Rabbit Holes and Taste Distortion: Distribution-Aware Recommendation with Evolving Interests

Xing Zhao, Ziwei Zhu, and James Caverlee
Department of Computer Science and Engineering, Texas A&M University
xingzhao,zhuziwei,caverlee@tamu.edu

ABSTRACT

To mitigate the *rabbit hole effect* in recommendations, conventional distribution-aware recommendation systems aim to ensure that a user's prior interest areas are reflected in the recommendations that the system makes. For example, a user who historically prefers comedies to dramas by 2:1 should see a similar ratio in recommended movies. Such approaches have proven to be an important building block for recommendation tasks. However, existing distribution-aware approaches enforce that the target taste distribution should exactly match a user's prior interests (typically revealed through training data), based on the assumption that users' taste distribution is fundamentally static. This assumption can lead to large estimation errors. We empirically identify this *taste distortion problem* through a data-driven study over multiple datasets. We show how taste preferences dynamically shift and how the design of a calibration mechanism should be designed with these shifts in mind. We further demonstrate how to incorporate these shifts into a taste enhanced calibrated recommender system, which results in simultaneously mitigated both the rabbit hole effect and taste distortion problem.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation, Distribution, Bias, Distortion, Diversification

ACM Reference Format:
Xing Zhao, Ziwei Zhu, and James Caverlee. 2021. Rabbit Holes and Taste Distortion: Distribution-Aware Recommendation with Evolving Interests. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450099

1 INTRODUCTION

One long-standing challenge in recommender systems is the *rabbit hole effect* [2, 35, 36]: 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 [41], fairness [18, 32, 47, 55], and diversity [6, 56]. 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 [35].

To address this rabbit hole effect, there are two main research directions: (i) diversity-focused recommendation, e.g., [3, 4, 13, 24, 26, 56]; and (ii) distribution-aware recommendation, e.g., [1, 27, 41, 53, 54]. *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 [39, 45]. Other methods eschew such explicit aspects in favor of identifying latent aspects as the basis of diversification [46]. In another direction, *distribution-aware recommendation* aims to ensure 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 [1, 27, 41], and has attracted considerable attention in the machine learning community [28, 43].

Although both diversity-focused recommenders and distribution-aware recommenders may help mitigate the *rabbit hole effect*, they further introduce a new ***taste distortion problem***: *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.

To illustrate this taste distortion problem, in Figure 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 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 [37] in orange) hints at the problem of the rabbit hole effect, as its

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450099

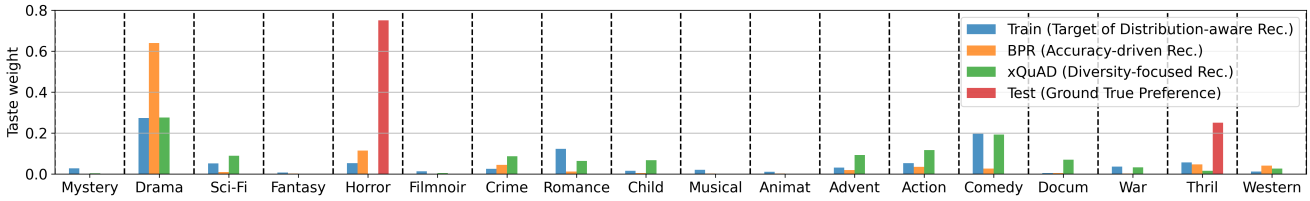

Figure 1: A sample user's taste distribution on the training set (blue, being the target of the distribution-aware recommendation), BPR predicted results (orange), xQuAD predicted results (green), and testing set (red, being the ground truth). Results suggest that none of these recommenders can provide the taste distribution in recommendations close to the ground truth.

recommendations skew towards drama even more than what is in the training distribution. A diversity-focused method (xQuAD [44] 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 [53], we focus in this paper 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 [30, 41]. 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 paper 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.

2 RELATED WORK

This section highlights related works in distribution-aware recommendations, diversity-focused approaches, and sequential recommendations.

2.1 Distribution-aware Recommendation

Recently, Steck proposed a distribution-aware recommendation (what we refer to as CaliRec in this paper), which focuses on the distribution of genres in a user's recommendation list [41]. 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 [27]. Zhao *et al.* focused on the distribution of target customers (with respect to certain demographics), proposing a post-ranking algorithm [53]. These previous 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. Furthermore, these and related studies have shown that the distribution adjustment and accurate recommendation tend to trade-off with each other [30, 41, 53].

2.2 Diversity-focused Approaches

Rather than aiming to match a user's taste distribution, some methods aim to introduce diversity into the list of ranked results [3, 4, 13, 24, 26, 56]. 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 [39, 45]. Some methods

eschew such explicit aspects in favor of identifying latent aspects as the basis of diversification [46].

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[53].

Furthermore, diversity-focused and distribution-aware recommender systems also differ in terms of evaluation. Traditional diversity-focused recommenders use feature-based metrics, such as α-*NDCG* [8] or subtopic recall (*S-recall*) [52]. Again, 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.

2.3 Sequential Recommendation

Another complementary approach to distribution-aware recommendation is sequential recommendation, which is designed specifically to track a user's changing interests, e.g., [20, 25, 28, 33, 42]. 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 [22]. Soon after, they improved their original work with a version that further boosts Top-N recommendation performance [21]. 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 [25]. Other sequential recommender systems also achieve impressive performance, e.g., [20, 49, 50]. 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 will not respect the requirements of taste distribution (and again could lead to the rabbit hole effect). In experiments reported in Section 6, we explore the connection between distribution-aware approaches and both diversity-focused and sequential-based methods.

3 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 tRain ($\mathbf{H}^R \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$), Validation ($\mathbf{H}^V \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$), and Test ($\mathbf{H}^T \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$) 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 $\mathbf{c}_i$:

$$\mathbf{c}_i = \left[\frac{v_{i,g_1}}{\Sigma}, \frac{v_{i,g_2}}{\Sigma}, \cdots, \frac{v_{i,g_{|\mathcal{G}|}}}{\Sigma}\right] \tag{1}$$

where $g_j \in \mathcal{G}$, $\Sigma = \sum_{g_j \in \mathcal{G}} v_{i,g_j}$, 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 $\mathbf{q}$ and the taste distribution in the training set as $\mathbf{p}$.

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

$$\mathbf{p}_{u,g_j} = \frac{\sum_{i\in\mathcal{I}} \mathbf{H}^R_{u,i} \times \mathbf{c}_{i,g_j}}{\sum_{i\in\mathcal{I}} \mathbf{H}^R_{u,i}} \tag{2}$$

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:

$$\mathbf{p}_u = \left[\mathbf{p}_{u,g_1}, \mathbf{p}_{u,g_2}, \cdots, \mathbf{p}_{u,g_{|\mathcal{G}|}}\right]. \tag{3}$$

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):

$$top(u,k) = \underset{\searrow,k}{\arg\operatorname{sort}} \left[\mathbf{D}_{u,1}, \mathbf{D}_{u,2}, \cdots, \mathbf{D}_{u,|I|}\right] \tag{4}$$

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:

$$\widetilde{\mathbf{q}}_u^{\text{Top}-k} = \frac{1}{k} \sum\nolimits_{i\in top(u,k)} \mathbf{c}_i. \tag{5}$$

4 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 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 $\widetilde{\mathbf{q}}_u^{\text{Top}-k}$. We define the *taste distortion* as $\varphi(\widetilde{\mathbf{q}}_u^{\text{Top}-k}, \mathbf{q}_u)$, where $\varphi(\cdot)$ denotes a distance measure,

	#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 1: Statistics for MovieLens-1M, MovieLens-20M, and Movie-Tweets datasets.

e.g., Kullback-Leibler (KL) divergence [31]. The larger this value is, the worse the estimation of the taste distribution on the test set is.

4.1 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 [19], 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 [19], which contains 20 million user-movie interactions collected from 138,493 users and 26,744 movies. The third is the MovieTweets dataset [10], 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 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) [37], and the conventional distribution-aware recommender (CaliRec) [41]. Experiments with other tested algorithms show similar results; our emphasis here is on the general problem of taste distortion that can manifest in recommenders.

4.2 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 $\mathbf{a}$ and $\mathbf{b}$, we use the Kullback-Leibler (KL) divergence [31] 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:

$$KL(\mathbf{a}||\mathbf{b}) = -\sum_{g_j \in \mathcal{G}} \mathbf{a}_{g_j} \log \frac{\mathbf{b}_{g_j}}{\mathbf{a}_{g_j}}. \quad (6)$$

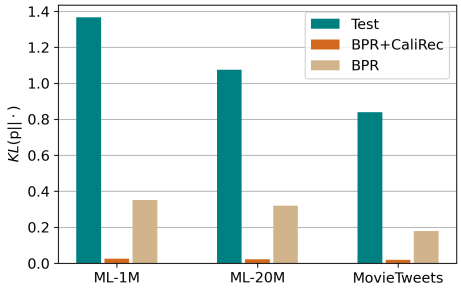

Figure 2: KL-Divergence between the taste distribution in training set (p) with the ones in testing set (q), in CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and in BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$).

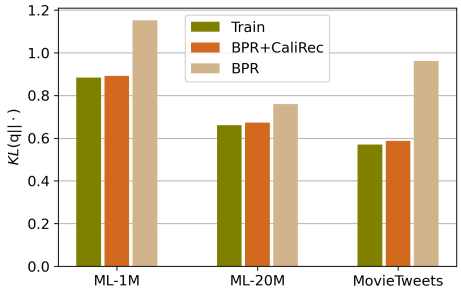

Figure 3: KL-Divergence between the taste distribution in ground truth (q) with the ones in training set (p), in CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and in BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$).

Figure 2 shows the KL-divergence for the taste distribution on the training dataset ($\mathbf{p}$) with the ground truth taste distribution on the testing dataset ($\mathbf{q}$), CaliRec results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR+CaliRec}}$), and BPR results ($\widetilde{\mathbf{q}}^{\text{Top}-100}_{\text{BPR}}$). 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 3 shows the taste distribution on the testing 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

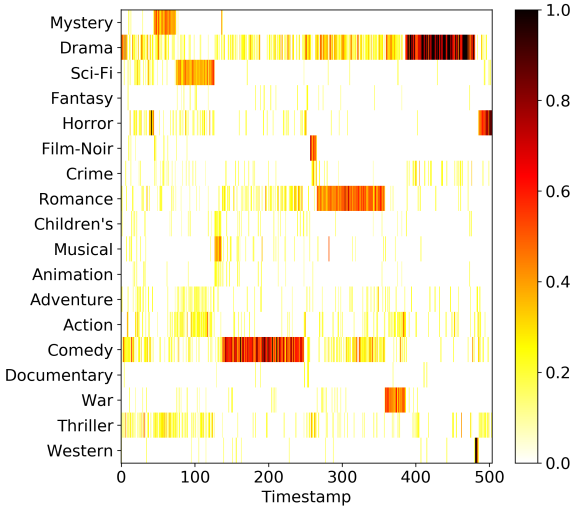

Figure 4: Trends of a sample user's taste overtime in the training set. As time goes by, this user changed her main taste through the path of Mystery → Sci-Fi → Comedy → Romance → War → Drama → Horror.

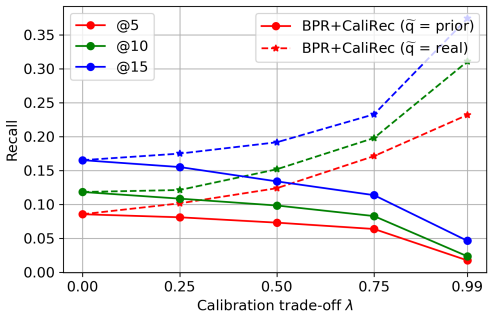

Figure 5: Recommendation results using CaliRec with prior and real taste distribution. Comparing with the BPR results ($\lambda = 0$), this "oracle" result is consistently better for all trade-off settings.

systems and distribution-aware recommender systems could underserve the ground truth taste distribution. However, it also indicates that a better estimation of taste distribution may produce more accurate recommendation results.

4.3 An Example of Taste Distortion

Figure 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 → Sci-Fi → Comedy → Romance → War → Drama → 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.

4.4 "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 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 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., [5, 25]; 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 5.1), and then incorporate these shifts into a post-ranking framework for improved distribution-aware recommendation (Section 5.2).

5.1 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 [41]. Yet, our observations in Section 4 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 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.

5.1.1 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 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.

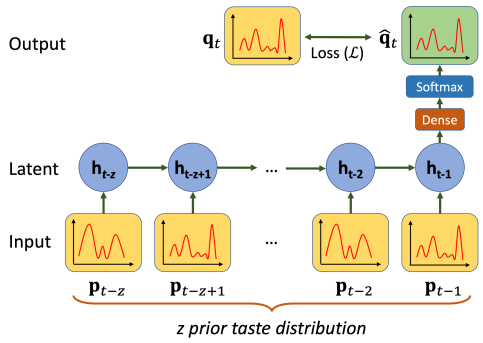

Figure 6: TecRec distribution prediction component, which takes the user's z prior time-series taste distributions as input, and outputs the predicted taste distribution in the next stage.

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:

$$\mathbf{p}_{u,t} = \frac{1}{\beta} \times \sum_{j=t-\beta+1}^{t} \mathbf{M}_{u,i,j} \times \mathbf{c}_i \tag{7}$$

$\mathbf{M} \in \mathbb{N}^{|\mathcal{U}|\times|\mathcal{I}|\times|\mathcal{T}|}$ 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, $\hat{\mathbf{q}}_{u,t}$.

5.1.2 Recurrent Model. Recurrent neural networks have shown great success in capturing the implicit dynamics of user-item interactions in recommender systems (e.g., [9, 25, 48]). 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 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) [11] 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 $\mathbf{W}$, $\mathbf{b}$ are the weights and bias term for each recurrent unit, and σ is a nonlinear activation function ($tanh$ in this paper).

Gated Recurrent unit (*abbr.* GRU) [7] 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 [14–17, 23]. 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:

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

5.1.3 Learning and Evaluation. In the learning process, we adopt the Adaptive Moment Estimation (Adam) [29] 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:

$$\mathcal{L}(\mathbf{q}_{u,t}, \hat{\mathbf{q}}_{u,t}) = -\sum_{g_i \in \mathcal{G}} \mathbf{q}_{u,g_i,t} \times \log\left(\frac{\hat{\mathbf{q}}_{u,g_i,t}}{\mathbf{q}_{u,g_i,t}}\right) \tag{8}$$

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):

$$arg\min_{\Theta} \frac{1}{N} \sum \mathcal{L}(\mathbf{q}, \hat{\mathbf{q}}) + \Phi \times \|\Theta\|_F^2. \tag{9}$$

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.

5.2 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 [41]. Therefore, a standard solution is post-ranking the predicted list in a post-processing step, which has been widely used in machine learning approaches [12, 41, 51]. Post-ranking

approaches could be integrated into existing models without retraining 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.

Algorithm 1: TecRec Post-ranking Mechanism

Input: u, $\mathbf{D}_{u,:}$, $\hat{\mathbf{q}}$, λ, Z, k
Output: $\mathbf{T}_u$; // Recommendation for u from TecRec

1 $\mathbf{T}_u \leftarrow \emptyset$;
2 $\mathbf{d} = \underset{\searrow Z}{\arg\text{sort}} \left[\mathbf{D}_{u,1}, \mathbf{D}_{u,2}, \cdots, \mathbf{D}_{u,|I|}\right]$;
3 **while** $|\mathbf{T}_u| < k$ **do**
4 $\quad i^* \leftarrow \underset{i \in \mathbf{d} \setminus \mathbf{T}_u}{\arg\max} \underbrace{(1-\lambda) \times \sum_{v \in \mathbf{T}_u \cup \{i\}} \frac{\mathbf{D}_{u,v}}{|\mathbf{T}_u|+1}}_{\text{accuracy term}} \underbrace{- \lambda \times \varphi\Big(\hat{\mathbf{q}}, \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_v\Big)}_{\text{calibration term}}$;
5 $\quad \mathbf{d} \leftarrow \mathbf{d} \setminus \{i^*\}$;
6 $\quad \mathbf{T}_u \leftarrow \mathbf{T}_u \cup \{i^*\}$; // Update current optimal result
7 **end**
8 **return** $\mathbf{T}_u$;

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 5.1), 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 [41]. Prior research [40] 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:

$$\begin{aligned}
&\varphi\Big(\hat{\mathbf{q}}, \sum_{v \in \mathbf{T} \cup \{i\}} \mathbf{c}_v\Big) = KL\Big(\hat{\mathbf{q}} \| \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_v\Big) \\
&= -\sum_{g_k \in \mathcal{G}} \hat{\mathbf{q}}_{g_k} \times \log\left(\frac{\sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_{v,g_k}}{\hat{\mathbf{q}}_{g_k}}\right) \\
&= -\Big(\sum_{g_k \in \mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_{v,g_k} - \sum_{g_k \in \mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \hat{\mathbf{q}}_{g_k}\Big) \\
&= \underbrace{\text{Entropy}(u)}_{\text{constant}} - \sum_{g_k \in \mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_{v,g_k}
\end{aligned} \tag{10}$$

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

$$\begin{aligned}
i^* \leftarrow \underset{i \in \mathbf{d} \setminus \mathbf{T}_u}{\arg\max} &(1-\lambda) \times \sum_{v \in \mathbf{T}_u \cup \{i\}} \frac{\mathbf{D}_{u,v}}{|\mathbf{T}_u|+1} \\
&+ \lambda \times \sum_{g_k \in \mathcal{G}} \hat{\mathbf{q}}_{g_k} \log \sum_{v \in \mathbf{T}_u \cup \{i\}} \mathbf{c}_{v,g_k}
\end{aligned} \tag{11}$$

where the greedy optimization of submodular functions achieves a $(1 - \frac{1}{e})$ guarantee of optimality (e is Euler's number) [34].

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.

6 EXPERIMENTS AND RESULTS

In this section, we conduct a series of experiments over the three datasets introduced in Section 4.1. 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?

6.1 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.

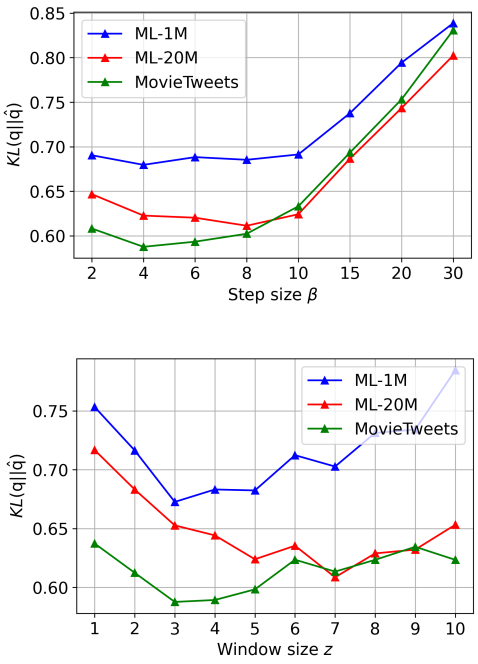

Figure 7: Hyper-parameters tuning for TecRec Distribution Prediction Component.

6.1.1 Hyper-parameter Tuning. Figure 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 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 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.

6.1.2 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 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 [37];
- **SASRec**: a popular sequential recommender using a two-layer Transformer decoder to capture user's sequential behaviors and achieving state-of-the-art results [25];
- **xQuAD**: a diversity-focused recommender proposed in [44] adapted from the Query Aspect Diversification framework [38], where user u's preferences are formulated as a probability distribution over aspects (i.e., genres);
- **SPAD**: a variant of xQuAD proposed in [26], 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., $\tilde{\mathbf{q}}^{\text{Top}-10}_{\text{xQuAD}}$, $\tilde{\mathbf{q}}^{\text{Top}-10}_{\text{SPAD}}$, $\tilde{\mathbf{q}}^{\text{Top}-10}_{\text{BPR}}$, and $\tilde{\mathbf{q}}^{\text{Top}-10}_{\text{SASRec}}$. 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 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 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

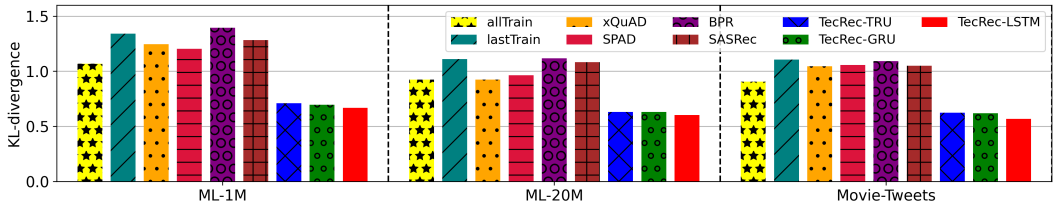

Figure 8: Dynamic taste distribution prediction results.

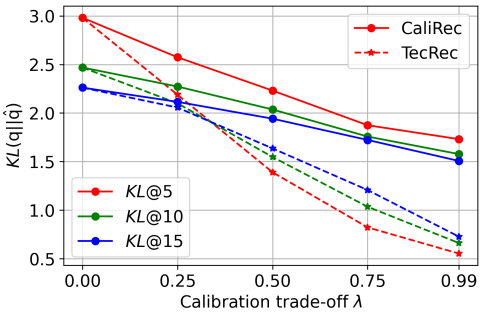

Figure 9: KL-divergence between the true distribution q with $\tilde{\mathbf{q}}^{\text{Top}-k}_{\text{CaliRec}}$ and $\tilde{\mathbf{q}}^{\text{Top}-k}_{\text{TecRec}}$ on all calibration trade-off settings.

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 6.2), though the particular choice appears not to be critical.

6.1.3 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., [12, 41, 54]). With a choice of $\lambda = 0$, we default to the baseline recommender (e.g., BPR). Figure 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?

6.2 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 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 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 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

		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 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.

results. Secondly, the recommendation results improve by different amounts for different base methods (e.g., Δ_{BPR} = 10.5% v.s. Δ_{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 8). Third, comparing the results with the oracle results (refer to Figure 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 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 λ = 0.99). More specifically, with the growth of K, the improvement from the CaliRec increases more quickly (refer to Figure 10). For example, when we set λ = 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 λ = 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 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 observation also motivates our continued research: how can we accurately predict users' taste distribution for the next few stages, rather than just one?

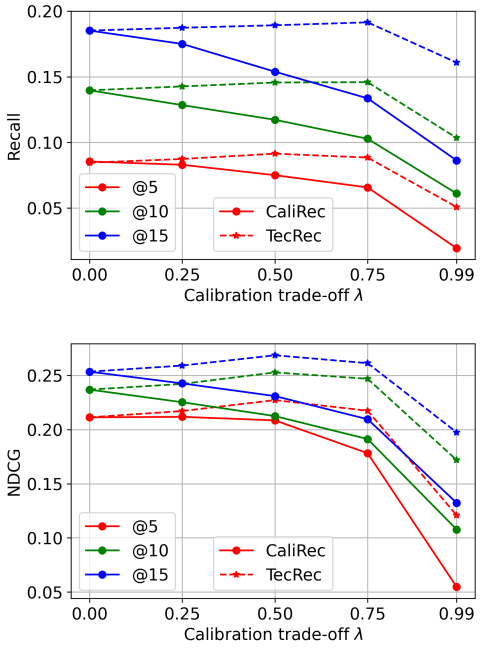

Figure 10: Recommendation results comparison: CaliRec and TecRec with the same base SASRec on ML-1M dataset.

7 CONCLUSION AND FUTURE WORK

In this paper, we first identified the taste distortion problem, and 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.

Motivated by the "oracle" results, we observed considerable room to improve the recommendation results using a better estimation of taste distribution. In our future work, we would like to introduce a more advanced time-series model to study users' taste distribution. Also, we plan to propose an end-to-end recommender that could monitor and detect in real-time the distribution shifts and calibrate it during the recommendation process. Another direction is to incorporate cyclical patterns of user preferences. For example, a user's preference for fashion may change over the course of a year, but return to Christmas styles during the holiday season. Hence, we need not only more extended memory of the prediction model, but also a more accurate and self-controlled timing slice and more flexible learning mechanism.

REFERENCES

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286* (2019).

[2] Titipat Achakulvisut, Daniel E Acuna, Tulakan Ruangrong, and Konrad Kording. 2016. Science Concierge: A fast content-based recommendation system for scientific publications. *PloS one* 11, 7 (2016).

[3] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2014. Diversified Utility Maximization for Recommendations.. In *RecSys Posters*.

[4] Aditya Bhaskara, Mehrdad Ghadiri, Vahab Mirrokni, and Ola Svensson. 2016. Linear relaxations for finding diverse elements in metric spaces. In *Advances in Neural Information Processing Systems*. 4098–4106.

[5] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 79–82.

[6] Laming Chen, Guoxin Zhang, and Hanning Zhou. 2017. Improving the diversity of top-N recommendation via determinantal point process. In *Large Scale Recommendation Systems Workshop*.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 659–666.

[9] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 152–160.

[10] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, Vol. 2013. 43.

[11] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[12] Dean P Foster and Rakesh V Vohra. 1998. Asymptotic calibration. *Biometrika* 85, 2 (1998), 379–390.

[13] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. 2016. Bayesian low-rank determinantal point processes. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 349–356.

[14] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2002. Applying LSTM to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*. Springer, 193–200.

[15] Felix A Gers and E Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12, 6 (2001), 1333–1340.

[16] Felix A Gers and Jürgen Schmidhuber. 2000. 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. IEEE, 189–194.

[17] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).

[18] Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*. 3315–3323.

[19] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2016), 19.

[20] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[21] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[22] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[24] Neil Hurley and Mi Zhang. 2011. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)* 10, 4 (2011), 14.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[26] Mesut Kaya and Derek Bridge. 2018. Accurate and diverse recommendations using item-based subprofiles. In *The Thirty-First International Flairs Conference*.

[27] Mesut Kaya and Derek Bridge. 2019. A comparison of calibrated and intent-aware recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, 151–159.

[28] Mesut Kaya and Derek Bridge. 2019. Subprofile-aware diversification of recommendations. *User Modeling and User-Adapted Interaction* 29, 3 (2019), 661–700.

[29] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[30] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807* (2016).

[31] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[32] Weiwen Liu, Jun Guo, Nasim Sonboli, Robin Burke, and Shengyu Zhang. 2019. Personalized fairness-aware re-ranking for microlending. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, 467–471.

[33] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 483–491.

[34] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[35] Derek O'Callaghan, Derek Greene, Maura Conway, Joe Carthy, and Pádraig Cunningham. 2015. Down the (white) rabbit hole: The extreme right and online recommender systems. *Social Science Computer Review* 33, 4 (2015), 459–478.

[36] Alexander M Petersen, Woo-Sung Jung, Jae-Suk Yang, and H Eugene Stanley. 2011. Quantitative and empirical demonstration of the Matthew effect in a study of career longevity. *Proceedings of the National Academy of Sciences* 108, 1 (2011), 18–23.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[38] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*. 881–890.

[39] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. 2012. On the role of novelty for search result diversification. *Information retrieval* 15, 5 (2012), 478–502.

[40] Yusuke Shinohara. 2014. A submodular optimization approach to sentence set selection. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4112–4115.

[41] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM conference on recommender systems*. ACM, 154–162.

[42] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[43] Virginia Tsintzou, Evaggelia Pitoura, and Panayiotis Tsaparas. 2018. Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461* (2018).

[44] Saúl Vargas. 2015. *Novelty and diversity evaluation and enhancement in recommender systems.* Ph.D. Dissertation. Ph. D. Dissertation. Universidad Autónoma

de Madrid.

[45] Saul Vargas, Pablo Castells, and David Vallet. 2011. Intent-oriented diversity in recommender systems. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 1211–1212.

[46] Jacek Wasilewski and Neil Hurley. 2016. Intent-aware diversification using a constrained PLSA. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 39–42.

[47] Blake Woodworth, Suriya Gunasekar, Mesrob I Ohannessian, and Nathan Srebro. 2017. Learning non-discriminatory predictors. *arXiv preprint arXiv:1702.06081* (2017).

[48] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[49] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future data helps training: Modeling future contexts for session-based recommendation. In *Proceedings of The Web Conference 2020*. 303–313.

[50] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.

[51] Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Icml*, Vol. 1. Citeseer, 609–616.

[52] ChengXiang Zhai, William W Cohen, and John Lafferty. 2015. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *ACM SIGIR Forum*, Vol. 49. ACM New York, NY, USA, 2–9.

[53] Xing Zhao, Ziwei Zhu, Majid Alfifi, and James Caverlee. 2020. Addressing the Target Customer Distortion Problem in Recommender Systems. In *Proceedings of The Web Conference 2020*. 2969–2975.

[54] Xing Zhao, Ziwei Zhu, Yin Zhang, and James Caverlee. 2020. Improving the Estimation of Tail Ratings in Recommender System with Multi-Latent Representations. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 762–770.

[55] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1153–1162.

[56] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.