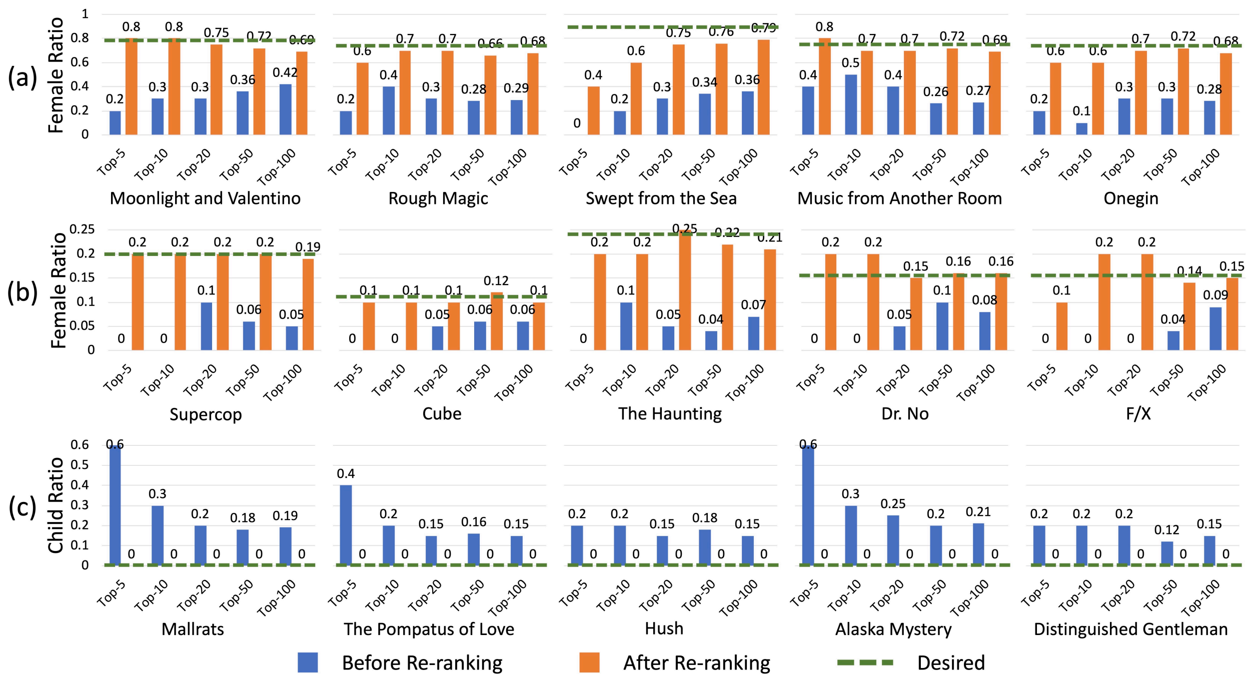


Figure (4.6) The adjusted ratios of (a) females on sample female-preferred movies, (b) females on sample male-preferred movies, and (c) children on sample R-rated movies before and after applying U2I-Calibration, in the setting of $\lambda = 0.5$. The dashed lines are the corresponding desired p.[†]

case of $o \in Gender$, $KL(p||q)$ is harder to converge in the case of $o \in Age$, and the $KL(p||q)$ is still far from 0 when we set the largest λ in our experiment. This is because we choose the optimal user u in the Top-Z candidates of original $top^{\mathcal{I}}(i, k)$ each interaction, instead of the entire user set. Within an extreme condition, Top-Z candidates do not contain an optimal choice to improve the current $KL(p||q)$. Such a phenomenon becomes even more evident when the class contains more valid values, i.e., there are 7 valid values for *Age* and 2 valid values for *Gender*.

Recalling our Top-Z candidates mechanism, in the case of the small size of class (e.g., *Gender*), the Top-Z candidates mechanism performs well due to the fast processing speed, high recommendation accuracy, as well as almost unharmed $KL(p||q)$ value. However, as we can see, with the growing size of class, the effect caused by this selection mechanism to $KL(p||q)$ will be more obvious.

We also observe that, $KL(p||q)$ with a smaller k drops more quickly and converges more slowly with the growth of λ, than $KL(p||q)$ with a larger k. This is due to the original ratio of one class in the entire user set. Given a user set contains 10 females and 90 males and desired distribution $\frac{f}{m} = \frac{1}{2}$, the Top-10 target customers with the expected distribution should obviously be easier to satisfy than the Top-100 target customers.

Figure 4.5b shows KL scores of *popular items*. Here, we define the popular items as items in users' Top-k preference list. Comparing with the class distribution of all items (refer to Figure 4.5a), similar downtrends are observed: $KL(p||q)$ will decrease with the growth of λ. However, the $KL(p||q)$ is always lower for popular items than the one for all items, especially when λ is quite small. This observation indicates that a traditional recommender brings more bias of class distributions for unpopular items.

Furthermore, Figure 4.6 shows the re-ranked Top-k target customers using our re-ranking algorithm of those examples we showed in Section 4.2, in the setting of $\lambda = 0.5$. It is not surprising that the re-ranked Top-k target customers fit the desired distribution. A reasonable portion of females are included in to corresponding movie's target customer set (refer to Figure 4.6a and 4.6b). And no child will be set as target customers of R-rated movie by setting the desired distribution $p(o = children|i_{R-rated}) = 0$ (refer to Figure 4.6c).

4.4.2 Influence of Recommendation Accuracy Cased by Re-ranking

There is an inherent trade-off between a reasonable class distribution and an accurate recommendation. We already showed good results of the class distribution of target customers for an item using the proposed U2I-Calibration algorithm. Next, we show how recommendation accuracy is affected.

Figure 4.7 shows the F-1 score of user-viewed Top-k recommendations after applying U2I-Calibration optimized for gender and age class distribution, respectively. As we can see, in the case of optimizing gender distribution, with the growth of λ, the recommendation accuracy is almost unaffected. Taking the benefits of class with fewer valid values, the re-ranking is extremely slight, with little impact on F-1. Surprisingly, even in the case of class with more valid values, e.g., *Age*, the recommendation accuracy only drops sightly by taking the benefit of the Top-Z selection mechanism. From an item-viewed perspective, some "ranking metrics" may be affected,

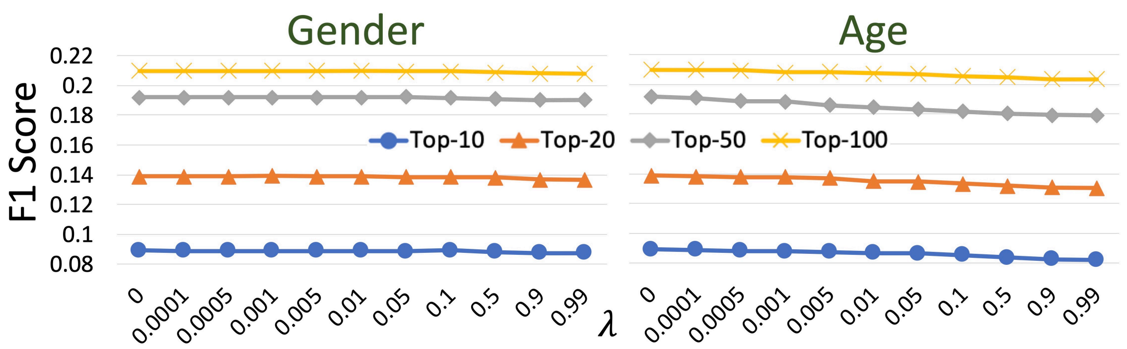


Figure (4.7) F-1 Score only mildly affected after applying U2I-Calibration.†

e.g., NDCG. However, NDCG is not always the appropriate ranking metric for target customer prediction, because target customers are usually considered as a group, e.g., group ads injection [117].

4.5 Summary

In this chapter, we focus on the distribution bias of predicted target customers. We first conducted a data-driven study in to reveal several distortions that arise from conventional recommenders. Then, we proposed a target customer re-ranking algorithm – U2I-Calibration – for addressing the target customer distortion problem. Experimental results suggested our proposed method can effectively adjust the class distribution of the target customers for items toward a desired distribution, thereby mitigating the distortion problem.

5. RABBIT HOLES AND TASTE DISTORTION: DISTRIBUTION-AWARE RECOMMENDATION WITH EVOLVING INTERESTS [‡]

In the previous two chapters, we proposed our methods to mitigate the estimation bias for items with different ratings and the distribution bias of predicted target customers. Specifically, Chapter 4 focused on the distribution bias from a user's perspective, and proposed U2I-Calibration to address the target customer distortion problem. In this chapter, we turn to the item's perspective and introduce the next contribution of this dissertation which addresses the item's categorical distribution bias of the predicted user's taste.

One long-standing challenge in recommender systems is the *rabbit hole effect* [118, 119, 120] – a representative instance of an item's categorical distribution bias: as the system evolves, the recommended results may concentrate on the main areas of interest of a user, while the user's lesser areas of interest can be underrepresented or even absent. By narrowing down a user's interest areas and limiting exploration of new areas, this rabbit hole effect can lead to undesirable (and often unforeseen) outcomes, raising concerns of echo chambers [18], fairness [19, 20, 121, 122], and diversity [22, 23]. In one extreme direction, O'Callaghan *et al.* observed that users accessing extreme right video content are likely to be recommended further extreme right content, leading to immersion in an ideological bubble in just a few clicks [120].

To address this rabbit hole effect, there are two main research directions: (i) diversity-focused recommendation, e.g., [82, 83, 84, 22, 85, 86]; and (ii) distribution-aware recommendation, e.g., [18, 77, 123]. *Diversity-focused recommendations* aim to introduce the diversification of users' interests in the recommendations. Some diversity-focused approaches may use the category of an item or the genre of a movie as an "interest area" to cover more diverse aspects [87, 88]. Other methods eschew such explicit aspects in favor of identifying latent aspects as the basis of diversification [89]. In another direction, *distribution-aware recommendation* aims to ensure

[‡]Reprinted with permission from "Rabbit Holes and Taste Distortion: Distribution-Aware Recommendation with Evolving Interests" by Xing Zhao, Ziwei Zhu, and James Caverlee, 2021. Proceedings of the Web Conference 2021. Copyright 2021 by ACM. DOI: https://doi.org/10.1145/3442381.3450099

that a user's prior *taste distribution* (i.e., the distribution of interest areas) are reflected in the recommendations that a system makes, so that the system neither over-emphasizes main interest areas, nor under-serves lesser areas. For example, consider a user who historically prefers comedies to dramas by 2:1. A distribution-aware recommender would aim to make recommendations follow a similar ratio, whereas a conventional recommender might incrementally focus on comedies to the detriment of dramas. Such an approach has proven to be an important building block for recommendation tasks [18, 77, 123], and has attracted considerable attention in the machine learning community [124, 55].

Although both diversity-focused recommenders and distribution-aware recommenders may help mitigate the *rabbit hole effect*, they further introduce a new ***taste distortion problem*** – another representative instance of item's categorical distribution bias in the recommendations: *the taste distribution in recommendation results differs from a user's actual tastes.* Diversity-focused recommenders aim to cover as many interest areas as possible in recommendations rather than targeting a user's future taste distribution, resulting in the taste distortion problem. Distribution-aware recommenders enforce that a user's taste distribution exactly matches the user's prior ones based on the assumption that this distribution is fundamentally static (i.e., without considering the dynamic changes and shifts of user's interests), leading to the taste distortion issue as well.

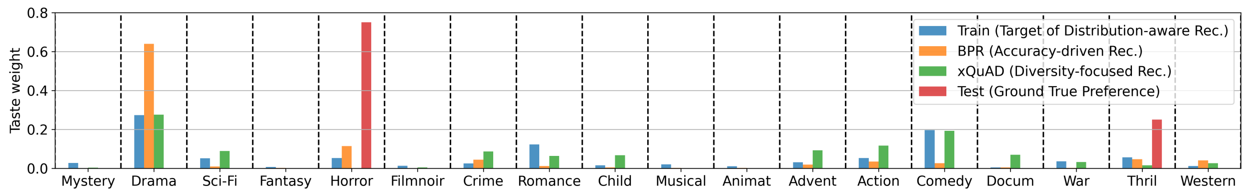


Figure (5.1) A sample user's taste distribution on the training set (blue, being the target of the distribution-aware recommendation), BPR predicted results (orange), xQuAD predicted results (green), and testing set (red, being the ground truth). Results suggest that none of these recommenders can provide the taste distribution in recommendations close to the ground truth.‡

To illustrate this taste distortion problem, in Figure 5.1, we show a random user in the MovieLens-

1M dataset, where her view history is split into training and testing in chronological order. The x-axis in Figure 5.1 lists the genres, and the y-axis is the ratio of the current genre in this user's entire taste distribution (hence, the sum is 1). Blue bars show the taste distribution of this user in the training data. As we can see, this user shows great interest in drama movies much more than others in the training set. The recommendation result from a traditional recommender (BPR [109] in orange) hints at the problem of the rabbit hole effect, as its recommendations skew towards drama even more than what is in the training distribution. A diversity-focused method (xQuAD [125] in green) brings more diverse recommendations comparing with the traditional recommender, while a distribution-aware recommender aims to maintain the training distribution (in blue). **However, none of these recommendation results are close to the user's real taste distribution in the next stage**, which is shown by the red bars and illustrates this user's strong interests in the horror and thriller genres.

Hence, our objective is to simultaneously mitigate both the *rabbit hole effect* and this *taste distortion* problem. But, how can we achieve this objective even in the presence of dynamically shifting tastes? And how can we integrate methods to address these problems into existing recommenders (including both traditional and recent neural-based ones) without re-training the original recommendation pipeline? Since distribution-aware recommenders do not only address the *rabbit hole effect*, but also offer benefits of interpretability (by respecting a user's existing preferences) and provide possibilities to control the diversification to avoid undesirable recommendation, we focus in this chapter on new distribution-aware methods that can also counter the *taste distortion problem*. Concretely, we propose a new taste-enhanced calibrated recommender called TecRec that predicts a user's shift in tastes, and then incorporate these shifts into a post-ranking framework for an improved distribution-aware recommendation.

Furthermore, many studies have shown that distribution adjustment and accurate recommendation tend to trade-off with each other [78, 18]. That is, conventional distribution-aware recommendations will lead to worse prediction accuracy. However, considering users' dynamic taste shifting enables our proposed method to provide a better estimation of a user's real taste distribution. Thus, one additional benefit of our proposed recommender is the potential of even better recommendations while mitigating the rabbit hole effect and the taste distortion problem.

In sum, this chapter studies the potential of a calibrated recommendation in the presence of dynamic tastes:

- First, we empirically reveal the taste distortion problem through a data-driven study over multiple datasets, to show how taste preferences dynamically shift and how a calibration mechanism should be designed with these shifts in mind.
- Then, we propose a Taste-Enhanced Calibrated Recommender that is designed to firstly predict a user's shift in preferences, and then incorporate these shifts into a post-ranking framework for an improved distribution-aware recommendation.
- Finally, we compare the proposed method against traditional recommenders, sequential recommenders, diversity-focused recommenders, and conventional distribution-aware recommenders. We show how the taste-enhanced calibration is complementary to these approaches' goals and can result in a high-quality recommendation with that mitigates the "rabbit hole effect" and "taste distortion" while evolving with a user's dynamically shifting tastes.

5.1 Preliminaries

Suppose we have user set $\mathcal{U}$, an item set $\mathcal{I}$, and a binary *user-item interaction matrix* $\mathbf{H} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$ (where $\mathbf{H}_{u,i} = 1$ indicates that user $u \in \mathcal{U}$ has interacted with item $i \in \mathcal{I}$ for example). $\mathbf{H}$ is split into t$\underline{\text{R}}$ain ($\mathbf{H}^R \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$), $\underline{\text{V}}$alidation ($\mathbf{H}^V \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$), and $\underline{\text{T}}$est ($\mathbf{H}^T \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$) sets by a time-series order. Furthermore, we have explicit labels of interest on the items (e.g., genre, category, etc.), which we refer to as a genre set $\mathcal{G}$. As one item may belong to more than one genre, each item i has a genre vector $\mathbf{c}_i$:

$$\mathbf{c}_i = \left[\frac{v_{i,g_1}}{\Sigma}, \frac{v_{i,g_2}}{\Sigma}, \cdots, \frac{v_{i,g_{|\mathcal{G}|}}}{\Sigma}\right] \tag{5.1}$$

where $g_j \in \mathcal{G}$, $\Sigma = \sum\limits_{g_j \in \mathcal{G}} v_{i,g_j}$, and v_{i,g_j} is a binary variable where $v_{i,g_j} = 1$ if i belongs to genre g_j.

Taste Distribution on Training and Testing Sets. In the following, we denote the ground truth taste distribution for a user as **q** and the taste distribution in the training set as **p**.

We define u's taste ratio for genre g_j in the training set as $\mathbf{p}_{u,g_i}$:

$$\mathbf{p}_{u,g_j} = \frac{\sum_{i \in \mathcal{I}} \mathbf{H}^R_{u,i} \times \mathbf{c}_{i,g_j}}{\sum_{i \in \mathcal{I}} \mathbf{H}^R_{u,i}} \tag{5.2}$$

where $\mathbf{H}^R_{u,i}$ is the entry in u^{th} row and i^{th} column of $\mathbf{H}^R$, and $\mathbf{H}^R_{u,i}$ is 1 if user u interacted with item i in the training dataset. Thus, user u's entire taste distribution is:

$$\mathbf{p}_u = \left[\mathbf{p}_{u,g_1}, \mathbf{p}_{u,g_2}, \cdots, \mathbf{p}_{u,g_{|\mathcal{G}|}}\right]. \tag{5.3}$$

Similarly, we can compute user u's taste distribution for genre g_j in the test set as $\mathbf{q}_{u,g_j}$ and the user's entire taste distribution $\mathbf{q}_u$. For a given historical matrix $\mathbf{H}$, user u's taste distribution in the training set $\mathbf{p}_{u,g_i}$ and in the testing set $\mathbf{q}_{u,g_j}$ could be a fixed ratio based on the historic interaction record.

Taste Distribution on Top-K Predicted Results. We represent a recommender's predicted results as a *score matrix* $\mathbf{D} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$. Each value $\mathbf{D}_{u,i}$ expresses the predicted score from user $u \in \mathcal{U}$ to item $i \in \mathcal{I}$. To obtain the Top-k recommended items for user u, we can return the first k items with the largest predicted score in row $\mathbf{D}_{u,:}$, calculated as follows ($\searrow$ symbolizes descending sort):

$$top(u,k) = \underset{\searrow,k}{\arg\operatorname{sort}} \left[\mathbf{D}_{u,1}, \mathbf{D}_{u,2}, \cdots, \mathbf{D}_{u,|I|}\right] \tag{5.4}$$

Similarly, we calculate the *predicted taste distribution*, $\widetilde{\mathbf{q}}_u^{\text{Top}-k}$, of predicted Top-*k* recommended items $top(u,k)$ for user u as follows:

$$\widetilde{\mathbf{q}}_u^{\text{Top}-k} = \frac{1}{k} \sum\nolimits_{i \in top(u,k)} \mathbf{c}_i. \tag{5.5}$$

5.2 A Data-driven Study of Taste Distortion

The key idea of distribution-aware recommenders is to incorporate a user's taste distribution into the recommendation results to potentially benefit recommendations that fit a "desired" taste distribution. However, a strong assumption that has been widely used is the target distribution should fit the distribution in the training set (i.e., the prior data), while ignoring the dynamic shifting

	#User	#Item	#Genre	#Interaction	Density(%)
ML-1M	6,040	3,706	18	1,000,209	4.468
ML-20M	138,493	26744	20	20,000,263	0.540
MovieTweets	60,283	34,437	29	814,504	0.039

Table (5.1) Dataset statistics for taste distortion analysis.[‡]

nature of taste preferences. In this section, we conduct a data-driven study into the resulting *taste distortion problem*. Concretely, we define the taste distortion problem as follows:

Taste distortion problem. Consider a user u with a time-series of interactions with several items. The taste distribution in the prior (i.e., training data) is $\mathbf{p}_u$, the true distribution in the future (i.e., testing data) is $\mathbf{q}_u$, and the taste distribution of Top-k recommended items for u by a recommender system is $\widetilde{\mathbf{q}}_u^{\mathrm{Top}-k}$. We define the *taste distortion* as $\varphi(\widetilde{\mathbf{q}}_u^{\mathrm{Top}-k}, \mathbf{q}_u)$, where $\varphi(\cdot)$ denotes a distance measure, e.g., Kullback-Leibler (KL) divergence [115]. The larger this value is, the worse the estimation of the taste distribution on the test set is.

5.2.1 Datasets and Setup

We adopt three datasets that contain clearly defined genres, which help identify a taste distribution, and timestamps for considering the temporal shifts in these taste distributions. The first dataset is the MovieLens-1M (*abbr.* ML-1M) dataset [96], which contains 1 million user-movie interactions collected from 6,040 users and 3,706 movies. The second is the larger MovieLens-20M (*abbr.* ML-20M) dataset [96], which contains 20 million user-movie interactions collected from 138,493 users and 26,744 movies. The third is the MovieTweets dataset [126], which contains 814,504 user-movie interactions collected from 60,283 users and 34,435 movies. Due to the sparsity of MovieTweets dataset, we only consider users who have five or more interactions. More details are shown in Table 5.1.

To analyze a user's tastes across different genres, we consider user-item interactions (e.g., clicks, plays) rather than explicit ratings, i.e., all interacted user-movie pairs are considered as 1. For each movie, we use the genre available in each dataset: ML-1M contains 18 genres, ML-20M contains 20 genres, and MovieTweets contains 29 genres. One movie may belong to one or several genres. This genre information lets us build each user's explicit taste preference distribution. We sort each user's interaction history in chronological order and split them into Training (60%), Validation (20%), and Testing (20%) sets. By splitting in chronological order, we emulate what information is actually available to a recommendation system (rather than considering a random split of the data that will mix past and future interactions).

In the following, we study the taste distortion problem in the context of a standard recommender, Bayesian Personalized Ranking (BPR) [109], and the conventional distribution-aware recommender (CaliRec) [18]. Experiments with other tested algorithms show similar results; our emphasis here is on the general problem of taste distortion that can manifest in recommenders.

5.2.2 Taste Distortion in Recommendation

We begin with a look at the taste distortion problem through analysis over all three datasets. To compare the distortion between two taste distributions **a** and **b**, we use the Kullback-Leibler (KL) divergence [115] as the distance measure $\varphi(\cdot)$, where $KL(\mathbf{a}||\mathbf{b}) = 0$ indicates the distributions $\mathbf{a}_{g_j}$ and $\mathbf{b}_{g_j}$ are exactly the same for all $g_j \in \mathcal{G}$; and a larger $KL(\mathbf{a}||\mathbf{b})$ indicates they are more distant. $KL(\cdot)$ is defined as below:

$$KL(\mathbf{a}||\mathbf{b}) = -\sum_{g_j \in \mathcal{G}} \mathbf{a}_{g_j} \log \frac{\mathbf{b}_{g_j}}{\mathbf{a}_{g_j}}. \tag{5.6}$$

Figure 5.2 shows the KL-divergence for the taste distribution on the training dataset (**p**) with the ground truth taste distribution on the testing dataset (**q**), CaliRec results ($\widetilde{\mathbf{q}}_{\text{BPR+CaliRec}}^{\text{Top}-100}$), and BPR results ($\widetilde{\mathbf{q}}_{\text{BPR}}^{\text{Top}-100}$). We set the trade-off parameter $\lambda = 0.9$ and the basis recommendation result *base = BPR* for CaliRec. As we can see, the taste distributions on the testing set are far from the ones on the training set. Encouragingly, CaliRec does significantly improve on BPR, since it is focused on matching the distribution in the training data. However, the training data is not reflective of the distribution in the testing data, and so this approach may bring worse accuracy in the recommendation.

For a comparison from another perspective, Figure 5.3 shows the taste distribution on the testing

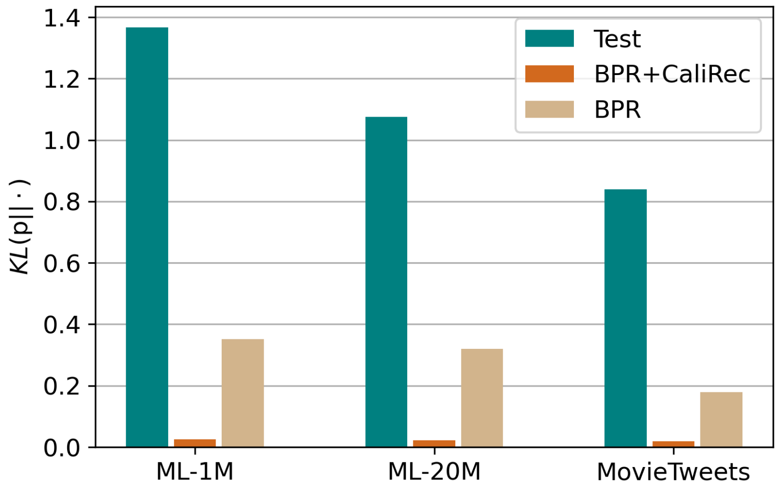


Figure (5.2) KL-Divergence between the taste distribution in training set (**p**) with the ones in testing set (**q**), in CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and in BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$).‡

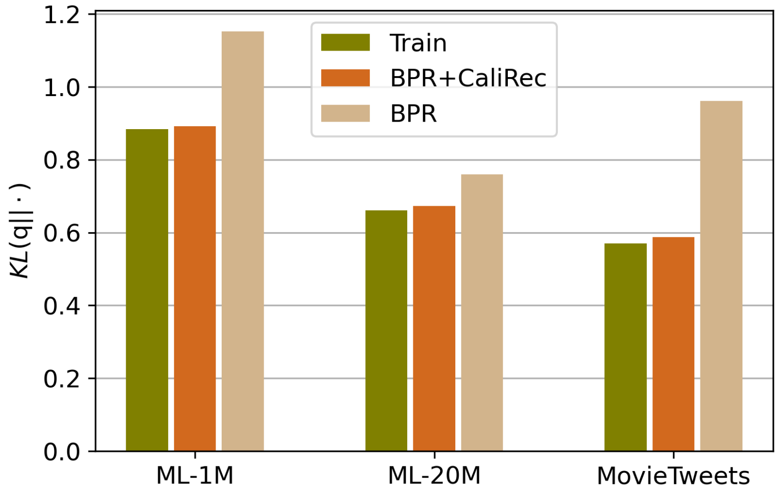


Figure (5.3) KL-Divergence between the taste distribution in ground truth (**q**) with the ones in training set (**p**), in CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and in BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$).‡

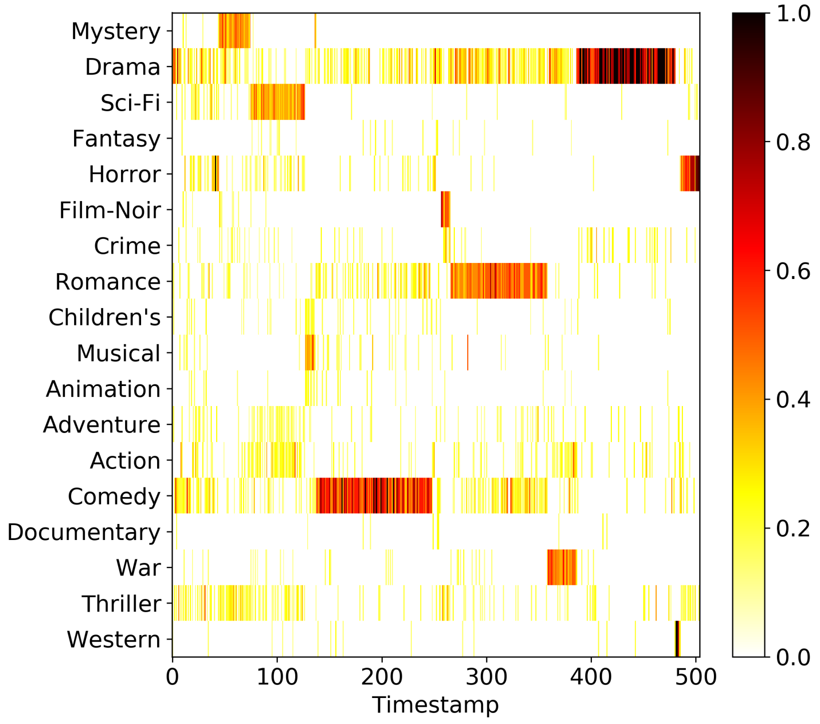


Figure (5.4) Trends of a sample user's taste overtime in the training set. As time goes by, this user changed her main taste through the path of Mystery $\rightarrow$ Sci-Fi $\rightarrow$ Comedy $\rightarrow$ Romance $\rightarrow$ War $\rightarrow$ Drama $\rightarrow$ Horror.[‡]

set ($\mathbf{q}$) versus the taste distribution on the training set ($\mathbf{p}$), the CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and the BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$). Similarly, we observe a large difference of the distribution between the training set and testing set (note the KL divergence may differ between $KL(\mathbf{a}||\mathbf{b})$ and $KL(\mathbf{b}||\mathbf{a})$). Both CaliRec and BPR return a poor estimation of the (unknown) taste distribution in the testing set, which is our ultimate goal. These results validate our hypothesis that both traditional recommender systems and distribution-aware recommender systems could under-serve the ground truth taste distribution. However, it also indicates that a better estimation of taste distribution may produce more accurate recommendation results.

5.2.3 An Example of Taste Distortion

Figure 5.4 clearly shows the time-series change of the tastes of a random user in the ML-1M dataset. Each row indicates the frequency of the genres the user interacted with over time. We can observe that as time goes by, this user changed tastes through the path of Mystery $\rightarrow$ Sci-Fi

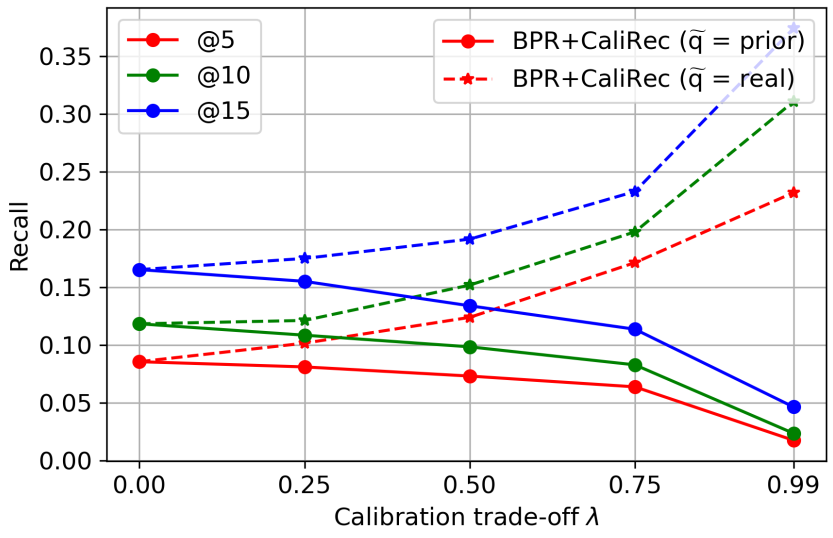


Figure (5.5) Recommendation results using CaliRec with prior and real taste distribution. Comparing with the BPR results ($\lambda = 0$), this "oracle" result is consistently better for all trade-off settings.[‡]

$\rightarrow$ Comedy $\rightarrow$ Romance $\rightarrow$ War $\rightarrow$ Drama $\rightarrow$ Horror. This user's jump in taste over time goes against the assumption that the taste distribution derived from a user's entire training set should be the target preference for the distribution-aware recommendation. Ignoring or underestimating the time-series trends and change of users' preferences could bring serious estimation errors to the distribution-aware recommendation results.

5.2.4 "Oracle" CaliRec with True Distribution

Finally, we explore the potential of improving distribution-aware recommendation if the real taste distribution was known. That is, given a user u, *if we know the real (though unknown) distribution for u in the next stage*, how does a conventional distribution-aware recommender perform? In other words, we would like to use the real taste distribution $\mathbf{q}_u$ instead of the prior taste distribution $\mathbf{p}_u$ as the target. Of course, $\mathbf{q}_u$ is not visible to a recommender in practice; therefore, we refer to this as an "oracle" result. Figure 5.5 shows the ideal results by CaliRec using the real taste distribution in the test set. As we can see, comparing with the base results ($\lambda = 0$, i.e., unchanged BPR result), this "oracle" result is significantly better on all trade-off weight settings. This result indicates that a better estimation of the user's taste distribution would bring improved distribution-aware recommendation results.

5.3 Our Approach: Taste-Enhanced Calibrated Recommendation

Our ultimate objective is mitigating both the *rabbit hole effect* and *taste distortion* simultaneously. Since the distribution-aware recommendation is inherently designed to alleviate the rabbit hole effect, can we also mitigate taste distortion? One idea is to dynamically learn the trends and shifts of each user's taste distribution to better estimate future taste preferences. Previous studies often learn a user's preference from the interactions in an embedded space, e.g., [127, 56]; here, rather than the embedded preference, we focus on predicting a user's explicit preferences based on the given categories of items. In the following, we introduce a Taste-enhanced calibrated Recommendation (**TecRec**) that is designed to learn a user's shift in preferences (Section 5.3.1), and then incorporate these shifts into a post-ranking framework for improved distribution-aware recommendation (Section 5.3.2).

5.3.1 Learning Taste Distribution

Previous studies in distribution-aware recommendation assume a user's taste preference should be similar to the historical preference in the entire training set or in the latest time window [18]. Yet, our observations in Section 5.2 show that a user's preferences frequently shift in each observed time window. Thus, in the following, we show how to predict a user's taste distribution $\hat{\mathbf{q}}$ in the next stage. Inspired by the time-series trends and changes of users' preferences (recall Figure 5.4), we propose the TecRec distribution prediction component to learn the evolving taste distribution of users, towards overcoming the taste distortion problem. We explore the potentials of neural network approaches to learn these taste shifts.

5.3.1.1 User's Taste Sequence

For each user-item interaction (u, i) for $u \in \mathcal{U}$ and $i \in \mathcal{I}$, we transform the user-item pair into user-genre $(u, \mathbf{c}_i)$ pair (refer to Equation 5.1). In this way, every user's historical interaction data could be transferred to a $1 \times |\mathcal{G}|$ vector, which represents the user taste distribution in this timestamp.

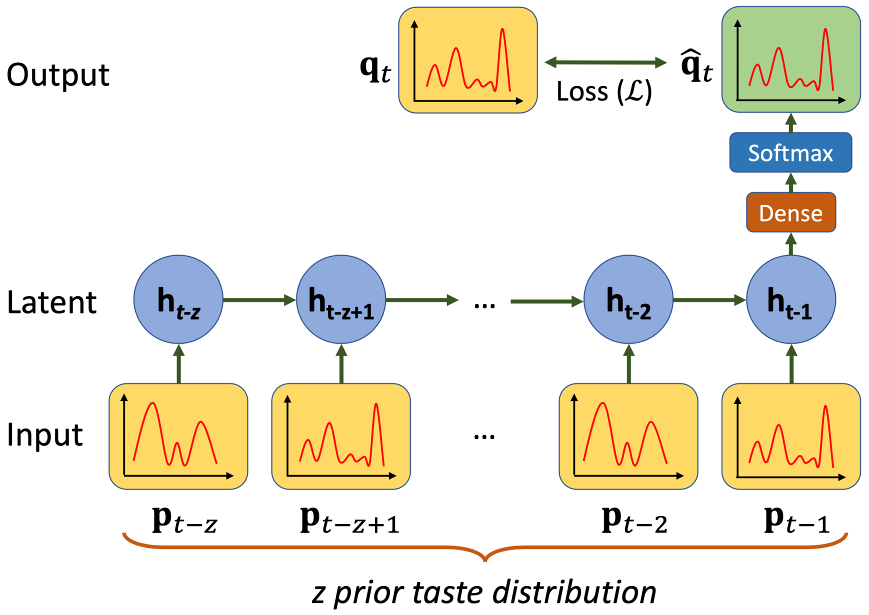


Figure (5.6) TecRec distribution prediction component, which takes the user's z prior time-series taste distributions as input, and outputs the predicted taste distribution in the next stage.‡

To summarize a user's taste distribution in each "stage" (over some duration, which may contain several items), we employ a hyper-parameter *step size*, β. That is, we slice the N user-item interaction history into $\frac{N}{\beta}$ stages, and each stage summarizes sequential β items this user interacted with. The taste distribution of each stage – t – can then be calculated as:

$$\mathbf{p}_{u,t} = \frac{1}{\beta} \times \sum_{j=t-\beta+1}^{t} \mathbf{M}_{u,i,j} \times \mathbf{c}_i \tag{5.7}$$

$\mathbf{M} \in \mathbb{N}^{|\mathcal{U}|\times|\mathcal{I}|\times|\mathcal{T}|}$ is a time-expanded binary 3-D matrix from $\mathbf{H}$, where $\mathbf{M}_{u,i,j} = 1$ indicates user u and item i have an interaction at time j.

In addition, we use a hyper-parameter *window size*, z, that is how many stages we use to predict the taste distribution at the next stage t. Our objective is: given a user's temporal taste distributions $[\mathbf{p}_{u,t-z}, \mathbf{p}_{u,t-z+1}, \cdots, \mathbf{p}_{u,t-1}]$, to predict the user's taste distribution in the next stage, $\hat{\mathbf{q}}_{u,t}$.

5.3.1.2 Recurrent Model

Recurrent neural networks have shown great success in capturing the implicit dynamics of user-item interactions in recommender systems (e.g., [56, 128, 129]). We now explore the potential of using recurrent neural networks to learn the trends and changes of a user's explicit taste. To better understand the structure of TecRec distribution prediction component, Figure 5.6 shows a simplified model structure for predicting the taste distribution for the next stage, $\hat{\mathbf{q}}_t$. To this end, TecRec takes the user's z prior time-series taste distributions, $[\mathbf{p}_{u,t-z}$, ..., $\mathbf{p}_{u,t-1}]$, as an input.

In our task, both sequential events (inputs) and predictions (outputs) are the user's explicit taste preferences (i.e., ratio of each explicit genre), and this preference vector is highly dense with size $1 \times |\mathcal{G}|$. For this reason, the prior preference sequence will directly join the next recurrent layer without embedding.

We use $\mathbf{h}_t$ to represent the latent vector at time t in the recurrent layer. A recurrent layer consists of z recurrent units. Here, we consider three variants of recurrent units for the task of taste preference prediction:

Traditional Recurrent unit (*abbr.* TRU) [130] takes the previous latent state h_{t-1} and current input taste preference $\mathbf{p}_t$ as input:

$$\mathbf{h}_t := \sigma(\mathbf{W}_q \mathbf{p}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b})$$

where **W**, **b** are the weights and bias term for each recurrent unit, and σ is a nonlinear activation function ($tanh$ in this chapter).

Gated Recurrent unit (*abbr.* GRU) [131] is a variant of traditional recurrent unit, which introduces two more gates on each unit – an update gate $\mathbf{z}_t$ and a reset gate $\mathbf{r}_t$ – to control the long short-term dependencies. The update gate $\mathbf{z}_t$ decides how much information from previous time steps is going to be retained, while the reset gate $\mathbf{v}_t$ determines how much of the previous information is to be forgotten with another recurrent state $\widetilde{\mathbf{h}}_t$. Lastly, the output $\mathbf{h}_t$ of the GRU cell at step t is the weighted sum of the current and the last hidden state.

Long Short-Term Memory unit (*abbr.* LSTM-unit) is another variant of a traditional recurrent unit that has been shown to outperform TRU on numerous temporal processing tasks [132, 133, 134, 135, 136]. Comparing with GRU, LSTM-unit introduces more gates: a forget gate to decide what information should be kept, and an output gate to decide what the next hidden state should be. Each LSTM-unit $\mathbf{h}_t$ consists of input gates $\mathbf{i}_t$, forget gates $\mathbf{f}_t$, output gates $\mathbf{o}_t$ and cell activation vector $\mathbf{c}_t$ at time t.

From the recurrent layer with recurrent units of TRU, GRU, or LSTM-unit, the final recurrent unit $\mathbf{h}_{t-1}$ outputs a $|\mathcal{G}|$-dimension vector, which will go through a dense layer and be activated by the softmax activation function. After activation, the output vector – the predicted preference ratio for each explicit aspect (i.e., a genre in our data) – is normalized into the same scale as the input, where the sum of the vector is 1. This vector would then be our predicted taste distribution, $\hat{\mathbf{q}}_t$, for the next stage t, calculated by:

$$\hat{\mathbf{q}}_t := softmax(\mathbf{W}_{dense}\mathbf{h}_{t-1} + \mathbf{b}_{dense}).$$

5.3.1.3 Learning and Evaluation

In the learning process, we adopt the Adaptive Moment Estimation (Adam) [104] method as the optimizer to train the model, since it yields faster convergence compared to SGD. To compare the output predicted distribution $\hat{\mathbf{q}}_{u,t}$ with the ground truth $\mathbf{q}_{u,t}$, we use a Kullback–Leibler (KL) divergence loss function, shown below:

$$\mathcal{L}(\mathbf{q}_{u,t}, \hat{\mathbf{q}}_{u,t}) = -\sum_{g_i \in \mathcal{G}} \mathbf{q}_{u,g_i,t} \times \log\left(\frac{\hat{\mathbf{q}}_{u,g_i,t}}{\mathbf{q}_{u,g_i,t}}\right) \tag{5.8}$$

Thus, given a regularization weight Φ and sample size N, the objective function with regularization for the model is to search for a choice of Θ (which includes all weights and bias term):

$$arg\min_{\Theta} \frac{1}{N} \sum \mathcal{L}(\mathbf{q}, \hat{\mathbf{q}}) + \Phi \times \|\Theta\|_F^2. \tag{5.9}$$

Through this step, the optimal predicted taste distribution in the next state t, $\hat{\mathbf{q}}_{u,g_i,t}\ \forall g_i \in \mathcal{G}$ will be used in the post-ranking stage.

Algorithm 1: TecRec Post-ranking Mechanism[‡]

```
Input: $u$, $\mathbf{D}_{u,:}$, $\hat{\mathbf{q}}$, $\lambda$, $Z$, $k$
Output: $\mathbf{T}_u$ ;        // Recommendation for $u$ from TecRec
$\mathbf{T}_u \leftarrow \varnothing$;
$\mathbf{d} = \underset{\searrow, Z}{\arg\operatorname{sort}} \left[\mathbf{D}_{u,1}, \mathbf{D}_{u,2}, \cdots, \mathbf{D}_{u,|I|}\right]$;
while $|T_u| < k$ do
    $i^* \leftarrow \underset{i \in \mathbf{d} \setminus \mathbf{T}_u}{\arg\max} \underbrace{(1-\lambda) \times \sum_{v \in \mathbf{T}_u \cup \{i\}} \frac{\mathbf{D}_{u,v}}{|\mathbf{T}_u|+1}}_{\text{accuracy term}} - \underbrace{\lambda \times \varphi\Big(\hat{\mathbf{q}}, \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_v\Big)}_{\text{calibration term}}$;
    $\mathbf{d} \leftarrow \mathbf{d} \setminus \{i^*\}$;
    $\mathbf{T}_u \leftarrow \mathbf{T}_u \cup \{i^*\}$ ;        // Update current optimal result
end
return $\mathbf{T}_u$;
```

5.3.2 Post-Ranking Mechanism

As many recommenders are trained in a pairwise manner, many studies state that one might not be able to include calibration into the training [18]. Therefore, a standard solution is post-ranking the predicted list in a post-processing step, which has been widely used in machine learning approaches [113, 114, 18]. Post-ranking approaches could be integrated into existing models without re-training the original model pipeline, bringing great convenience in practice. Thus, we propose a post-ranking mechanism that calibrates the recommendation result based on the learned "next stage" taste distribution, toward overcoming the taste distortion problem. This approach is summarized in Algorithm 1.

To obtain the calibrated results, TecRec takes as input a user u, predicted relevance scores $\mathbf{D}_{u,:}$, u's learned taste distributions $\hat{\mathbf{q}}$ (introduced in 5.3.1), a trade-off parameter λ, a candidates boundary Z (introduced below), and a required recommendation length k. To re-rank the Top-k most relevant items and let users' taste distribution $\widetilde{\mathbf{q}}_{\text{TecRec}}$ be as close to our learned distribution $\hat{\mathbf{q}}$, we consider the accuracy and closeness of distributions together for the ranking optimization. We can obtain the optimized new item list, $\mathbf{T}_u$, for user u as Algorithm 1 lines 3-6, where $\lambda \in [0, 1]$ is the calibration weight to control the balance between the original recommendation results and the distribution metric, and recommendation score $\mathbf{D}_{u,:}$ is provided from any base recommender system. Directly using the KL-divergence as the calibration function $\varphi(\cdot)$ to find the optimal set $\mathbf{T}_u$ is a combinatorial optimization problem and NP-hard [18]. Prior research [137] has shown that the greedy optimization of this problem could be equivalent to the greedy optimization of a surrogate submodular function. Hence, we can re-write the calibration term, $\varphi(\cdot)$, in Algorithm 1 Line 4 as follows:

$$\begin{aligned}
&\varphi\Big(\hat{\mathbf{q}}, \sum_{v\in\mathbf{T}\cup\{i\}} \mathbf{c}_v\Big) = KL\Big(\hat{\mathbf{q}}|| \sum_{v\in\mathbf{T}_u\cup\{i\}} \mathbf{c}_v\Big)\\
&= -\sum_{g_k\in\mathcal{G}} \hat{\mathbf{q}}_{g_k} \times \log\left(\frac{\sum\limits_{v\in\mathbf{T}_u\cup\{i\}} \mathbf{c}_{v,g_k}}{\hat{\mathbf{q}}_{g_k}}\right)\\
&= -\Big(\sum_{g_k\in\mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v\in\mathbf{T}_u\cup\{i\}} \mathbf{c}_{v,g_k} - \sum_{g_k\in\mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \hat{\mathbf{q}}_{g_k}\Big)\\
&= \underbrace{\text{Entropy}(u)}_{\text{constant}} - \sum_{g_k\in\mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v\in\mathbf{T}_u\cup\{i\}} \mathbf{c}_{v,g_k}
\end{aligned} \tag{5.10}$$

Therefore, updating the optimized $\mathbf{T}_u$ (Line 4) could be equivalent to:

$$\begin{aligned}
i^* \leftarrow \arg\max_{i\in\mathbf{d}\setminus\mathbf{T}_u} (1-\lambda) \times \sum_{v\in\mathbf{T}_u\cup\{i\}} \frac{\mathbf{D}_{u,v}}{|\mathbf{T}_u|+1}\\
+ \lambda \times \sum_{g_k\in\mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v\in\mathbf{T}_u\cup\{i\}} \mathbf{c}_{v,g_k}
\end{aligned} \tag{5.11}$$

where the greedy optimization of submodular functions achieves a $(1-\frac{1}{e})$ guarantee of optimality (e is Euler's number) [138].

Top-Z Selection for Post-ranking. To add each item into the calibrated recommendation results, a traditional post-ranking method would go through the entire candidate item list, $top(u, |\mathcal{I}|)$ with the size of $|\mathcal{I}|$), then select the one with the most optimal KL-weighted score. To save running time and maintain prediction quality, for each iteration to choose the optimal item to add into calibrated recommendation results, instead of going through the entire list of $top(u, |\mathcal{I}|)$, we only consider the Top-Z items in $top(u, |\mathcal{I}|)$, where Z is also a given hyper-parameter for Algorithm 1. In our experiments, we set Z as 30 times the number of valid genres for $\mathcal{G}$ (e.g., $Z = 540$ for $|\mathcal{G}|= 18$ in the MovieLens-1M dataset).

The benefits of only selecting candidates from the Top-Z items in the $top(u, Z)$ rather than the entire item set are not only significantly faster processing time ($Z \ll |\mathcal{I}|$), but also further ensuring recommendation quality. That is, we need not engage our post-ranking algorithm with items on the bottom of $top(u, |\mathcal{I}|)$, although it may slightly improve KL-divergence.

5.4 Experiments and Results

In this section, we conduct a series of experiments over the three datasets introduced in Section 5.2.1. Our goal is to examine the proposed distribution-aware recommendation in the presence of dynamic tastes, to achieve the ultimate goal of simultaneously mitigating both the rabbit hole effect and the taste distortion problem. We focus on two main questions: How well can we mitigate the taste distortion problem? And what impact does this have on recommendation?

5.4.1 Mitigating Taste Distortion

We begin by examining how well TecRec learns the evolving taste distribution of users. We first discuss hyper-parameter tuning, in terms of the *step size* and *window size*. Then, we analyze the results of TecRec with special attention to alternative approaches, including two diversity-focused recommenders (xQuAD and SPAD), a traditional accuracy-based recommender (BPR), and a popular sequential recommender system (SASRec).

To better reflect user's taste distribution in a stage, we employ a hyper-parameter *step size* – β – which selects consecutive β movies as a watching stage. $\beta_u = 1$ treats every watched movie as an individual series, which may not sufficiently express a user's taste distribution at that time. $\beta_u = \sum_{i \in \mathcal{I}} \mathbf{H}^R_{u,i}$ treats all watched movies as one single series to reflect the user's taste distribution (in essence, the standard assumption in the literature). The objective of tuning this hyper-parameter is to find an optimal value of *step size* to sufficiently express a user's taste distribution and effectively avoid the effects of outliers and noise.

5.4.1.1 Hyper-parameter Tuning

Figure 5.7 first shows the effects of *step size* on the KL-divergence. The result of hyper-parameter tuning suggests the KL-divergence firstly decreases with the growth of step size β. After

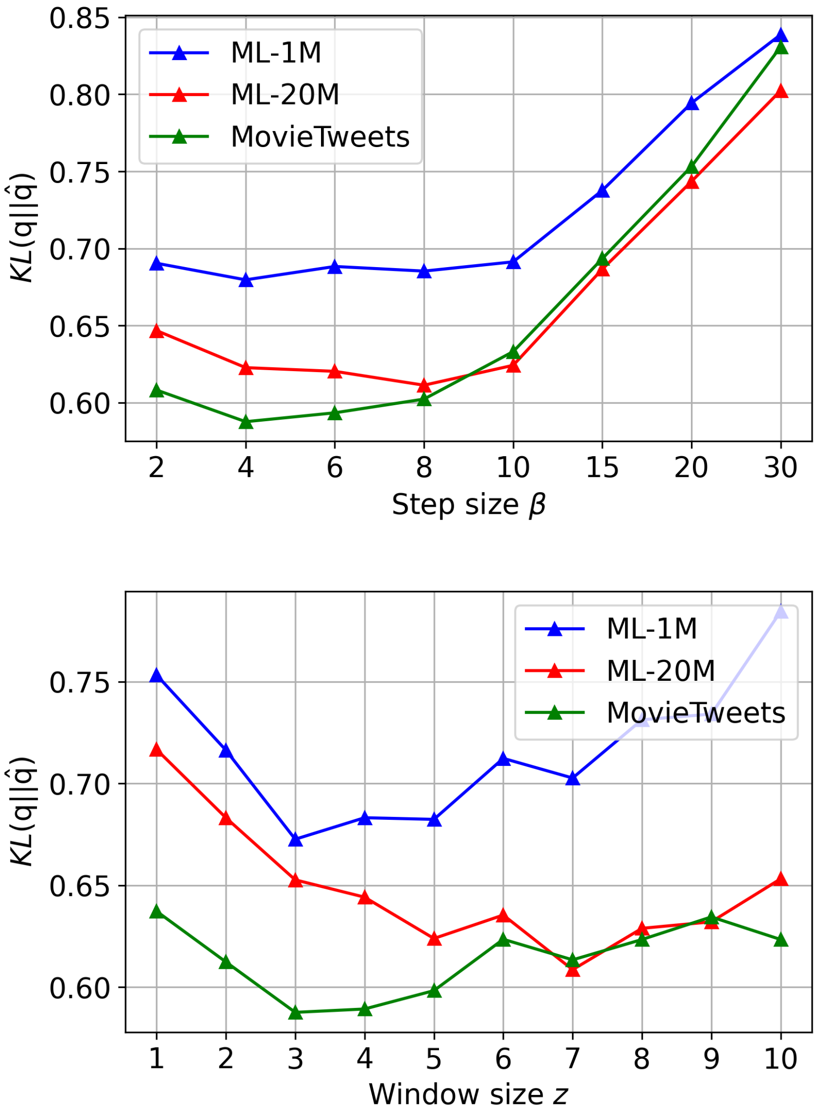


Figure (5.7) Hyper-parameters tuning for TecRec Distribution Prediction Component.‡

a minimal value ($\beta = 4$ for ML-1M and MovieTweets and $\beta = 8$ for ML-20M), KL-divergence sharply increases. Too small step size brings more outliers and noise into the training and may not represent a user's taste at a certain stage. Conversely, too large step size would flatten the taste preference and may not have a strong ability to express the shift from one genre to another.

We also tune another hyper-parameter of the model – *window size* (z, or lag value). Unlike the step size, which is to determine the number of movies in each window (stage), window size determines how many prior windows (stages) should be used for prediction. Recall our TecRec is a many-to-one recurrent model, which utilizes previous taste preference sequence – $[\mathbf{p}_{t-z}, \mathbf{p}_{t-z+1}, ..., \mathbf{p}_{t-1}]$ – as the inputs to predict the taste distribution in the next stage $\hat{\mathbf{q}}_t$. Figure 5.7 also shows the effects of *window size* on the KL-divergence. Similar to the step size, a too-small window size indicates there is no overlap in the training set, then it would be equivalent to not using a time relationship between elements of the training set of taste sequence. However, due to the limited number of user interactions, a too-large window size indicates we have to lose many cold users in the training set, which would also harm the model's training performance. In the following, we set the window size $z = 3$ for ML-20M and MovieTweets dataset, and set $z = 7$ for ML-1M dataset; and we set the step size $\beta = 4$ for ML-1M and MovieTweets and $\beta = 8$ for ML-20M, based on the tuning results.

5.4.1.2 Comparison with Alternatives

To compare the learning performance of TecRec for predicting a user's taste distribution, we choose the three variants of the recurrent unit (TRU, GRU, and LSTM-unit) for TecRec and six competitor methods: two widely-used assumptions (allTrain and lastTrain), an accuracy-driven recommender (BPR), a sequential recommender (SASRec), and two diversity-focused recommenders (xQuAD and SPAD), briefly introduced as below:

- **allTrain**: this widely-used method assumes taste distribution should be the same as the historical distribution in the training set;
- **lastTrain**: this method assumes the taste distribution in the next observed window should be

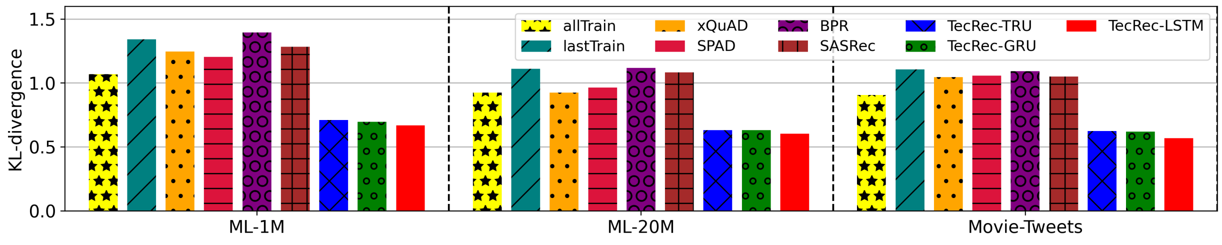


Figure (5.8) Dynamic taste distribution prediction results.[‡]

similar to the latest time window in the training set;

- **BPR**: a traditional accuracy-driven recommender with a generic optimization criterion for optimal personalized ranking [109];

- **SASRec**: a popular sequential recommender using a two-layer Transformer decoder to capture user's sequential behaviors and achieving state-of-the-art results [56];

- **xQuAD**: a diversity-focused recommender proposed in [125] adapted from the Query Aspect Diversification framework [139], where user u's preferences are formulated as a probability distribution over aspects (i.e., genres);

- **SPAD**: a variant of xQuAD proposed in [86], which uses the same objective function and greedy post-ranking approach as xQuAD, but uses sub-profiles as aspects to model the user's interests rather than item features.

Since the ground truth in the next window has a fixed size (i.e., β), for the two diversity-focused recommenders and the sequential recommender, we measure the taste distribution from their Top-10 recommended results, i.e., $\widetilde{\mathbf{q}}_{\text{xQuAD}}^{\text{Top}-10}$, $\widetilde{\mathbf{q}}_{\text{SPAD}}^{\text{Top}-10}$, $\widetilde{\mathbf{q}}_{\text{BPR}}^{\text{Top}-10}$, and $\widetilde{\mathbf{q}}_{\text{SASRec}}^{\text{Top}-10}$. Recall that these methods aim for diversity or to better capture a user's dynamic shifts in preference; however, they are not designed with recommendation distribution in mind, so the resulting recommendations will not respect the requirements of taste distribution.

Figure 5.8 shows the results for dynamically learning and predicting users' taste distribution in the testing set for all three datasets. In terms of KL-divergence between the true taste distributions

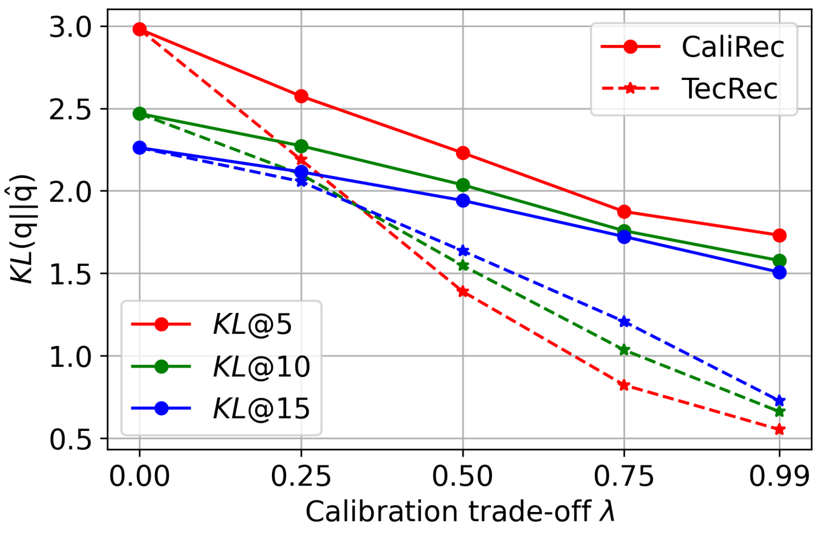


Figure (5.9) KL-divergence between the true distribution $\mathbf{q}$ with $\widetilde{\mathbf{q}}^{\text{Top}-k}_{\text{CaliRec}}$ and $\widetilde{\mathbf{q}}^{\text{Top}-k}_{\text{TecRec}}$ on all calibration trade-off settings.[‡]

and the predicted ones, TecRec-LSTM performs the best among all baseline methods. First of all, for the intra-comparison of the baselines, lastTrain performs worse than allTrain and most others, which indicates users may not always follow the latest taste in the next watching stage. TecRec-LSTM improves over $30\%+$ from allTrain and over $40\%+$ from lastTrain on all datasets. Secondly, traditional diversity-focused recommenders, xQuAD and SPAD, perform worse than the baseline allTrain, which indicates that these methods may improve the latent diversity of the recommendation results; however, they do not follow the distribution-sensitive requirements of distribution-aware recommendation. Similar results hold for BPR and the sequential recommender SASRec. Thirdly, the time-series-based neural network models with three types of recurrent units obtain significantly better results than the others, which suggests that the pattern of taste shifts could be recognized and learned by these models. Of the three variants of TecRec, the one with LSTM-unit results in the best prediction results (and so will be used in the discussions in Section 5.4.2), though the particular choice appears not to be critical.

5.4.1.3 Calibration Trade-off λ

Recall that we can control the degree of calibration with λ, a "customized" knob to control the influence of calibration which is widely used (e.g., [113, 18]). With a choice of $\lambda = 0$, we default to the baseline recommender (e.g., BPR). Figure 5.9 shows the impact of calibration on both CaliRec and the proposed TecRec in terms of the KL-divergence between the true taste distribution and the taste distribution of final recommended items. Solid lines correspond to CaliRec, and dashed lines for TecRec. As we can see, comparing with the results from CaliRec, the results by our proposed methods have significantly lower KL-divergence on each calibration-weight settings. One of the interesting observations from the three solid lines is, in both the BPR ($\lambda = 0$) and CaliRec ($\lambda > 0$) results, a larger K is often accompanied by a better KL-divergence (meaning less taste distortion). This may indicate that recommendation accuracy would be worse with a smaller K (recall that a wrongly assumed taste distribution would result in worse recommendation accuracy), which we will validate below. We also observe that the improvement from CaliRec is more impressive when K is small, e.g., $K = 5$. For example, when $\lambda = 0.5$, the $KL@5$ is improved 37.7% (from 2.23 to 1.39), which is more than the improvement of $KL@10$ (24.1%) and $KL@15$ (15.9%). Besides, we also observe that the KL-divergence could not be 0 due to the predicted error of taste distribution and the Top-Z selection mechanism, even if we set a large calibration weight, e.g., $\lambda = 0.99$. Up to now, TecRec shows the effectiveness and robustness to simultaneously ameliorate both the rabbit hole effect and taste distortion. However, what impact does TecRec have on recommendation?

5.4.2 Improving Recommendation

Our previous experiments demonstrate the viability of mitigating the taste distortion problem by predicted a distribution that is close to the real taste distribution, compared with two widely-used assumptions (allTrain and lastTrain), an accuracy-driven recommender (BPR), a sequential recommender (SASRec), and two diversity-focused recommenders (xQuAD and SPAD). Next, we explore the impact on recommendation results using this predicted distribution. We consider two variants of the proposed Taste-enhanced calibrated Recommender based on the two accuracy-driven

		Recall@10				NDCG@10			
		+ CaliRec	+ TecRec	Δ_{Base}	Δ_{CaliRec}	+ CaliRec	+ TecRec	Δ_{Base}	Δ_{CaliRec}
BPR	λ =0	0.1182	0.1182			0.1975	0.1975		
	λ =0.25	0.1166	0.1244	+5.25%	+6.69%	0.1939	0.2053	+3.95%	+5.88%
	λ =0.5	0.1106	0.1306	+10.49%	+18.08%	0.1827	0.2198	+11.29%	+20.31%
	λ =0.75	0.0858	0.1370	+15.91%	+59.67%	0.1687	0.2103	+6.48%	+24.66%
	λ =0.99	0.0264	0.0887	-24.96%	+235.98%	0.1274	0.1763	-10.73%	+38.38%
SASRec	λ =0	0.1396	0.1396			0.2368	0.2368		
	λ =0.25	0.1284	0.1426	+2.15%	+11.06%	0.2252	0.2421	+2.25%	+7.51%
	λ =0.5	0.1172	0.1456	+4.30%	+24.18%	0.2124	0.2527	+6.73%	+18.99%
	λ =0.75	0.1027	0.1458	+4.44%	+41.97%	0.1912	0.2468	+4.25%	+29.11%
	λ =0.99	0.0610	0.1033	-26.00%	+69.34%	0.1077	0.1718	-27.44%	+59.55%

Table (5.2) Recommendation results, Recall@10 and NDCG@10 on ML-1M dataset by setting different calibration weights. Columns Δ_{Base} shows the improvement of our proposed TecRec from two basis recommenders, BPR and SASRec. And Δ_{CaliRec} shows the improvement from CaliRec.[‡]

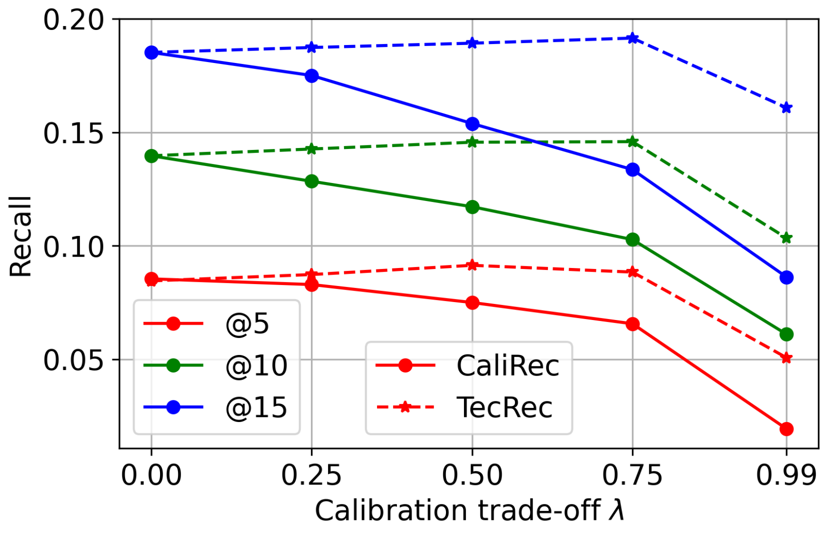
recommenders: BPR and SASRec. **Recall that the TecRec post-ranking component can be adapted to the recommendation results of any base recommender without re-training.**

For fair comparison, we consider the first stage – which contains β (step size, $\beta = 4$ for ML-1M and Movie-Tweetings dataset and $\beta = 8$ for ML-20M dataset) movies – in the test set as the ground truth. Since the length of ground truth data is fixed, we use $Recall@K$ and $NDCG@K$ as the evaluation metrics.

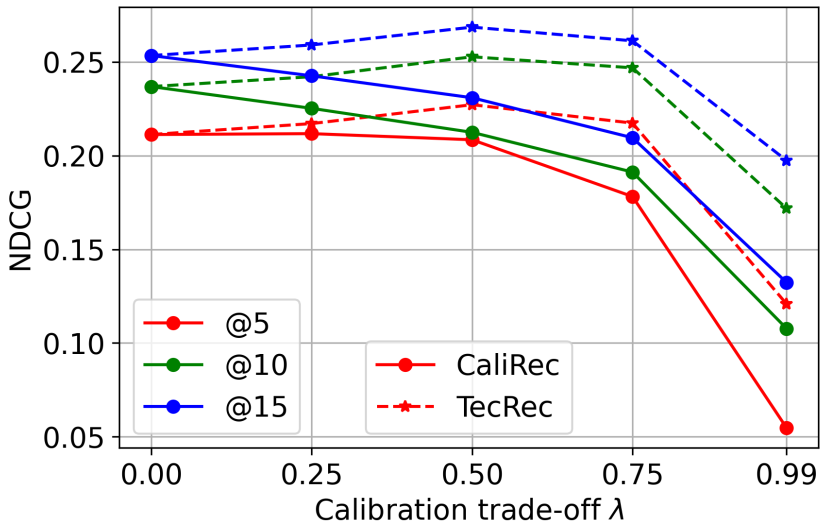
Table 5.2 first shows the recommendation results on the ML-1M dataset (the other two datasets present similar observations and results) comparing with CaliRec for different settings of the calibration weight and the two base recommenders (BPR and SASRec). Column Δ_{CaliRec} presents our improvement rate of recommendation performance from CaliRec. On the one hand, TecRec obtains better results ($Recall@K$ and $NDCG@K$) than the traditional CaliRec algorithm. For example, when we set the calibration weight $\lambda = 0.75$, $Recall@10$ improves 59.7% from BPR + CaliRec and 42.0% from SASRec + CaliRec settings, respectively. This improvement indicates that a reasonable estimation of users' taste distribution would provide not only a more reasonable recommender list but also more accurate results.

Table 5.2 also shows the comparison of the recommendation results between the proposed TecRec with BPR and SASRec (refer to Column Δ_{Base}). The $Recall@K$ and $NDCG@K$ improves in most of the calibration-weight settings (except $\lambda = 0.99$), e.g., when we set the calibration weight $\lambda = 0.5$, $Recall@10$ improves 10.5% from BPR and 4.3% from SASRec, respectively. Firstly, this improvement shows that there is not necessarily a trade-off between the distribution adjustment and accurate recommendation. A better estimation of taste distribution will produce more accurate recommendation results. Secondly, the recommendation results improve by different amounts for different base methods (e.g., $\Delta_{BPR} = 10.5\%$ v.s. $\Delta_{SASRec} = 4.3\%$). One possible reason for the larger improvements of BPR + TecRec over BPR is TecRec's inherent consideration of sequence (which is absent from BPR). In contrast, the improvement over SASRec is smaller, possibly since sequence is already part of that method. Note however that conventional sequential recommenders are still limited in addressing "taste distortion", which uses explicit categories as the preference rule (refer to Figure 5.8). Third, comparing the results with the oracle results (refer to Figure 5.5), there is still much room for improvement by better estimating the taste distribution.

Besides, for evaluating the recommendation results in all aspects, we also display the $Recall@K$ and $NDCG@K$ comparing TecRec with CaliRec with the same base – SASRec – on Figure 5.10. For each setting of K, TecRec performs better than CaliRec for all settings of calibration trade-off, λ, and performs even better than SASRec for most settings of λ (excepts $\lambda = 0.99$). More specifically, with the growth of K, the improvement from the CaliRec increases more quickly (refer to Figure 5.10). For example, when we set $\lambda = 0.75$ on the ML-1M dataset, comparing with the 34.6% improvement from SASRec + CaliRec on $Recall@5$ results, the improvement increases to 42.0% and 43.4% on $Recall@10$ and $Recall@15$ recommendations, respectively. However, on the contrary, this improvement from the base recommender only increases more slowly. For example, when $\lambda = 0.75$, comparing with the 4.59% improvement from SASRec on $Recall@5$ results, the improvement drops to 4.44% and 3.40% on $Recall@10$ and $Recall@15$ recommendations, respectively. Similar observations and results are presented on $NDCG@K$. One explanation is traditional methods provide less taste distortion when K is larger (recall Figure 5.9). Another explanation is that every user's testing data has been set to only the next stage with β movies. Therefore, the positive effect from the predicted taste distribution could be tapering off. This



(a) Recall@K



(b) NDCG@K

Figure (5.10) Recommendation results comparison between CaliRec and TecRec with the same base SASRec on ML-1M dataset.[‡]

observation also motivates our continued research: how can we accurately predict users' taste distribution for the next few stages, rather than just one?

5.5 Summary

In this chapter, we first identified the taste distortion problem – a representative instance of the distribution bias of the predicted user's taste. We empirically showed the prevalence of this problem through a data-driven study. Then, we proposed a Taste-enhanced calibrated Recommender (TecRec) that incorporates a time-series neural network sub-model to predict users' preference shifts. Results show TecRec improves both taste distribution estimation (i.e., mitigating both the rabbit hole effect and the taste distortion problem) and recommendation quality, compared with traditional distribution-aware recommenders, as well as diversity-focused and sequential recommenders. Besides, the proposed recommender offers interpretable taste distribution (by respecting a user's existing preferences), and can be integrated into existing recommenders without re-training the original recommendation pipeline.

6. MITIGATING DISTRIBUTION BIAS IN DYNAMIC RECOMMENDATION

In Chapter 4 and Chapter 5, we investigated two types of distribution bias in recommendation results: the target customer distortion problem and the taste distortion problem. These two studies follow the style of most other state-of-the-art distribution-aware recommenders by focusing on the one-shot static setting. In this chapter, we study the general class of distribution bias in a dynamic environment. That is, we consider a recommendation setting where users and the system dynamically evolve, which more closely reflects how users engage with these systems. We begin this chapter by clarifying the key differences of distribution bias between the static and dynamic environments.

As we discussed in Chapter 5, echo chambers and rabbit holes are two serious threats to the ongoing success of recommendation systems. Both can narrow down a user's interest areas, leading to undesirable (and often unforeseen) outcomes [18, 118, 119, 120, 21, 22, 23]. As discussed before, there is evidence of users descending into ideological bubbles within just a few clicks [120]. While there are many factors that contribute to the rise of such threats, we view these as examples of a general class of *distribution bias* problems. Distribution bias arises due to a preference misalignment that can be iteratively exacerbated as a recommendation system evolves. For example, consider an ostensibly gender balancing job candidate system that initially identifies slightly more men for software engineering positions. Over time, the system's initial preference for gender balance (the distribution of interest in this case) may be skewed towards identifying mostly men due to a feedback loop if the system ignores the preferred balanced gender distribution.

Figure 6.1 illustrates this problem of *distribution bias*. At time t, the user clicks three blue items and one orange item from the system's recommendation list (here, colors could correspond to genders, political ideology, genres, or other item attributes of interest). Once the system is retrained based on this new feedback, it recommends a new list at time $t+1$. But what should the distribution of these items be with respect to the attributes of interest?

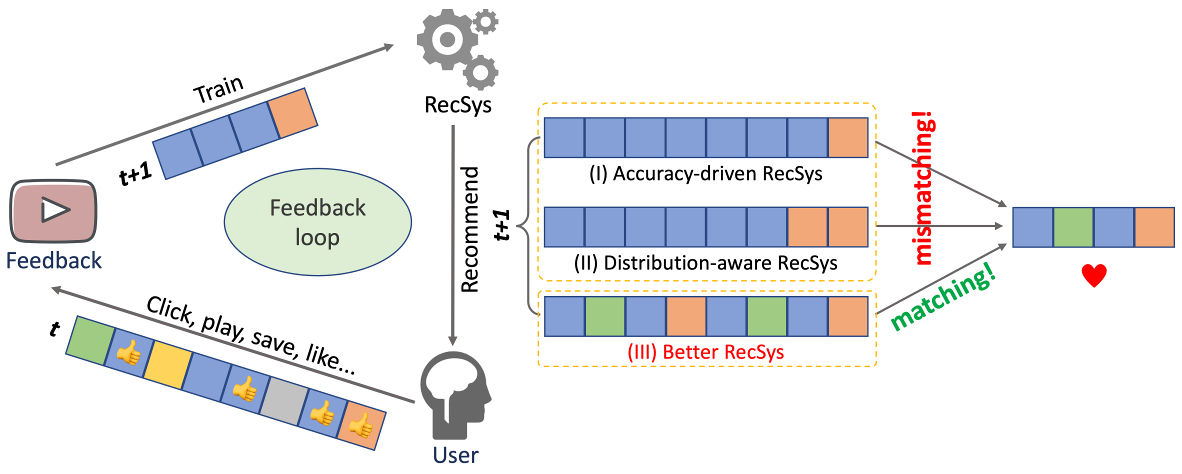


Figure (6.1) An illustration of preference misalignment in dynamic recommendation

- (I) An accuracy-driven recommender (i.e., one that focuses solely on optimizing an accuracy metric like precision or NDCG without regard for distributional impacts) may amplify a user's interests in blue items, leading to an over-emphasis on blue items to the detriment of other items; over time, the feedback loop of new user feedback and model retraining may further exacerbate this problem.
- (II) A distribution-aware recommender [18, 77, 123, 24] aims to ensure that a particular preference distribution is reflected in the recommendations that a system makes, so that the system neither over-emphasizes main interest areas, nor under-serves lesser areas. However, existing works have focused primarily on one-shot static settings, ignoring the distorting effects present in dynamic environments, or fixing on a user's prior preferences without regard to changes over time in these preferences.
- (III) A better recommender, we posit, should seek to balance distribution preferences with the utility of the system, and do so intelligently as the system evolves. We hypothesize that by dynamically estimating a user's preference distribution in the next stage and recommending appropriate items, this could lead to a mitigation of distribution bias and more engagement from the user (e.g., in terms of clicks or other feedback).

But first, is distribution bias a real problem as we have asserted? While there has been recent interest in simulating and modeling dynamic recommendation in the context of feedback loops, e.g., [140, 141, 142, 64, 65, 67], there are no studies of distribution-aware recommendation or distribution bias in this setting. Hence, this chapter begins by conducting a data-driven study of distribution bias in dynamic recommendation (in Section 6.2). We find that distribution bias is a seemingly inherent outgrowth of traditional accuracy-based recommenders, wherein a small initial distribution bias grows as the dynamic recommendation iterates. Further, we demonstrate that existing distribution-aware approaches that have been proposed in the context of static recommendation (i.e., in a traditional one-shot train/test split scenario, versus a dynamic system that iteratively updates as is more common in real-world applications) does not mitigate this distribution bias in dynamic recommendation settings and may even increase the bias level.

Based on this study, we next propose a dynamic *distribution-correction* framework that aims to close the gap between the "better" recommender and what is practical (in Section 6.3). The proposed method dynamically corrects the preference distribution in each loop of recommendations based on the consideration of both a user's prior interactions and dynamic preference shift. The proposed method carefully incorporates user feedback in the dynamic recommendation loop to mitigate distribution bias.

Finally, with experiments based on real-world datasets, we analyze the effectiveness of the proposed approach and compare it with both accuracy-driven recommenders and conventional distribution-aware recommenders (in Section 6.4). Empirical results show the proposed method can strongly mitigate distribution bias by recommending items with a closer preference distribution to a user's historical and future preferences, resulting in an increase in agreement with the recommended items and improved recommendation utility.

6.1 Preliminaries

Suppose we have a user set $\mathcal{U} = \{1, 2, 3, \ldots, M\}$, and an item set $\mathcal{I} = \{1, 2, 3, \ldots, N\}$ in the system. We use a binary *user-item interaction matrix* $\mathbf{H} \in \{0, 1\}^{M \times N}$ to capture a user's activities over time, where $\mathbf{H}^{0:t}_{u,i} = 1$ indicates that user $u \in \mathcal{U}$ has interacted with item $i \in \mathcal{I}$ up to time t, otherwise $\mathbf{H}^{0:t}_{u,i} = 0$. This interaction matrix $\mathbf{H}$ is updated after every user activity (e.g., clicking an item). From an oracle perspective that can view all user-item interactions, we refer to such an "all-seeing" matrix as $\mathbf{H}^{\infty}$; therefore, we define the rest of the ground truth interactions as $\mathbf{H}^{t:\infty} = \mathbf{H}^{\infty} - \mathbf{H}^{0:t}$.

Furthermore, we assume there are labels of interest associated with each item. These labels can vary by domain, but might reflect item categories, gender, location, or other attributes that are important from a distribution-aware perspective. Without loss of generality, we refer to these labels as a genre set $\mathcal{G} = \{1, 2, 3, \ldots, G\}$. Since an item may belong to more than one genre, each item i has a genre vector $\mathbf{c}_i$:

$$\mathbf{c}_i = \left[\frac{\kappa_{i,g_1}}{\Sigma}, \frac{\kappa_{i,g_2}}{\Sigma}, \cdots, \frac{\kappa_{i,g_G}}{\Sigma}\right] \tag{6.1}$$

where $g_j \in \mathcal{G}$, $\Sigma = \sum\limits_{g_j \in \mathcal{G}} \kappa_{i,g_j}$, and κ_{i,g_j} is a binary variable where $\kappa_{i,g_j} = 1$ if i belongs to genre g_j.

Preference Distribution of Past and Future. In the following, we denote the historical preference distribution over these genres up to time t as $\mathbf{Past}^t$. We first define u's preference ratio for genre g_j up to time t as $\mathbf{Past}^t_{u,g_i}$:

$$\mathbf{Past}^t_{u,g_j} = \frac{\sum_{i \in \mathcal{I}} \mathbf{H}^{0:t}_{u,i} \times \mathbf{c}_{i,g_j}}{\sum_{i \in \mathcal{I}} \mathbf{H}^{0:t}_{u,i}} \tag{6.2}$$

where $\mathbf{H}^{0:t}_{u,i}$ is the entry in u^{th} row and i^{th} column of $\mathbf{H}$ up to time t. Thus, we define user u's entire preference distribution up to time t as a vector:

$$\mathbf{Past}^t_u = \left[\mathbf{Past}^t_{u,g_1}, \mathbf{Past}^t_{u,g_2}, \cdots, \mathbf{Past}^t_{u,g_G}\right]. \tag{6.3}$$

Similarly, we can compute user u's preference ratio for genre g_j in the future as $\mathbf{Future}^t_{u,g_j}$. Although $\mathbf{Future}^t_{u,g_j}$ is unknown to the system, we can use it to evaluate how well the distribution of a recommendation matches a user's real (future) preference distribution.

Preference Distribution of Recommendation and Click. Following the design of many platforms, we assume the system recommends to each user a list of k items. That is, in each iteration, the recommender system would provide the Top-k items to a user, based on the predicted user-item relevance for this iteration. We call such a recommendation list for u as $\mathbb{E}^t_u$, which is a ranked list with length of k. Thus, we calculate the *preference distribution of iterative recommendation*, $\mathbf{Rec}^t_u$, of predicted Top-k recommended items at time t as follows:

$$\mathbf{Rec}_u^t = \frac{1}{k} \sum\nolimits_{i \in \mathbb{E}_u^t} \mathbf{c}_i. \tag{6.4}$$

From the Top-k recommendation list $\mathbb{E}_u^t$, a user would interact with (e.g., click) ones she liked. We denote the set of items clicked from $\mathbb{E}_u^t$ at time t as $\mathbb{C}_u^t \subseteq \mathbb{E}_u^t$. We have $\mathbf{Rec}_u^t$ to express the preference distribution of the iterative recommendation; similarly, we denote $\mathbf{Click}_u^t$ to express the preference distribution of the user's interactions in the current iteration:

$$\mathbf{Click}_u^t = \frac{1}{|\mathbb{C}_u^t|} \sum\nolimits_{i \in \mathbb{C}_u^t} \mathbf{c}_i. \tag{6.5}$$

Distribution Closeness. To compare the distance between two preference distributions **a** and **b**, we use the Kullback-Leibler (KL) divergence [115] as the distribution closeness measure $\varphi(\cdot)$, where $\varphi(\mathbf{a}||\mathbf{b}) = 0$ indicates the distributions $\mathbf{a}_{g_j}$ and $\mathbf{b}_{g_j}$ are exactly the same for all $g_j \in \mathcal{G}$; and a larger $\varphi(\mathbf{a}||\mathbf{b})$ indicates they are more distant. $\varphi(\cdot)$ is defined as below:

$$\varphi(\mathbf{a}||\mathbf{b}) = -\sum_{g_j \in \mathcal{G}} \mathbf{a}_{g_j} \log \frac{\mathbf{b}_{g_j}}{\mathbf{a}_{g_j}}. \tag{6.6}$$

So far, we have defined four important preference distributions. Two correspond to a cumulative perspective: the user's historical distribution $\mathbf{Past}^t$ and the user's future distribution $\mathbf{Future}^t$. Two correspond to an iterative perspective: the distribution at time t of recommendations $\mathbf{Rec}^t$ and the distribution at time t of the user's clicks $\mathbf{Click}^t$. Given the distance $\varphi(\cdot)$, we analyze the distribution closeness from the following perspectives:

- $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$: How does the current recommended preference distribution match the user's historical distribution? Conventional recommenders ignore this preference distribution, leading to distribution bias such as in the rich get richer effect [18, 24]. In contrast, the main purpose of a distribution-aware recommender is to reduce $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$ to mitigate such bias.

- $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$: How does the current recommended preference distribution match the user's real future distribution? A large $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$ indicates the presence of distribution bias

Algorithm 2: Dynamic Recommendation

```
Initialization: Collect initial feedback $\mathbf{H^0}$ from users by showing random $k$ items.
Bootstrap: Train recommendation model by $\mathbf{H}^0$, and get the predicted user-item relevance.
for $t = 1 : T$ do
    Recommends Top-$k$ items $\mathbb{E}_u^t$ to $u$ based on predicted user-item relevance from $\mathbf{H}^{t-1}$;
    Collect $u$'s new clicks $\mathbb{C}_u^t$ in iteration $t$;
    $\mathbf{H}^t = \mathbb{C}_u^t + \mathbf{H}^{t-1}$;
    if $t \mod L == 0$ then
        Re-train model by $\mathbf{H}^t$ ;
        Update predicted user-item relevance from $\mathbf{H}^t$;
end
```

between the recommendation list and a user's evolving interests [143]. Such evaluation has been ignored by traditional distribution-aware recommenders that has been proposed in a static setting.

- $\varphi(\mathbf{Click}^t \| \mathbf{Rec}^t)$: How well does the user agree with the preference distribution provided by the current recommendation? Similar to the previous perspective, the consideration of $\varphi(\mathbf{Click}^t \| \mathbf{Rec}^t)$ has not been studied in traditional distribution-aware recommenders. For example, suppose among the current top-20 job candidates recommended, there is a 80%:20% ratio between females and all other genders. If four of the 20 candidates are interviewed (in our notation, a "click") but the ratio is 50%:50% for these four, then there is a large gender distribution gap.

In the rest of the chapter, we first aim to investigate the degree to which such distribution bias does manifest in dynamic recommendation. Then we propose a dynamic *distribution-correction* framework that aims mitigate this distribution bias. Finally, we empirically evaluate the proposed approach and compare it with both accuracy-driven recommenders and conventional distribution-aware recommenders.

Dynamic Recommendation. To study the presence of distribution bias over time, we investigate a dynamic recommendation process as summarized in Algorithm 2. In the beginning, for each user, the system randomly exposes several items to bootstrap the user and thus collects initial user-item clicks. Based on the initial feedback data $\mathbf{H}^0$, a recommendation model is trained. Then, as users come to the system one by one, the system uses the up-to-date model to provide k ranked items $\mathbb{E}_u^t$ as recommendations and collects new user-item clicks $\mathbb{C}_u^t$. After every L users visit, the system retrains the recommendation model with all clicks collected up to now.

6.2 A Data-driven study of Distribution Bias in a Dynamic Environment

This section begins our investigation of distribution bias in dynamic environments. We adopt two datasets that contain clearly defined genres, which help identify a preference distribution. The first dataset is the MovieLens-100K (*abbr.* ML-100K) dataset [96], which contains 100,000 user-movie interactions collected from 943 users and 1,862 movies. The second is the Amazon CD & Vinyl (*abbr.* Amazon-CD) dataset [95], which contains 24,225 user-item interactions collected from 744 users and 798 songs, where we keep only users who have at least 100 interactions and items that have 20 interactions for density consideration. More details are shown in Table 6.1. For each item, we use the genre available in each dataset: ML-100K contains 19 genres, and Amazon-CD contains 11 genres. One item may belong to one or several genres. This genre information lets us build each user's preference distribution. We feed these datasets into our dynamic recommendation environment and analyze the first 5,000 recommendation iterations.[*] In the following, we study the preference distortion problem in the context of a standard recommender, Matrix Factorization (MF), and a conventional distribution-aware recommender (CaliRec) [18]. Experiments with other tested algorithms show similar results; our emphasis here is on the general problem of distribution bias in dynamic recommendations.

[*]Users in our simulation have a fixed number of iterations with items (ground truth, $\mathbf{H}^\infty$). As time goes by, the rest of the ground truth $\mathbf{H}^{t:\infty}$ for each person will gradually decrease. Therefore, using the preference distribution of a limited number of ground truth data ($\mathbf{Click}^{t:\infty}$) would not be sufficient to express users' real preference distribution in the future. To avoid such distortion, we checked the ratio of cumulative clicks in each iteration out of the whole ground truth of two datasets used in this work. We found, up to iteration $t = 5,000$, the cumulative click ratio only reaches 27.6% of all clicks in the ML-100K dataset and reaches 22.4% in Amazon-CD dataset, respectively. That is, there are still more than 70% ground truth are not revealed by the system. In other words, the pool with such a great number of ground truth is still trustworthy to express users' real preference in the future (after iteration t). Therefore, unless particularly stated, the following analysis will be shown based on iteration $t = 0$ to $5,000$.

	#User	#Item	#Genre	#Interaction	Density(%)
ML-100K	943	1,862	19	1,000,209	6.30
Amazon-CD	744	798	11	24,225	4.08

Table (6.1) Dataset statistics for distribution bias analysis.

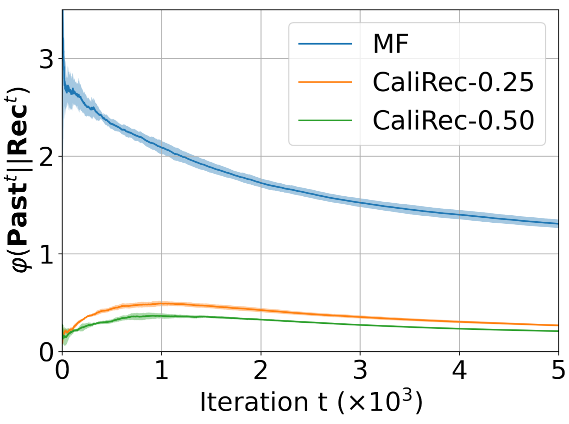
6.2.1 Distribution Bias: Analysis and Observations

Traditional approaches for making recommendations are not distribution-aware by design. Therefore, as many related research works have found, the recommendation results may cause preference biases, e.g., echo chambers or diversification issues [18, 24]. However, recently, distribution-aware recommendations (e.g., CaliRec) try to match the preference distribution of the recommendations to a user's historical interactions [24, 18]. In our dynamic environment, the objective is then to match the preference distribution of the recommendations in each iteration ($\mathbf{Rec}^t$) with the distributions revealed through historical interactions ($\mathbf{Past}^t$): a lower value of $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$ indicates better distribution matching (and less distribution bias). As we have discussed in Section 6.1, a dynamic environment raises other important criteria which we shall study as well.

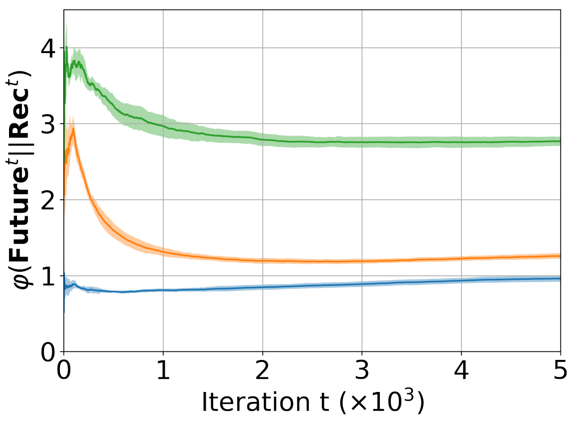
In the following, we use ML-100K as an example to fully analyze the results, and the other dataset presents similar observations and results. Through this analysis, we aim to illustrate how traditional accuracy-driven recommenders introduce distribution bias into the dynamic environment, and uncover if conventional distribution-aware recommenders could truly mitigate this distribution bias, or possibly introduce new side effects as well.

6.2.1.1 *Current Recommendation vs. History:* $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$

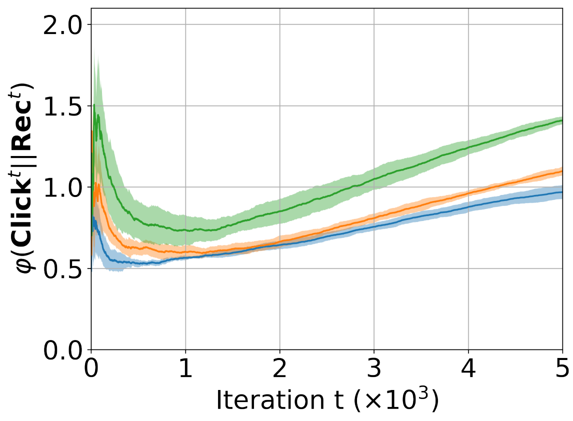
First, we evaluate the distribution bias between a user's historical preferences with the current recommendation's preferences by measuring $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$, shown in Figure 6.2 (a). We observe that as the system evolves, the recommended preferences by MF are gradually close to the historical preferences (the blue line representing $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$ drops down), illustrating that the system is learning a user's preferences from the historical interactions. The distribution bias between a user's history and the recommendation is gradually mitigated (although still with a high KL-divergence



(a) Historical Matching



(b) Future Matching



(c) Agreement

Figure (6.2) Distribution bias: (a) user's historical preference $\mathbf{Past}^t$ vs. the recommended preference $\mathbf{Rec}^t$, (b) user's real future preference $\mathbf{Future}^t$ with recommended preference $\mathbf{Rec}^t$, and (c) user's agreement with the recommended preference $\mathbf{Rec}^t$.

> 1.3). However, such a learning process could narrow down the diversification of recommendation and ignore the dynamic shifts in a user's preference distribution. This may also cause rabbit holes or echo chamber effects, as studied in many related works.

Figure 6.2 also illustrates the distribution bias induced by the distribution-aware recommender, CaliRec [18]. To better demonstrate the effectiveness of CaliRec, we choose two settings with different calibration weights ($\lambda = 0.25$ and $\lambda = 0.50$). As we can see, as the main objective of CaliRec, the distribution bias between the recommendations and the user's historical interactions ($\mathbf{Rec}^t$ vs. $\mathbf{Past}^t$) is dramatically reduced. Specifically, after a slight growth in the first 1,000 interactions. $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$ continues to fall and sustains around 0.3 after 5,000 iterations, which is much lower than the results from MF (blue line). It is no surprise that CaliRec achieves this result as this mitigation of distribution bias is its fundamental design. Up to now, the results from CaliRec are better than MF in the first comparison ($\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$), indicating that a conventional distribution-aware recommender can achieve its primary goal. In the following, we will analyze the results from the other two perspectives.

6.2.1.2 Current Recommendation vs. Future: $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$

How does the current recommended preference distribution match the user's real future distribution? A large $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$ indicates the presence of distribution bias between the recommendation list and a user's evolving interests [143]. Such evaluation has been ignored by traditional distribution-aware recommenders that has been proposed in a static setting. Comparing with the future ground truth is an oracle-like perspective, and invisible to the system in practice. However, this comparison may reveal the presence of distribution bias between the recommendation list and a user's evolving interests that is not captures by traditional distribution-aware recommenders. As shown in Figure 6.2 (b), we observe that the distribution bias between the recommended preference ($\mathbf{Rec}^t$) and users' real future preference ($\mathbf{Future}^t$) by MF (blue line) stays on a lower level (less than 1) in the entire observation window with a slightly increase. The increase indicates that the recommendation is farther away from users' actual preferences in the future.

Will the recommendations from a distribution-aware approach reveal less distribution bias over the future preferences of a user? To compare the current recommendation ($\mathbb{E}^t$) with a user's future ground truth, Figure 6.2 (b) presents disappointing outcomes comparing with Figure 6.2 (a) in the previous section. Specifically, comparing the distribution of the current recommendation ($\mathbf{Rec}^t$) with users' real future preference ($\mathbf{Future}^t$), CaliRec gives a much worse estimation of the distribution matching than the accuracy-driven recommender MF. With a large $\lambda = 0.50$, the $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$ for CaliRec is almost $3\times$ the results by MF, which indicates the distribution-aware recommendation is farther from a user's real preferences in the future, even comparing with MF which completely ignores such distributional concerns. In other words, the distribution-aware recommender does not adapt well to the change in preference distribution, leading to an emphasis on outdated preferences as the system evolves.

6.2.1.3 User Agreement with Current Recommendations: $\varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$

Next, we turn to how well the user agrees with the preference distribution provided by the current recommendation. We measure agreement here by actual user engagement (e.g., by clicks or likes). Unlike the preference distribution in the current recommendation ($\mathbf{Rec}^t$), the preference distribution over the real clicks ($\mathbf{Click}^t$) from this recommendation can more accurately reflect a user's real preferences at this stage. Furthermore, because the iterative interactions must be chosen from the recommendation list (i.e., $\mathbb{C}_u^t \subseteq \mathbb{E}_u^t$), the preference distribution of a user's real clicks could reflect how well the user agrees with the recommended preference distribution. Figure 6.2 (c) shows this "agreement" from users to the recommended preferences. As we can see, after a quick drop in the beginning, users increasingly disagree with the provided preference distribution by the system. Surprisingly, CaliRec provides even worse results. As we can see, the degree of disagreement ($\varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$) by CaliRec has not only larger magnitude (more than 40% worse) than MF, but also faster growth rate. These findings indicate that both traditional accuracy-driven recommenders and distribution-aware recommenders may not provide a satisfying estimation of user preferences.

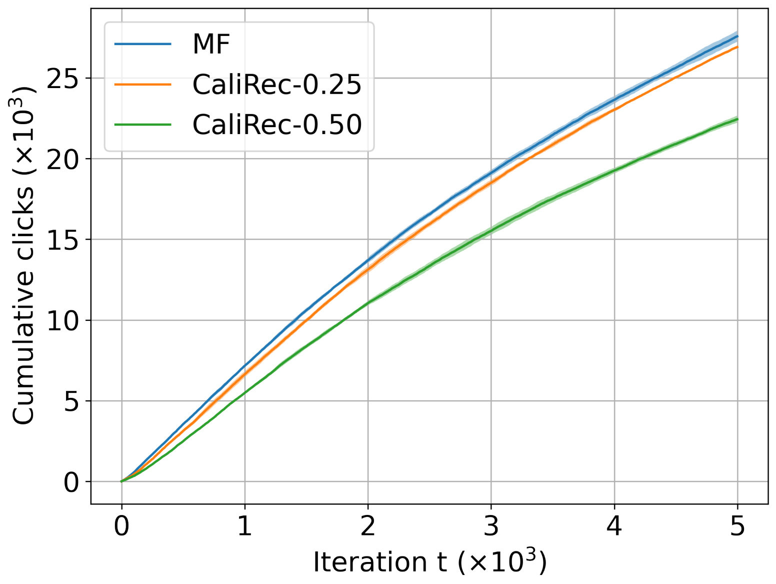


Figure (6.3) Recommendation utility: cumulative clicks.

6.2.1.4 Recommendation Utility

Finally, we examine the trade-off between recommendation utility and distribution matching. This trade-off has been observed in the static setting [18, 24, 77, 143], but is it also present in dynamic recommendations? In Section 6.2.1.2, we found that the preference distribution from the conventional distribution-aware recommendation is far away from users' real preference in the future, and users increasingly disagree their recommended preference in each iteration. Therefore, understandably, CaliRec would bring worse recommendation utility during the dynamic process. We show users' cumulative user-item clicks from each iteration in Figure 6.3. Compared with the results from the accuracy-driven recommender (MF), CaliRec results in fewer clicks during the entire observation window. Besides, the recommendation utility is even less as the calibration weight (λ) increases. Specifically, there is a 2.4% drop with $\lambda = 0.25$ and a 18.6% drop with $\lambda = 0.50$ from MF results at the end of 5,000 iterations. NDCG is also suffering from the calibration. As a result, NDCG has fallen 2.6% with $\lambda = 0.25$ and 19.7% with $\lambda = 0.50$ by CaliRec comparing with the results from MF.

6.2.2 Summary and Expectation

In sum, Figure 6.2 captures the different distribution bias issues that arise in dynamic recommendation for both accuracy-driven recommenders and distribution-aware recommenders. These results suggest an opportunity to improve upon the mitigation of distribution bias in dynamic recommendation. We propose that such an approach should be (1) aware of the user's prior preference to avoid the rich-get-richer effect or to narrow down users' interest (i.e., a lower $\varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$), (2) capture user's real preference in the future and try to recommend matching this preference (i.e., a lower $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$), and (3) most importantly, such recommendation should maintain sufficient accuracy to enable user agreement with this the preference provided (i.e., a lower $\varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$).

6.3 $\mathbf{DisCo}$: A Distribution-Correction Approach

In this section, we introduce a new approach called $\mathbf{DisCo}$ – Dynamic **Dis**tribution **Co**rrection – for mitigating distribution bias in dynamic recommendations. $\mathbf{DisCo}$ is motivated by four key research goals following the analysis in the previous section:

- RG0. Reflect user's historical preference into recommendation – $\Downarrow \varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$
- RG1. Predict users' future preference and reflect it in the recommendation – $\Downarrow \varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$
- RG2. Stimulate user agreement with the recommended preference – $\Downarrow \varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$
- RG3. Improve recommendation utility by mitigating distribution bias – $\Uparrow \mathbf{Utility}$

Given that conventional distribution-aware recommenders already achieve the goal of matching user's historical preference (RG0), we focus on RG1, RG2, and RG3 as motivations to introduce our proposed approach. Concretely, $\mathbf{DisCo}$ is characterized by three key contributions: (i) first, we introduce a distribution-matching component to incorporate each user's historical preference distribution into the recommendation, so that $\mathbf{DisCo}$ is fundamentally distribution-aware (Section 6.3.1); (ii) since our data-driven study showed how naively matching distributions to can raise other preference distortion problems, we next introduce a false-positive preference correction component to better model preference distributions in a dynamic environment (Section 6.3.2); and (iii) finally, we incorporate these two components into a distribution-aware re-ranking mechanism to dynamically correct for distribution bias at each step of the recommendation loop (Section 6.3.3).

6.3.1 Distribution Matching

The initial goal RG0 ($\Downarrow\ \varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$) motivates us to adopt a user's prior preference distribution as a reference. Inspired by conventional distribution-aware recommenders (e.g., [18, 143]), we aim to match the preference distribution of the current recommendation ($\mathbf{Rec}^t$) to a user's historical distribution ($\mathbf{Past}^t$) through a *distribution-matching* component (what we refer to as DM in the following). But how can we enable this distribution matching between the past and the current recommendations? A natural idea is to directly measure the distance between two distributions using KL-divergence and use this distance to drive the distribution matching. However, directly using KL-divergence to find the optimal set is a combinatorial optimization problem and NP-hard [18]. Prior research [137] has shown that the greedy optimization of this problem could be equivalent to the greedy optimization of a surrogate sub-modular function. Hence, we define $\varphi'(\cdot)$ as a step-by-step distribution-closeness measure derived from the original Kl-divergence function that supports efficient computation. The derivation is shown below:

$$\begin{aligned}
\varphi'\Big(\mathbf{Rec}^t||\mathbf{Past}^t\Big) &= -\sum_{g\in\mathcal{G}}\mathbf{Rec}_g^t\log\frac{\mathbf{Past}_g^t}{\mathbf{Rec}_g^t}\\
&= \sum_{g\in\mathcal{G}}\mathbf{Rec}_g^t\log\mathbf{Rec}_g^t - \sum_{g\in\mathcal{G}}\mathbf{Rec}_g^t\log\mathbf{Past}_g^t\\
&= \underbrace{\text{Entropy}(u)}_{\text{constant}} - \sum_{g\in\mathcal{G}}\mathbf{Rec}_g^t\log\mathbf{Past}_g^t\\
&\simeq -\sum_{g\in\mathcal{G}}\mathbf{Rec}_g^t\log\mathbf{Past}_g^t
\end{aligned} \tag{6.7}$$

where $\mathbf{Past}_g^t$ and $\mathbf{Rec}_g^t$ are defined in Eq. (6.2) and Eq. (6.4), respectively. Entropy(u) refers to the current recommended preference distribution, which is a constant once a recommendation is provided and will not reflect the closeness between the recommended preference distributions and user's historical ones. Hence, based on the original Eq. (6.6), we use $\varphi'(\cdot)$ in each iteration as the distribution matching component to re-rank the recommendation list.

6.3.2 False-Positive Correction

This distribution matching component aims to match the user's historical preference distribution in the current recommendations. However, solely relying on such historical preferences may not reflect the future interests of the user as we highlighted in the data-driven study in Section 6.2. Recall that we also aim to match a user's future preferences (RG1 $\Downarrow \varphi(\mathbf{Future}^t || \mathbf{Rec}^t)$), leading to better matching of user engagement (RG2 $\Downarrow \varphi(\mathbf{Click}^t || \mathbf{Rec}^t)$), and potentially providing more accurate recommendations (RG3 $\Uparrow \mathbf{Utility}$). RG1 and RG2 motivate us to "guess" a user's preference distribution in the future. Therefore, we introduce the second component of our approach, false-positive correction, which is referred to FPC in the following.

First, we define a binary *false positive signal*, $\Gamma \in \{0, 1\}$, as an indicator that item i was exposed to user u but the user did not interact with it (e.g., click or play), defined as below:

$$\Gamma^t_{u,i} = \begin{cases} 1, & \text{if } i \in \mathbb{E}^t_u \text{ and } i \notin \mathbb{C}^t_u; \\ 0, & \text{Otherwise.} \end{cases} \tag{6.8}$$

$\Gamma \in \{0, 1\}$ is a strong indicator whether we should penalize a recommendation or not. From the perspective of items, we should penalize a recommendation i to u if $\Gamma_{u,i} = 1$. In this work, we use this indicator to check whether a genre g_j should be penalized if many items belong to this genre but most have $\Gamma_{u,i} = 1$.

Next, we define the main factor of our FPC component, $\mathcal{F}_{u,g_j}$ – a cumulative degree of *dislikes* from user u to genre g_i, defined as below:

$$\mathcal{F}_{u,g_j} = \sum_{t=0}^{T} \left(\rho^{T-t} \times \sum_{i \in \mathbb{E}^t_u, k \in [1, |\mathbb{E}^t_u|]} \Gamma^t_{u,i} \times \mathbf{c}_{i,g_j} \times \delta_k \right). \tag{6.9}$$

Here, we first introduce a temporal discount factor, $\rho \in [0, 1]$, to reduce the impact from users' preference long ago. ρ^{T-t} is steadily declining while $t \to 0$. Next, we check the impact of preference distribution from users' activity in each iteration. For each item $i \in \mathbb{E}^t_u$ recommended by the system, if the *false positive signal* is 1, we should give a penalty to the genre that item i belongs to. For example, if a system continually recommends Sci-Fi movies (which may be liked by u a long time ago) to u, but most of the recommendations are poor ($\Gamma^t_{u,i} = 1$), that indicates that Sci-Fi movies have been over-represented. Hence, we should lessen recommendations of this genre. Given the truth that item i may belong to several genres, we penalize each genre according to their weight, $\mathbf{c}_{i,g_j}$.

Last but not least, we consider the position bias in this correction formula as well. Many research studies indicated the probability to examine items at top-ranking positions is higher than at lower positions in the recommender system [144, 66]. Here we more heavily penalize false-positive genres ranked at the top of the recommendation list by using the position bias $\delta_k = \frac{e}{\log_2(1+k)}$, where k is the position and e is Euler's number. Up to now, all item with $\Gamma_{u,i} = 1$ contribute equally to the degree of dislike $\mathcal{F}_{u,g_j}$. However, we should also consider an item-wise weight to differentiate the impact for each item $\Gamma_{u,i} = 1$. Similar to $\mathcal{F}_{u,g_j}$, we define $\omega_{u,i}$ as the dislike degree from user u to item i as below:

$$\omega_{u,i} = \sum_{t=0}^{T} \rho^{T-t} \times \Gamma^t_{u,i} \times \delta_k \tag{6.10}$$

where $\omega_{u,i}$ is the item-wise weight to impact the degree of *dislike* from u to a genre g_j. That is to say, given two items i and j with exactly the same genre vector ($\mathbf{c}_i = \mathbf{c}_j$), if i has been recommended more times and u did not interact with it, we should penalize the genre (which i and j belonging to) more from the behavior of recommending i than from the behavior of recommending j. We apply this weight and use $\omega_{u,i} \times \mathcal{F}_{u,g_j}$ as the final penalty.

Now, we already have user's degree of *dislike* from a user u to a genre g_j. We will now reflect this preference (*dislike*) into each individual item i. We finally define our FPC component, $\mathcal{C}$, as below:

$$\mathcal{C}_{u,i} = \omega_{u,i} \sum_{g_j \in \mathcal{G}} \mathcal{F}_{u,g_j} \tag{6.11}$$

Algorithm 3: DisCo Re-ranking Process

```
Input: $u$, $\mathbf{D}^t$, $\mathbf{Past}_u^t$, $\lambda$, $\alpha$, $\beta$, $K$
Output: $\mathbb{T}_u^t$ ;                                    // Recommendation for $u$ from DisCo
$\mathbb{T}_u^t \leftarrow \varnothing$;
$\mathbb{E}_u^t =$ item list sorted by predicted user-item relevance in $t$;
while $|\mathbb{T}_u^t| < K$ do
    $i^* \leftarrow$
    $\arg\max_{i \in \mathbb{E}_u^t \setminus \mathbb{T}_u^t} \Big[ \underbrace{(1-\lambda) \times \sum_{v \in \mathbb{T}_u^t \cup \{i\}} \frac{\mathbf{D}_{u,v}}{|\mathbb{T}_u^t|+1}}_{\text{Accuracy component}} + \underbrace{\alpha \times \varphi'\Big(\mathbf{Past}_u^t, \sum_{v \in \mathbb{T}_u^t \cup \{i\}} \mathbf{c}_v\Big)}_{\text{Distribution Matching Component}} + \underbrace{\beta \times \sum_{v \in \mathbb{T}_u^t \cup \{i\}} \frac{\mathcal{C}_{u,v}}{|\mathbb{T}_u^t|+1}}_{\text{False-Positive Correction Component}} \Big]$ ;
    $\mathbb{E}_u^t \leftarrow \mathbb{E}_u^t \setminus \{i^*\}$;
    $\mathbb{T}_u^t \leftarrow \mathbb{T}_u^t \cup \{i^*\}$ ;                    // Update optimal result
end
return $\mathbb{T}_u^t$;
```

6.3.3 Re-Ranking with Distribution Correction

Finally, we incorporate both distribution matching and false-positive correction into a distribution-aware re-ranking process that can be applied to any existing recommendation algorithm. That is to say, for the recommendation list provided by the system from each iteration, we re-rank the recommendation list by adding the corrections. Such re-ranking follows a long line of related work [24, 18, 113, 114, 143] and can be deployed for existing approaches without a need to modify production or legacy code. In this setting, users would examine and click from the re-ranked results and provide feedback to the system; therefore, the re-ranking process would impact the system training in the closed feedback loop. We summarize this re-ranking process with distribution correction in Algorithm 3.

In Algorithm 3, we first initialize a new list $\mathbb{T}_u^t$ as the final recommendation in iteration t (line 1). From the original recommendation result, we sort items according to the predicted user-item relevance score as the candidate pool $\mathbb{E}_u^t$ (line 2). Next, we iteratively choose the optimal item to feed into $\mathbb{T}_u^t$, by calculating the corrected relevance score (line $4-6$). λ, α, and β are the hyper-parameters to control the contribution of relevance component, DM component, and FPC component, respectively. We then delete the optimal item i^* from the candidate pool $\mathbb{E}_u^t$ and feed it into $\mathbb{T}_u^t$. We continue this process (line $3-8$) until we find K optimal items to recommend. Specifically, to balance the contribution from each component, we define the relationship of each hyper-parameter weight as $\lambda = \alpha + \beta$. In this way, in the following section, we can fairly compare our approach with the baseline method based on the same weight of the relevance score (λ).

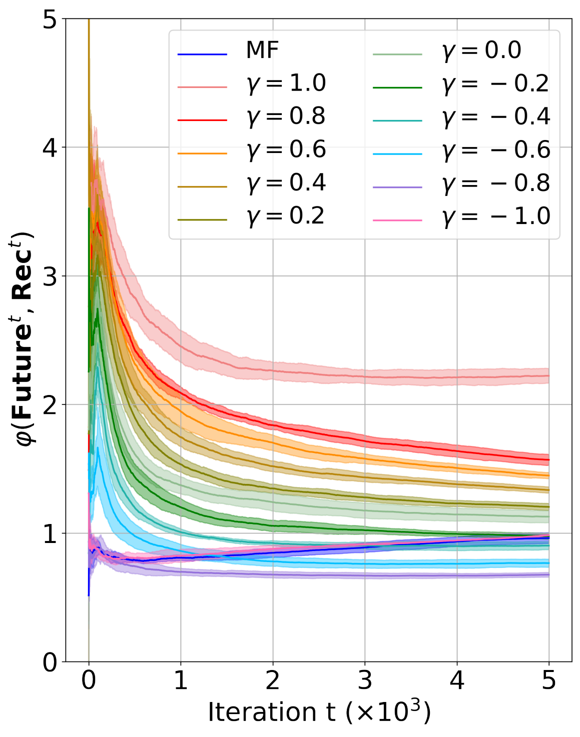
6.4 Experiments and Results

In this section, we first investigate the impact of hyper-parameters of **DisCo**. Secondly, we evaluate the effectiveness of the proposed method for different settings of the calibration weight. Then, we compare **DisCo** with accuracy-driven recommenders and distribution-aware recommenders, corresponding to our four research goals listed in Section 6.3.

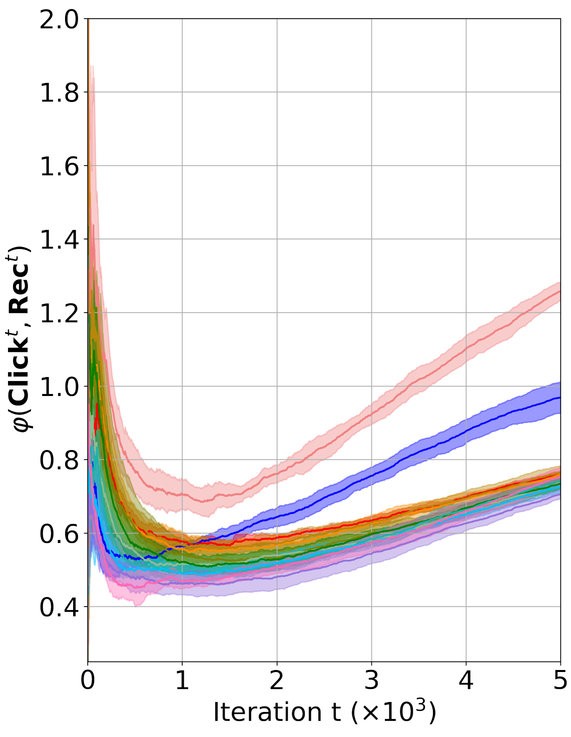
6.4.1 Hyper-parameter Tuning

The two main components of $\mathbf{DisCo}$ are the DM component controlled by α and the FPC controlled by β. To balance the contribution and effectiveness of these two components, we define $\gamma = \frac{\alpha-\beta}{\lambda}$ then analyze the relationship between these two knobs. The value of $\gamma \in [-1, 1]$ given that $\lambda = \alpha + \beta$. Specifically, $\gamma = -1$ means we only consider FPC and ignore the DM, $\gamma = 0$ means two components equally contribute to the final ranking, and $\gamma = 1$ means we only consider the DM.

First, we focus on RG1 and check which setting of γ could bring the best preference distribution closeness to a user's future preferences. Figure 6.4 (a) shows the distribution closeness between a user's future preferences with the preference distribution of the iterative recommendation. Again, $\gamma = 1.0$ indicates that we do not take the FPC component, which is similar to a traditional distribution-aware recommender (e.g., CaliRec). As we can see, with the falling of γ from 1 to -0.8, the preference distribution of iterative recommendation is closer to a user's real preferences in the future. However, at the extreme case of $\gamma = -1.0$, which means we ignore the matching of historical preference at all, $\varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$ is worse than some larger γ settings (i.e., $\gamma =$ -0.4, -0.6, or -0.8). Besides, comparing with the result from accuracy-driven recommender MF, we observe better results after 1,500 iterations when setting γ as -0.6 or -0.8.

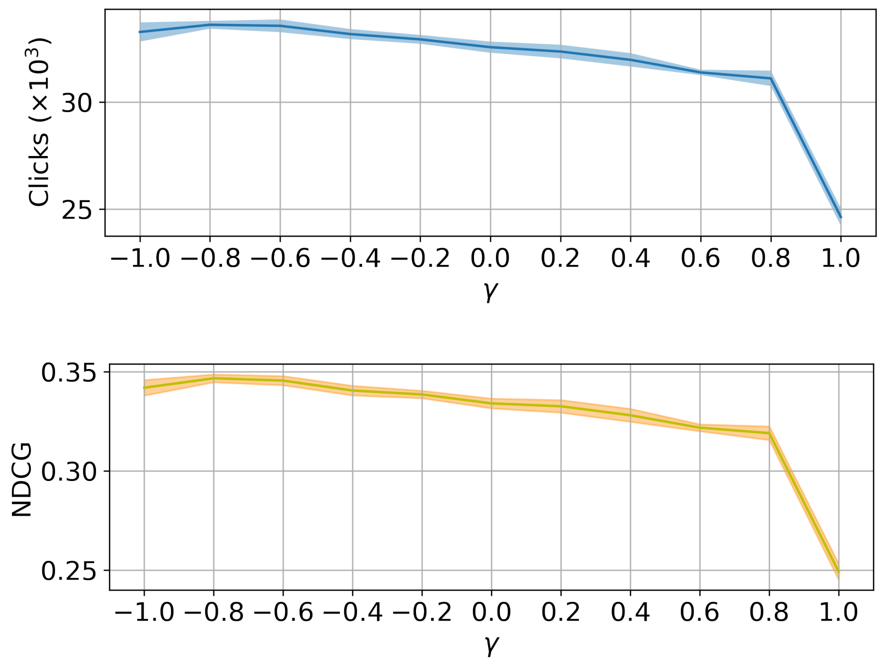


(a) Future



(b) Agreement

Figure (6.4) Distribution bias between users' real future preference with the current recommended preference (left) and users' interactive agreement of the recommended preference (right) by different settings of γ.



‘

Figure (6.5) Recommendation utilities on different settings of γ.

Next, we move to RG2. How do users agree with the iterative recommended preference distribution? In Figure 6.4 (b), we observed that when $\gamma = 1$, the degree of agreement is worse than the result from MF. With the decline of γ from 0.8 to -0.8, this degree is getting better (although without significant difference), and the degree of agreement is slightly worse when $\gamma = -1$ than $\gamma =$ -0.4, -0.6, or -0.8.

Then, we focus on RG3 and check how recommendation utility will change different values of γ. In Figure 6.5, we observed that the recommendation utility (both cumulative clicks and NDCG) is getting better with the growth of γ at the beginning. After reaching a peak (clicks at $\gamma = -0.6$ and NDCG at $\gamma = -0.8$), recommendation utility then declines. In the extreme case, when $\gamma = 1$, we see the dramatic drop of cumulative clicks and NDCG. Recall the preference distribution is getting closer to the user's real future preferences with the decline of γ; therefore, in the following section, we chose $\gamma = -0.6$ as our hyper-parameter setting.

6.4.2 Calibration Knob

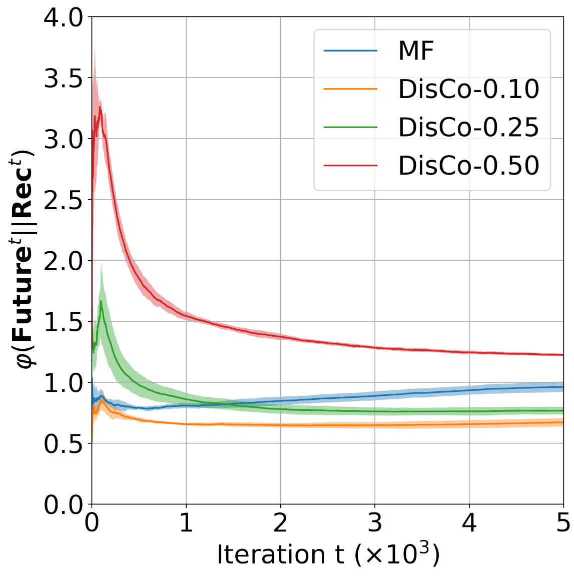
Similar to conventional distribution-aware recommendations, $\mathbf{DisCo}$ relies on several "knobs" (λ, α, and β) to control the degree of calibration. In Section 6.4.1 we discussed the relationship among these three knobs, and found the optimal proportion among them ($\gamma = -0.6$ means $\alpha = 0.2 \times \lambda$ and $\beta = 0.8 \times \lambda$). Therefore, in this section, we analyze the results for the approach with different settings of λ.

First, we focus on the distribution closeness with the real future data – one of our important criteria. Figure 6.6 (a) shows the distribution closeness between the preference distribution of a user's future ground truth $\mathbf{Future}^t$ with the recommended distribution $\mathbf{Rec}^t$ in the current iteration t with different settings of λ. With the growth of λ, the preference distribution of iterative recommendation is farther away from the real future distribution. Specifically, when $\lambda = 0.50$, the distribution closeness from $\mathbf{DisCo}$ is even worse than the results from MF. Results with $\lambda = 0.10$ are slightly better than the results with $\lambda = 0.25$ in the whole observation period, and both of them are better than results from MF after 1,400 iterations. Figure 6.6 (b) shows user's degree of agreement to the recommended preference distribution. Different from the result shown in (a), $\mathbf{DisCo}$ with all three settings of λ have better agreement than MF, and $\mathbf{DisCo}$ with $\lambda = 0.10$ and $\lambda = 0.25$ have similar results.

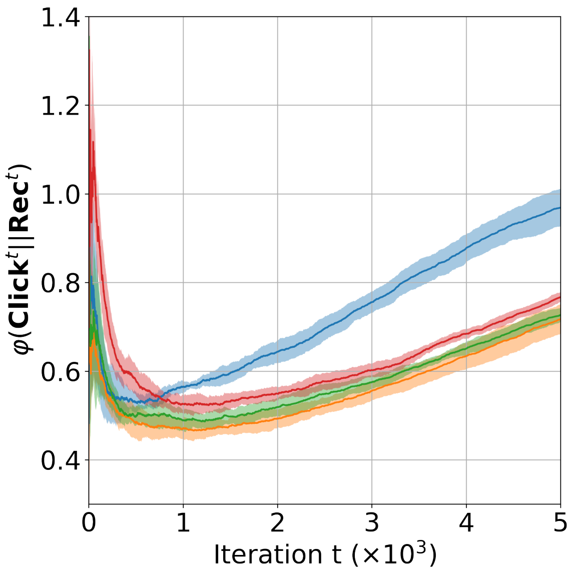
Next, we focus on the recommendation utility. Figure 6.7 shows the cumulative clicks using MF and $\mathbf{DisCo}$ with different settings of λ. The results with $\lambda = 0.10$ and 0.25 are very similar during the entire observation period, and slightly better than the result from $\mathbf{DisCo}$ with $\lambda = 0.50$; as we expected, results from all three settings are better than the result from MF.

6.4.3 Benchmark Results

In the following, we compare the results from the proposed $\mathbf{DisCo}$ approach with an accuracy-driven recommender (MF) and the conventional distribution-aware recommender (CaliRec) under the same configuration of calibration (λ).



(a) Future



(b) Agreement

Figure (6.6) Distribution bias between users' real future preference with the current recommended preference (left) and users' interactive agreement of the recommended preference (right) by **DisCo** with different settings of λ.

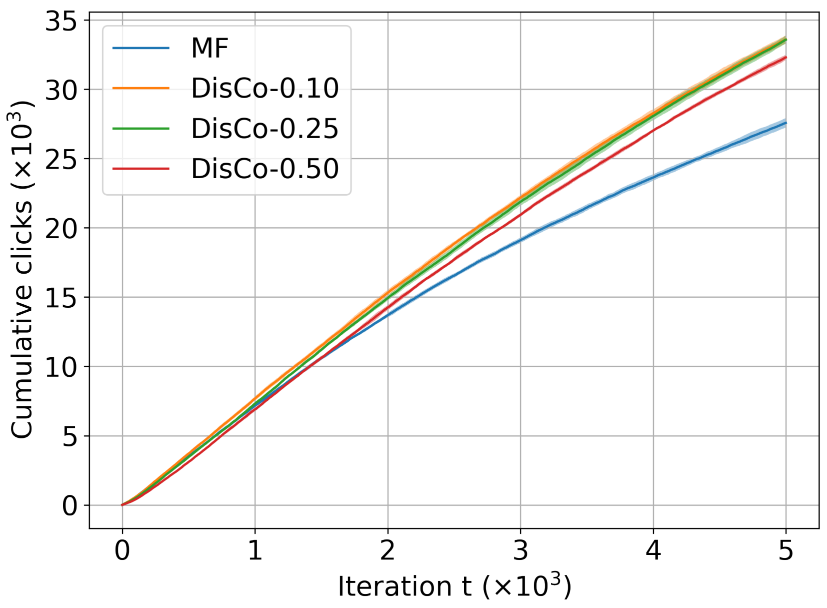
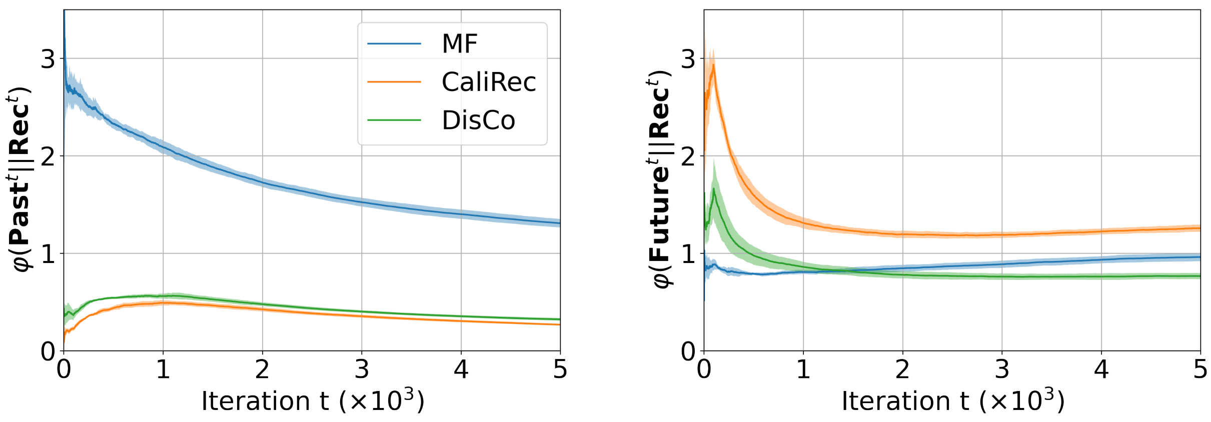
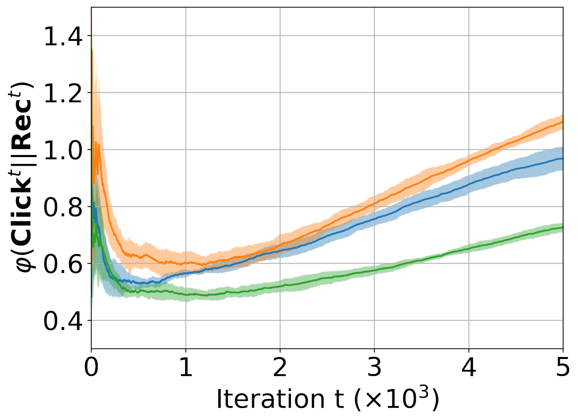


Figure (6.7) Recommendation utility – cumulative clicks by **DisCo** with different settings of λ.



(a) Historical Matching

(b) Future Matching



(c) Agreement

Figure (6.8) Preference distortion benchmark: (a) user's historical preference $\mathbf{Past}^t$ vs. the recommended preference $\mathbf{Rec}^t$, (b) user's real future preference $\mathbf{Future}^t$ with recommended preference $\mathbf{Rec}^t$, and (c) user's agreement of the recommended preference $\mathbf{Rec}^t$.

6.4.3.1 *Appetizer: RG0. Reflect user's historical preference into recommendation –* $\Downarrow \varphi(\mathbf{Past}^t||\mathbf{Rec}^t)$

Although matching the users' historical preference is not our primary purpose, we still start to analyze the closeness given that our DM component targets this goal. Figure 6.8 (a) illustrates the closeness between users' historical preference and the current recommendation's preference distribution. As we can see, comparing with the results from MF, both distribution-aware methods mitigate this distribution bias, where CaliRec performs slightly better than $\mathbf{DisCo}$. It is acceptable that $\mathbf{DisCo}$ loss a little in this comparison, since our three primary goals are matching users real preference in the future (RG1), let user agree with the recommended preference (RG2), and improve the recommendation utility taking benefits from this distribution correction (RG3). Next, we will focus on these three research goals and analyze $\mathbf{DisCo}$ results with baselines.

6.4.3.2 RG1. Predict users' future preference and reflect it in the recommendation – $\Downarrow \varphi(\mathbf{Future}^t||\mathbf{Rec}^t)$

Figure 6.8 (b) shows the distribution closeness between users' real future preference with the iterative recommendation. Comparing with the results from CaliRec, $\mathbf{DisCo}$ presents a much better distribution matching in the entire observation period. Furthermore, comparing with the results from MF, $\mathbf{DisCo}$ also have better distribution estimations after around 1,000 iterations. In addition, unlike the growing trend from distribution bias of users' click by MF and CaliRec, $\mathbf{DisCo}$ gives steady level of this bias. We discussed the difficulty of estimating users' future preferences in the dynamic environment. Therefore, such strong preference estimation results bring us great confidence in enabling user agreement with the recommended preference distribution and achieving better recommendation utility.

6.4.3.3 RG2. Stimulate user agreement with the recommended preference – $\Downarrow \varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$

Figure 6.8 (c) shows how users agree with the recommended preference distribution by comparing a user's real clicks in each iteration with the iterative recommended items. As we observed in Section 6.2, accuracy-driven recommenders are not designed with distribution bias in mind, so may raise many problems such as echo chambers and rabbit holes. Therefore, a user's disagreement with the recommended preference distribution would become more serious every iteration. Conventional distribution-aware recommenders even amplify this disagreement and provide even more erroneous recommendations (larger $\varphi(\mathbf{Click}^t||\mathbf{Rec}^t)$). However, As shown in Figure 6.8 (c), $\mathbf{DisCo}$ greatly mitigates this bias all the time, and eventually improved agreement more than 25% from MF and more than 34% from CaliRec.

	ML-100K		Amazon-CD	
	#click	NDCG	#click	NDCG
Δ **MF**	+6004 (+21.7%)	+ 0.0660 (+23.6%)	+1153 (+16.4%)	+0.0081 (+9.47%)
Δ **CaliRec**	+6668 (+24.8%)	+0.0729 (+26.7%)	+2000 (+32.4%)	+0.0139 (+17.6%)
Δ $\mathbf{DisCo_{no-FPC}}$	+8941 (+36.3%)	+0.0962 (+38.6%)	+2078 (+34.1%)	+0.0115 (+14.2%)
Δ $\mathbf{DisCo_{no-DM}}$	+288 (+0.865%)	+0.0035 (+1.05%)	+1370 (+20.6%)	+0.0097 (+11.6%)

Table (6.2) Difference between Recommendation Utility of $\mathbf{DisCo}$ and Other Benchmark Approaches

6.4.3.4 RG3. Improve recommendation utility by mitigating distribution bias – $\Uparrow$ $\mathbf{Utility}$

Table 6.2 shows the improvement of cumulative clicks and average NDCG after 5,000 iterations by $\mathbf{DisCo}$ from 4 baselines. Baselines include the accuracy-driven recommender MF, conventional distribution-aware recommender CaliRec, as well as two variants of $\mathbf{DisCo}$: $\mathbf{DisCo_{no-FPC}}$ without FPC component ($\beta = 0$), and $\mathbf{DisCo_{no-DM}}$ without DM component ($\alpha = 0$). From the table, we find that with the same level of calibration weight ($\lambda = 0.25$), $\mathbf{DisCo}$ can generate significantly more clicks than two conventional methods: after 5,000 iterations, 21.7% and 16.4% more clicks are received on average for ML-100K and Amazon-CD dataset, respectively, comparing with MF, and these improvements even reach 24.8% and 32.4% comparing with CaliRec. We have similar observations when we compare with NDCG. These results suggest $\mathbf{DisCo}$ dashed the stereotype that there is always a trade-off between calibration and recommendation. The comparison between $\mathbf{DisCo}$ with $\mathbf{DisCo_{no-FPC}}$ and $\mathbf{DisCo_{no-DM}}$ suggests the importance of simultaneously considering the previous preference distribution and carefully correcting for future preferences.

6.5 Summary

In this chapter, we deepened our understanding of distribution bias by conducting the first study of this bias in a dynamic recommendation scenario. We first identified the preference distribution biases in dynamic recommendation, and empirically showed the prevalence of these biases by both accuracy-driven recommenders and conventional distribution-aware recommenders through a data-driven study. Then, we proposed a Distribution Correction Recommender (**DisCo**) that dynamically corrects the preference distribution in interactive recommendations. Results show that **DisCo** improves the closeness of the preference distribution in the recommendation with users' real future preference, and increases users' agreement with the preference distribution of the recommendations. Results also indicate that the recommendation quality can be improved compared with accuracy-driven recommenders and conventional distribution-aware recommenders, giving more evidence that there is not always a trade-off between calibration and recommendation.

7. CONCLUSIONS AND FUTURE WORK

In this chapter, we present the conclusion of this dissertation and potential future research opportunities.

7.1 Conclusions

With the increasing ubiquity of bias in recommendation results, correspondent de-biasing approaches need more attention. Hence, this dissertation is first grounded in a holistic conceptual framework for identifying bias in recommender systems from multiple perspectives. This dissertation identifies gaps in the literature for identifying and addressing bias in recommender systems through this conceptual framework. It then couples this conceptual framework with four new methods for mitigating bias in recommender systems from different perspectives.

First, to mitigate the estimation bias for items with different ratings, we proposed the MLR recommender system, which could improve the estimation of tail ratings with multi-latent representation. We conducted a data-driven investigation and theoretical analysis of the challenges posed by conventional methods for estimating tail ratings, which lead to over- and under-estimations of tail ratings, with particularly pronounced errors on controversial items. Our proposed MLR is explicitly designed to estimate these tail ratings better. Experimental results show the estimation improvement is especially great for those items far from the ratings mean. Furthermore, the proposed model is generalizable and can be easily extended to take advantage of other conventional models.

Second, to mitigate the distribution bias of predicted target costumers, we proposed a target customer re-ranking algorithm – U2I-CaliRec, for addressing the target customer distortion problem. We began with a look at the taste distortion problem through analysis over all three datasets. Then we illustrated how our proposed method could effectively adjust the class distribution of the target customers for items toward the desired distribution, thereby mitigating the distortion problem.

Third, to mitigate the distribution bias of the predicted user's taste, we proposed a Taste-enhanced Calibrated Recommendation – TecRec. We first identified the taste distortion problem and empirically showed its prevalence through a data-driven study. Then, we proposed TecRec, which incorporates a time-series neural network sub-model to predict users' preference shifts. Results show TecRec improves both taste distribution estimation and recommendation quality, compared with conventional recommender systems. Besides, the proposed recommender offers interpretable taste distribution and can be integrated into existing recommenders without re-training the original recommendation pipeline.

Last but not least, we continue the studies of distribution bias and extended the application scenario from static recommendations to dynamic recommendations. We empirically showed the prevalence of these biases by both accuracy-driven recommenders and conventional distribution-aware recommenders through a data-driven study. Then, we proposed a Distribution Correction Recommender (**DisCo**) that dynamically corrects preference distribution in interactive recommendations. Results suggest **DisCo** improves the closeness of preference distribution in the recommendation with users' real future preference, and increases users' agreement of the recommended preference. Results also suggest that the recommendation quality has been improved compared with accuracy-driven recommenders and conventional distribution-aware recommenders, dashed the stereotype that there is always a trade-off between the calibration and recommendation.

7.2 Future Research Opportunities

Our conceptual framework of bias in the recommendation system still has many research opportunities and deserves more efforts. We identify three future research directions as follows:

- First, approaches are still missing to mitigate the bias between **User Activities** and **Appearance**, as presented in Figure 1.1. In Chapter 4, we analyzed the distribution bias (**Appearance**) of predicted target costumers with different demographics, e.g., gender and age. We can expand this simple classification to a broad-view, such as active and inactive users. For example, given the task of sending email advertisements with a limited number, does an active user have more chance to be appeared in this email list (being predicted as target customer)? This appearance bias is also related to the cold-start problem.

- Second, many research works incorporate shopping sessions as a consideration of accuracy-driven recommendations [145, 146, 147, 148]. Besides improving the recommendation utility, this shopping session information could also be incorporated into predicting uses' future taste in the dynamic recommendations to mitigate the taste distortion problem. More additional information, such as profiles of users and items, could also be employed to predict user's preferences in the future, mitigate the taste distribution bias, and avoid undesirable outcomes, e.g., echo chambers or rabbit holes.

- Third, there is a solid motivation to analyze the formation of bias problems in the recommendation result. In this dissertation, we analyzed the observed biases and proposed new de-biasing approaches. However, how are these biases generated in the first place? How can we avoid these biases from their source of generation? This "source control" brings us the great research opportunity to analyze the generative mechanisms of biases and build appropriate end-to-end bias-free recommender systems.

REFERENCES

[1] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio, "Folksonomies, the semantic web, and movie recommendation," 2007.

[2] G. Lekakos and P. Caravelas, "A hybrid approach for movie recommendation," *Multimedia tools and applications*, vol. 36, no. 1-2, pp. 55–70, 2008.

[3] T. Bogers, "Movie recommendation using random walks over the contextual graph," in *Proc. of the 2nd Intl. Workshop on Context-Aware Recommender Systems*, 2010.

[4] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, "Machine learned job recommendation," in *Proceedings of the fifth ACM Conference on Recommender Systems*, pp. 325–328, 2011.

[5] S. Yang, M. Korayem, K. AlJadda, T. Grainger, and S. Natarajan, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach," *Knowledge-Based Systems*, vol. 136, pp. 37–45, 2017.

[6] Y. Zhang, C. Yang, and Z. Niu, "A research of job recommendation system based on collaborative filtering," in *2014 Seventh International Symposium on Computational Intelligence and Design*, vol. 1, pp. 533–538, IEEE, 2014.

[7] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[8] A. V. Bodapati, "Recommendation systems with purchase data," *Journal of marketing research*, vol. 45, no. 1, pp. 77–93, 2008.

[9] D.-R. Liu and Y.-Y. Shih, "Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences," *Journal of Systems and Software*, vol. 77, no. 2, pp. 181–191, 2005.

[10] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in *Proceedings of the 15th international conference on Intelligent user interfaces*, pp. 31–40, 2010.

[11] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, pp. 167–176, 2018.

[12] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, pp. 1835–1844, 2018.

[13] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis, "User oriented trajectory search for trip recommendation," in *Proceedings of the 15th International Conference on Extending Database Technology*, pp. 156–167, 2012.

[14] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng, "Personalized trip recommendation with multiple constraints by mining user check-in behaviors," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pp. 209–218, 2012.

[15] C. Zhang, H. Liang, K. Wang, and J. Sun, "Personalized trip recommendation with poi availability and uncertain traveling time," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 911–920, 2015.

[16] T. G. Morrell and L. Kerschberg, "Personal health explorer: A semantic health recommendation system," in *2012 IEEE 28th International Conference on Data Engineering Workshops*, pp. 55–59, IEEE, 2012.

[17] A. K. Sahoo, S. Mallik, C. Pradhan, B. S. P. Mishra, R. K. Barik, and H. Das, "Intelligence-based health recommendation system using big data analytics," in *Big data analytics for intelligent healthcare management*, pp. 227–246, Elsevier, 2019.

[18] H. Steck, "Calibrated recommendations," in *Proceedings of the 12th ACM conference on recommender systems*, pp. 154–162, ACM, 2018.

[19] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors," *arXiv preprint arXiv:1702.06081*, 2017.

[20] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, pp. 3315–3323, 2016.

[21] M. Zhang and N. Hurley, "Avoiding monotony: improving the diversity of recommendation lists," in *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 123–130, ACM, 2008.

[22] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*, pp. 22–32, ACM, 2005.

[23] L. Chen, G. Zhang, and H. Zhou, "Improving the diversity of top-n recommendation via determinantal point process," in *Large Scale Recommendation Systems Workshop*, 2017.

[24] X. Zhao, Z. Zhu, M. Alfifi, and J. Caverlee, "Addressing the target customer distortion problem in recommender systems," in *Proceedings of The Web Conference 2020*, pp. 2969–2975, 2020.

[25] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *International Conference on Machine Learning*, pp. 1512–1520, PMLR, 2014.

[26] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," *arXiv preprint arXiv:1206.5267*, 2012.

[27] H. Steck, "Evaluation of recommendations: rating-prediction and ranking," in *Proceedings of the 7th ACM conference on Recommender systems*, pp. 213–220, 2013.

[28] H. 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*, pp. 713–722, 2010.

[29] J. Chen, C. Wang, S. Zhou, Q. Shi, Y. Feng, and C. Chen, "Samwalker: Social recommendation with informative sampling strategy," in *The World Wide Web Conference*, pp. 228–239, 2019.

[30] J. Ding, Y. Quan, X. He, Y. Li, and D. Jin, "Reinforced negative sampling for recommendation with exposure data.," in *IJCAI*, pp. 2230–2236, 2019.

[31] H.-F. Yu, M. Bilenko, and C.-J. Lin, "Selection of negative samples for one-class matrix factorization," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 363–371, SIAM, 2017.

[32] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, "Unbiased recommender learning from missing-not-at-random implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 501–509, 2020.

[33] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 42–46, 2017.

[34] J. Wasilewski and N. Hurley, "Incorporating diversity in a learning to rank recommender system.," in *FLAIRS Conference*, pp. 572–578, 2016.

[35] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Correcting popularity bias by enhancing recommendation neutrality.," in *RecSys Posters*, 2014.

[36] Z. Chen, R. Xiao, C. Li, G. Ye, H. Sun, and H. Deng, "Esam: Discriminative domain adaptation with non-displayed items to improve long-tail performance," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 579–588, 2020.

[37] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey, "An experimental comparison of click position-bias models," in *Proceedings of the 2008 international conference on web search and data mining*, pp. 87–94, 2008.

[38] G. E. Dupret and B. 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*, pp. 331–338, 2008.

[39] O. Chapelle and Y. Zhang, “A dynamic bayesian network click model for web search ranking,” in *Proceedings of the 18th international conference on World wide web*, pp. 1–10, 2009.

[40] J. Jin, Y. Fang, W. Zhang, K. Ren, G. Zhou, J. Xu, Y. Yu, J. Wang, X. Zhu, and K. Gai, “A deep recurrent survival model for unbiased ranking,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 29–38, 2020.

[41] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[42] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434, ACM, 2008.

[43] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, International World Wide Web Conferences Steering Committee, 2017.

[44] M. Lee, P. Choi, and Y. Woo, “A hybrid recommender system combining collaborative filtering with neural network,” in *International conference on adaptive hypermedia and adaptive web-based systems*, pp. 531–534, Springer, 2002.

[45] O. Barkan and N. Koenigstein, “Item2vec: neural item embedding for collaborative filtering,” in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2016.

[46] Y. Zheng, B. Tang, W. Ding, and H. Zhou, “A neural autoregressive approach to collaborative filtering,” *arXiv preprint arXiv:1605.09477*, 2016.

[47] A. Krishnan, A. Sharma, A. Sankar, and H. Sundaram, “An adversarial approach to improve long-tail performance in neural collaborative filtering,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1491–1494, ACM, 2018.

[48] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, “Autorec: Autoencoders meet collaborative filtering,” in *Proceedings of the 24th International Conference on World Wide Web*, pp. 111–112, ACM, 2015.

[49] L. Martinez, R. M. Rodriguez, and M. Espinilla, “Reja: a georeferenced hybrid recommender system for restaurants,” in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*, pp. 187–190, IEEE Computer Society, 2009.

[50] R. P. Biuk-Aghai, S. Fong, and Y.-W. Si, “Design of a recommender system for mobile tourism multimedia selection,” in *2008 2nd International Conference on Internet Multimedia Services Architecture and Applications*, pp. 1–6, IEEE, 2008.

[51] C. Matyas and C. Schlieder, “A spatial user similarity measure for geographic recommender systems,” in *International Conference on GeoSpatial Sematics*, pp. 122–139, Springer, 2009.

[52] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, “Aspect-aware latent factor model: Rating prediction with ratings and reviews,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 639–648, International World Wide Web Conferences Steering Committee, 2018.

[53] L. Ren, L. He, J. Gu, W. Xia, and F. Wu, “A hybrid recommender approach based on widrow-hoff learning,” in *2008 Second International Conference on Future Generation Communication and Networking*, vol. 1, pp. 40–45, IEEE, 2008.

[54] T. H. Roh, K. J. Oh, and I. Han, “The collaborative filtering recommendation based on som cluster-indexing cbr,” vol. 25, pp. 413–423, Elsevier, 2003.

[55] M. Kaya and D. Bridge, "Subprofile-aware diversification of recommendations," *User Modeling and User-Adapted Interaction*, vol. 29, no. 3, pp. 661–700, 2019.

[56] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, IEEE, 2018.

[57] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 191–200, IEEE, 2016.

[58] J. Ma, C. Zhou, H. Yang, P. Cui, X. Wang, and W. Zhu, "Disentangled self-supervision in sequential recommenders," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 483–491, 2020.

[59] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

[60] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[61] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 843–852, 2018.

[62] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 582–590, 2019.

[63] F. Yuan, X. He, H. Jiang, G. Guo, J. Xiong, Z. Xu, and Y. Xiong, "Future data helps training: Modeling future contexts for session-based recommendation," in *Proceedings of The Web Conference 2020*, pp. 303–313, 2020.

[64] A. J. Chaney, B. M. Stewart, and B. E. Engelhardt, "How algorithmic confounding in recommendation systems increases homogeneity and decreases utility," in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 224–232, 2018.

[65] S. Khenissi, B. Mariem, and O. Nasraoui, "Theoretical modeling of the iterative properties of user discovery in a collaborative filtering recommender system," in *Fourteenth ACM Conference on Recommender Systems*, pp. 348–357, 2020.

[66] M. Morik, A. Singh, J. Hong, and T. Joachims, "Controlling fairness and bias in dynamic learning-to-rank," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 429–438, 2020.

[67] W. Sun, S. Khenissi, O. Nasraoui, and P. Shafto, "Debiasing the human-recommender system feedback loop in collaborative filtering," in *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 645–651, 2019.

[68] G. Adomavicius and J. Zhang, "Impact of data characteristics on recommender systems performance," *ACM Transactions on Management Information Systems (TMIS)*, vol. 3, no. 1, p. 3, 2012.

[69] N. Hu, J. Zhang, and P. A. Pavlou, "Overcoming the j-shaped distribution of product reviews," *Communications of the ACM*, vol. 52, no. 10, pp. 144–147, 2009.

[70] P. Victor, C. Cornelis, M. De Cock, and A. M. Teredesai, "A comparative analysis of trust-enhanced recommenders for controversial items," in *Third International AAAI Conference on Weblogs and Social Media*, 2009.

[71] M. Badami, "Peeking into the other half of the glass: handling polarization in recommender systems.," 2017.

[72] J. O'Donovan and B. Smyth, "Trust in recommender systems," in *Proceedings of the 10th international conference on Intelligent user interfaces*, pp. 167–174, ACM, 2005.

[73] J. Golbeck, "Generating predictive movie recommendations from trust in social networks," in *International Conference on Trust Management*, pp. 93–104, Springer, 2006.

[74] P. Victor, C. Cornelis, M. De Cock, and P. P. Da Silva, “Gradual trust and distrust in recommender systems,” *Fuzzy Sets and Systems*, vol. 160, no. 10, pp. 1367–1382, 2009.

[75] P. Massa and P. Avesani, “Trust-aware recommender systems,” in *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 17–24, ACM, 2007.

[76] A. Beutel, E. H. Chi, Z. Cheng, H. Pham, and J. Anderson, “Beyond globally optimal: Focused learning for improved recommendations,” in *Proceedings of the 26th International Conference on World Wide Web*, pp. 203–212, International World Wide Web Conferences Steering Committee, 2017.

[77] M. Kaya and D. Bridge, “A comparison of calibrated and intent-aware recommendations,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 151–159, ACM, 2019.

[78] J. Kleinberg, S. Mullainathan, and M. Raghavan, “Inherent trade-offs in the fair determination of risk scores,” *arXiv preprint arXiv:1609.05807*, 2016.

[79] C.-E. Särndal, B. Swensson, and J. Wretman, *Model assisted survey sampling*. Springer Science & Business Media, 2003.

[80] J. E. Trost, “Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies,” *Qualitative sociology*, vol. 9, no. 1, pp. 54–57, 1986.

[81] G. W. Imbens and T. Lancaster, “Efficient estimation and stratified sampling,” *Journal of Econometrics*, vol. 74, no. 2, pp. 289–318, 1996.

[82] A. Bhaskara, M. Ghadiri, V. Mirrokni, and O. Svensson, “Linear relaxations for finding diverse elements in metric spaces,” in *Advances in Neural Information Processing Systems*, pp. 4098–4106, 2016.

[83] M. Gartrell, U. Paquet, and N. Koenigstein, “Bayesian low-rank determinantal point processes,” in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 349–356, ACM, 2016.

[84] A. Ashkan, B. Kveton, S. Berkovsky, and Z. Wen, "Diversified utility maximization for recommendations.," in *RecSys Posters*, 2014.

[85] N. Hurley and M. Zhang, "Novelty and diversity in top-n recommendation–analysis and evaluation," *ACM Transactions on Internet Technology (TOIT)*, vol. 10, no. 4, p. 14, 2011.

[86] M. Kaya and D. Bridge, "Accurate and diverse recommendations using item-based subprofiles," in *The Thirty-First International Flairs Conference*, 2018.

[87] R. L. Santos, C. Macdonald, and I. Ounis, "On the role of novelty for search result diversification," *Information retrieval*, vol. 15, no. 5, pp. 478–502, 2012.

[88] S. Vargas, P. Castells, and D. Vallet, "Intent-oriented diversity in recommender systems," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 1211–1212, 2011.

[89] J. Wasilewski and N. Hurley, "Intent-aware diversification using a constrained plsa," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 39–42, 2016.

[90] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, "Novelty and diversity in information retrieval evaluation," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 659–666, 2008.

[91] C. Zhai, W. W. Cohen, and J. Lafferty, "Beyond independent relevance: methods and evaluation metrics for subtopic retrieval," in *ACM SIGIR Forum*, vol. 49, pp. 2–9, ACM New York, NY, USA, 2015.

[92] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

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

[94] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 579–592, 2016.

[95] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, pp. 507–517, International World Wide Web Conferences Steering Committee, 2016.

[96] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016.

[97] J. Wang and J. Caverlee, "Recurrent recommendation with local coherence," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 564–572, ACM, 2019.

[98] K. Garimella, G. D. F. Morales, A. Gionis, and M. Mathioudakis, "Balancing opposing views to reduce controversy," *arXiv preprint arXiv:1611.00172*, 2016.

[99] D. J. Isenberg, "Group polarization: A critical review and meta-analysis.," *Journal of personality and social psychology*, vol. 50, no. 6, p. 1141, 1986.

[100] A. Matakos and P. Tsaparas, "Temporal mechanisms of polarization in online reviews," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 529–532, IEEE, 2016.

[101] L. Cai, Z. Wang, H. Gao, D. Shen, and S. Ji, "Deep adversarial learning for multi-modality missing data completion," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1158–1166, ACM, 2018.

[102] H. Zhang, Y. Yang, H. Luan, S. Yang, and T.-S. Chua, "Start from scratch: Towards automatically identifying, modeling, and naming visual attributes," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 187–196, ACM, 2014.

[103] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Advances in neural information processing systems*, pp. 2222–2230, 2012.

[104] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[105] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 4–pp, IEEE, 2005.

[106] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 471–475, SIAM, 2005.

[107] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[108] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, pp. 1257–1264, 2008.

[109] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pp. 452–461, AUAI Press, 2009.

[110] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, ACM, 2016.

[111] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: Multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 107–114, ACM, 2016.

[112] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1235–1244, ACM, 2015.

[113] D. P. Foster and R. V. Vohra, “Asymptotic calibration,” *Biometrika*, vol. 85, no. 2, pp. 379–390, 1998.

[114] B. Zadrozny and C. Elkan, “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers,” in *Icml*, vol. 1, pp. 609–616, Citeseer, 2001.

[115] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[116] J. G. Carbonell and J. Goldstein, “The use of mmr, diversity-based reranking for reordering documents and producing summaries.,” in *SIGIR*, vol. 98, pp. 335–336, 1998.

[117] D. Kanevsky, W. W. Zadrozny, and A. Zlatsin, “System and method for group advertisement optimization,” June 16 2009. US Patent 7,548,874.

[118] A. M. Petersen, W.-S. Jung, J.-S. Yang, and H. E. Stanley, “Quantitative and empirical demonstration of the matthew effect in a study of career longevity,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 1, pp. 18–23, 2011.

[119] T. Achakulvisut, D. E. Acuna, T. Ruangrong, and K. Kording, “Science concierge: A fast content-based recommendation system for scientific publications,” *PloS one*, vol. 11, no. 7, 2016.

[120] D. O’Callaghan, D. Greene, M. Conway, J. Carthy, and P. Cunningham, “Down the (white) rabbit hole: The extreme right and online recommender systems,” *Social Science Computer Review*, vol. 33, no. 4, pp. 459–478, 2015.

[121] Z. Zhu, X. Hu, and J. Caverlee, “Fairness-aware tensor-based recommendation,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1153–1162, ACM, 2018.

[122] W. Liu, J. Guo, N. Sonboli, R. Burke, and S. Zhang, “Personalized fairness-aware re-ranking for microlending,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 467–471, ACM, 2019.

[123] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation," *arXiv preprint arXiv:1907.13286*, 2019.

[124] V. Tsintzou, E. Pitoura, and P. Tsaparas, "Bias disparity in recommendation systems," *arXiv preprint arXiv:1811.01461*, 2018.

[125] S. Vargas, *Novelty and diversity evaluation and enhancement in recommender systems*. PhD thesis, Ph. D. Dissertation. Universidad Autónoma de Madrid, 2015.

[126] S. Dooms, T. De Pessemier, and L. Martens, "Movietweetings: a movie rating dataset collected from twitter," in *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, vol. 2013, p. 43, 2013.

[127] C.-M. Chen, M.-F. Tsai, Y.-C. Lin, and Y.-H. Yang, "Query-based music recommendations via preference embedding," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 79–82, 2016.

[128] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 152–160, 2017.

[129] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 495–503, 2017.

[130] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[131] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[132] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[133] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[134] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189–194, IEEE, 2000.

[135] F. A. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[136] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," in *Neural Nets WIRN Vietri-01*, pp. 193–200, Springer, 2002.

[137] Y. Shinohara, "A submodular optimization approach to sentence set selection," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4112–4115, IEEE, 2014.

[138] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[139] R. L. Santos, C. Macdonald, and I. Ounis, "Exploiting query reformulations for web search result diversification," in *Proceedings of the 19th international conference on World wide web*, pp. 881–890, 2010.

[140] B. Shi, G. Ifrim, and N. Hurley, "Learning-to-rank for real-time high-precision hashtag recommendation for streaming news," in *proceedings of the 25th international conference on World Wide Web*, pp. 1191–1202, 2016.

[141] S. U. Tareq, M. H. Noor, and C. Bepery, "Framework of dynamic recommendation system for e-shopping," *International Journal of Information Technology*, vol. 12, no. 1, pp. 135–140, 2020.

[142] P. Lopes and B. Roy, “Dynamic recommendation system using web usage mining for e-commerce users,” *Procedia Computer Science*, vol. 45, pp. 60–69, 2015.

[143] X. Zhao, Z. Zhu, and J. Caverlee, “Rabbit holes and taste distortion: Distribution-aware recommendation with evolving interests,” in *Proceedings of The Web Conference 2021*, 2021.

[144] T. Joachims, A. Swaminathan, and T. Schnabel, “Unbiased learning-to-rank with biased feedback,” in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 781–789, 2017.

[145] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural attentive session-based recommendation,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1419–1428, 2017.

[146] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung, “Streaming session-based recommendation,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1569–1577, 2019.

[147] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, “Stamp: short-term attention/memory priority model for session-based recommendation,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1831–1839, 2018.

[148] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for session-based recommendation,” in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 306–310, 2017.