Beyond Binary Preference: Leveraging Bayesian Approaches for Joint Optimization of Ranking and Calibration

Chang Liu*†
isonomialiu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Qiwei Wang*
shimmerwang@tencent.com
Tencent, Shenzhen, China

Wenqing Lin‡
edwlin@tencent.com
Tencent, Shenzhen, China

Yue Ding‡
dingyue@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Hongtao Lu
htlu@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Predicting click-through rate (CTR) is a critical task in recommendation systems, where the models are optimized with pointwise loss to infer the probability of items being clicked. In industrial practice, applications also require ranking items based on these probabilities. Existing solutions primarily combine the ranking-based loss, *i.e.,* pairwise and listwise loss, with CTR prediction. However, they can hardly calibrate or generalize well in CTR scenarios where the clicks reflect the binary preference. This is because the binary click feedback leads to a large number of ties, which renders high data sparsity. In this paper, we propose an effective data augmentation strategy, named Beyond Binary Preference (BBP) training framework, to address this problem. Our key idea is to break the ties by leveraging Bayesian approaches, where the beta distribution models click behavior as probability distributions in the training data that naturally break ties. Therefore, we can obtain an auxiliary training label that generates more comparable pairs and improves the ranking performance. Besides, BBP formulates ranking and calibration as a multi-task framework to optimize both objectives simultaneously. Through extensive offline experiments and online tests on various datasets, we demonstrate that BBP significantly outperforms state-of-the-art methods in both ranking and calibration capabilities, showcasing its effectiveness in addressing the limitations of existing methods. Our code is available at https://github.com/AlvinIsonomia/BBP.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

*Both authors contributed equally to this research.
†This work was done while Chang Liu was an intern at Tencent.
‡Corresponding authors: Wenqing Lin and Yue Ding.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671577

KEYWORDS

Click-through rate prediction; Bayesian approach; Ranking; Calibration; Recommendation Systems

ACM Reference Format:
Chang Liu, Qiwei Wang, Wenqing Lin, Yue Ding, and Hongtao Lu. 2024. Beyond Binary Preference: Leveraging Bayesian Approaches for Joint Optimization of Ranking and Calibration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671577

1 INTRODUCTION

Click-through rate (CTR) is the probability of a user clicking on a recommended item. CTR prediction, which attempts to estimate the CTR given a user-item impression accurately, is critical for achieving precise recommendations and increasing revenue for enterprises [34, 59]. Consequently, it plays a significant role in recommendation systems, such as online advertising and in-game player friend recommendations [18, 31, 32, 52, 55, 56, 61].

In industrial applications, there are two use cases for CTR prediction models: 1) forms an advertisement ranked list according to predicted CTR, and 2) calibrates click probabilities for downstreaming recommendation tasks. For example, in sponsored search, the final presentation to users is a ranked list of ads [27], and the calibrated scores produced is a required input for computing the cost-per-click (CPC) after ad impressions, which decides the presentation of an ad based on a threshold [51]. Therefore, there is a significant demand for a CTR prediction model that can align with the actual click-through rate (*i.e.,* calibration) and lead to a correct ranking [43]. Existing methods combine pairwise and listwise training in Learn-to-Rank (LTR) [7, 33] with CTR predictions. Recent works focus on enhancing ranking performance in CTR prediction with pairwise and listwise methods. For example, Bayesian Personalized Ranking [40] and Smooth-AUC [47] leverage the pairwise training. Regression Compatible Ranking [3], Scale Calibration [51], and Joint optimization of Ranking and Calibration [44] further introduce listwise loss to enhance the ranking performance.

Despite their reported success, they may fall into sub-optimal due to the binary feedback. The key issue ignored is that **the binary feedback leads to a large number of ties**, which consequently causes insufficient feasible training pairs. Since the LTR methods

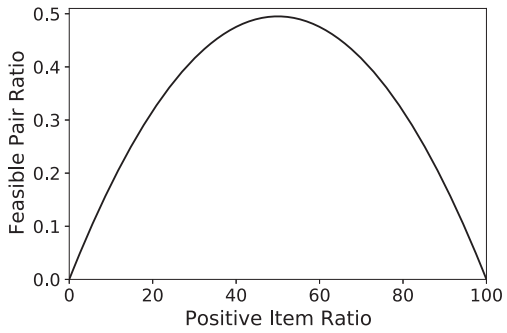

Figure 1: Visualization of the ratio of feasible preference pair to all possible pairs. Since the preference feedback in the CTR scenario is binary, the ratio of feasible pairs is strongly affected by the positive item ratio and can be very sparse.

optimize models based on preference pairs (*e.g.,* two items with different preferences for the same user), the positive ratio will strongly affect the number of feasible training pairs. We present a toy example in Figure 1 with N items to better illustrate the situation. Under the best condition, only half of the $\binom{N}{2}$ possible pairs can be constructed as the preference pairs, with the positive-item ratio being 50%. Furthermore, the positive feedback (user clicks) in the real world is highly sparse. For example, if only 10% are positive items, there will be only 18% feasible pairs. Therefore, binary preference restricts the effectiveness of existing methods for CTR predictions.

To address this issue, we propose the Beyond Binary Preference (BBP) training framework. Our key idea is to augment training labels with continuous Bayesian probability to break the ties. We treat the historical records as an observation sequence of Bernoulli trails and calculate the estimated probability score for all users and items through Bayesian smoothing [42]. Afterward, we combine the probability score and binary feedback to break the ties and optimize the model. Since the prior ranking score is continuous rather than binary, more preference pairs are constructed, which addresses the data sparsity caused by binary feedback. For joint optimization of ranking and calibration. In particular, we design a multi-task learning framework with a global context ranking loss that reuses the arbitrary user-item pairs in the same batch to augment more preference pairs. Extensive offline experiments on three public benchmark datasets and online A/B testing on two real-world scenarios demonstrate that our BBP framework outperforms the state-of-the-art (SOTA) competitors in both ranking and calibration capabilities.

Contributions. The contributions of our paper include:

- To the best of our knowledge, we are the first to identify and address the problem of insufficient samples for ranking loss caused by binary preference feedback in CTR prediction.
- We propose the Beyond Binary Preference (BBP), which models and estimates a personalized beta distribution to each user and item in the training set with Bayesian methods. BBP further generates a continuously comparable ranking score label to break the ties caused by the binary preference feedback.
- We demonstrate the effectiveness of the BBP framework through extensive offline experiments and online A/B tests. BBP outperforms all SOTA competitors significantly (p-value < 0.05) in both calibration and ranking metrics on three public benchmarks. The deployed BBP based recommendation modules increase at least 10.28% more friends in two Tencent online games than competitors relatively.

2 RELATED WORK

2.1 CTR Prediction

Pointwise training and calibration. CTR prediction usually optimizes the models with the Binary Cross Entropy (BCE) loss function; therefore its prediction scores strictly calibrate click probabilities that an item on a website (such as an advertisement) will be clicked [13, 38]. Existing methods study various model architectures for more accurate CTR prediction, including statistical methods [12, 48], logistic regression [36, 41], factorization machines [23, 39], and DNNs-based CTR models [8, 14, 15, 20, 29, 59]. However, pointwise training suffers from limited ranking performance, resulting in sub-optimal recommendation outcomes. It is worth mentioning that the calibration ability of CTR models can be confused with a different research topic called "uncertainty calibration" [9, 25]. In uncertainty calibration, the focus is on confidence estimates rather than calibrating the model's output with click probabilities.

Ranking-based CTR prediction. In industrial applications, the ranking ability of CTR models is also crucial [13, 33], as we need to provide an advertisement ranked list for down-streaming recommendation tasks. Recent work has demonstrated that the CTR models trained with pointwise loss suffer from limited ranking performance [3, 44, 47, 51] due to a lack of preference ordering information during the training. To address this issue, researchers have proposed various pairwise and listwise approaches, which differ from pointwise methods in that they directly optimize ranking objectives. Pairwise methods compare pairs of items, while listwise methods consider the entire impression list of items for the user. These approaches aim to directly optimize ranking metrics such as AUC (Area Under the Curve) [11] and NDCG (Normalized Discounted Cumulative Gain) [21]. Bayesian Personalized Ranking (BPR) [40] optimizes the order of pairs of positive and negative items. Smooth-AUC (SAUC) loss introduces a smooth approximation of AUC using a sigmoid function. Though advanced ranking losses may produce significantly better ranking performance, the training instability issues prohibit their usage in the CTR predictions, as the scores may keep drifting during model training [6, 51]. This will cause numerical failures in real-world large scale learning systems, especially under the continual learning paradigm. For example, one has to modify the threshold in CPC to keep up with the temporal dynamics of systems updates.

Joint optimization of ranking and calibration. Training CTR models that can rank and calibrate well has been extensively studied in the last decades [3, 44, 51]. Regression Compatible Ranking (RCR) [3] achieves a trade-off between ranking (ListNet [50]) and calibrated regression (BCE) rather than post-processing methods

[37]. Scale Calibration (SC) [51] performs scale calibration of ranking models by introducing a trainable parameter in the softmax loss, addressing training instability issues. Joint optimization of Ranking and Calibration (JRC) [44] simultaneously optimizes ranking and calibration abilities by contrasting the logit value for samples with different labels and constraining the predicted probability as the logit subtraction. However, they overlooked the data sparsity due to the ubiquitous ties (*i.e.,* pairs with the same label will not be used for training). In this paper, we propose the Beyond Binary Preference (BBP) training framework to address the limitations of ranking-based CTR training further.

2.2 Learning-to-rank with Ties

Learning-to-rank with ties has been a focus of research as it addresses the issue of instances having nearly identical degrees of relevance, referred to as 'ties' [5]. There are two lines of research to make full use of ties. One option is to force the model to output similar predictions for items with the same label. Examples of this approach include designing an extra loss function for the same relevance [60], re-ranking items with a given pairwise relevance threshold [57], and using a hashing function for tie-aware ranking [17], among others. The other option, in contrast, is to *break the ties* during training by ensuring that no equal relevance exists. For instance, uRank [62] overcomes the tie issue of existing Plackett-Luce models by maximizing the likelihood of selecting documents with high ratings over documents with low ratings. Savant [26] randomly assigns the same relevance to different labels using different random seeds. However, existing methods fall short in CTR prediction tasks when the relevance is binary. As shown in Figure 1, the number of ties is almost always greater than the number of comparable preference pairs. The high data sparsity greatly restricts the effectiveness of existing methods. Our BBP method leverages Bayesian probabilities to transform binary labels into continuous scores, effectively breaking the ties in CTR prediction.

3 PRELIMINARIES

3.1 CTR Prediction

Let $\mathcal{D}$ be an impression dataset. In CTR prediction, typically, each sample in $\mathcal{D}$ consists of examples represented as tuples $(u, v, y^{u,v})$, where $u \in \mathcal{U}$ be a user ID, $v \in \mathcal{V}$ be an item ID, and $y^{u,v} \in \{0, 1\}$ is the binary preference label, *i.e.,* user clicks. The CTR prediction model f aims to assign a score s to a user item pair (u, v) given its feature vector $\mathbf{x}^{u,v}$, *i.e.,* $s^{u,v} = f(\mathbf{x}^{u,v})$. The output s indicates the likelihood that user u will click item v. The model f is usually optimized from the following two aspects.

Calibration. Traditionally, the calibrated CTR model is learned and evaluated with binary preference in the pointwise setting [3, 44]. The loss function in learning is defined on a single data point $\mathbf{x}^{u,v}$ of each user-item pair in dataset $\mathcal{D}$. Formally, the empirical risk function of calibration $\mathcal{L}_{\text{cal}}$ is defined as:

$$\mathcal{L}_{\text{cal}} = \sum_{(u,v,y^{u,v}) \in \mathcal{D}} \ell_{\text{cal}} \left(\sigma(s^{u,v}), y^{u,v}\right), \tag{1}$$

where $s^{u,v} = f(\mathbf{x}^{u,v})$, σ is the sigmoid function, $y^{u,v}$ is the binary preference feedback for user item pair (u, v), $\ell_{\text{cal}}\left(s^{u,v}, y^{u,v}\right)$ denotes a pointwise loss function based on a binary classification measure [16]. In this way, the prediction of the model f indicates click probability.

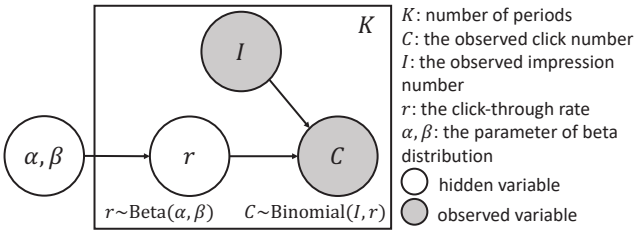

Figure 2: The graphical model of the CTR, where shaded variables (impression I and click C) are observable.

Ranking. As discussed above, industrial applications also leverage prediction score $f(u, v)$ to rank the candidate items. The CTR model f is still defined on a user-item pair (u, v), and the loss function is defined on two data points $\mathbf{x}^{u,v}$ and $\mathbf{x}^{u,v'}$ that can formulate preference judgement pairs [58].

Definition 1 (preference judgement pair). *Formally, given two items v and v' to the same user u, a preference judgement pair is defined as: $(u, v) \succ (u, v')$, where $y^{u,v}$ is greater than $y^{v,v'}$. In the context of binary preference, it means v is the clicked item, and v' is unclicked by the user u.*

Therefore, the empirical risk function $\mathcal{L}_{\text{rank}}$ is defined as:

$$\mathcal{L}_{\text{rank}} = \sum_{\langle (u,v) \succ (u,v') \rangle} \ell_{\text{rank}} \left(s^{u,v}, y^{u,v}, s^{u,v'}, y^{u,v'}\right), \tag{2}$$

where $\ell_{\text{rank}}\left(s^{u,v}, y^{u,v}, s^{u,v'}, y^{u,v'}\right)$ denotes a pairwise loss function. As the labels are binary, *i.e.,* click or not click, there will be a large number of ties, as mentioned in Section 1. As ties cannot perform preference judgments, existing methods usually drop them during the training, which may lead to sub-optimal.

In this work, we address this limitation by bringing an augmented label to break the ties. Specifically, we model the click behaviors and use the overall historical CTR as prior probabilities. With more impressions observed in the chronic sequence, we gradually update user-item pair posterior probabilities with Bayesian Smoothing as the augmented label.

3.2 Click Behavior Modeling

Our click behavior model is shown in Figure 2: when a user receives an impression of an item, the user decides to click or not according to his/her interest. Considering the binary nature of the click behavior, we treat each user-item impression as a Bernoulli trial with an underlying probability r, *i.e.,* $C \sim \text{Binomial}(I, r)$. In addition, the underlying probability r that a user clicks an item follows a beta distribution, *i.e.,* $r \sim \text{Beta}(\alpha, \beta)$.

We choose the beta distribution because it can represent a variety of distributions, which has been widely applied to many different settings [22, 24]. Despite the wide applicability, the beta distribution only has two parameters: $\alpha > 0$ and $\beta > 0$. Its probability density

function (PDF) is defined as:

$$\mathrm{PDF}(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \tag{3}$$

where the normalization factor, $B(\alpha,\beta) = \int_0^1 \mathrm{PDF}(t;\alpha,\beta)dt$, is the complete regularized beta function that ensures that the PDF integrates into one. It models the likelihood function of the parameter underlying a Bernoulli trail, where $\alpha-1$ is the number of successes (click) and $\beta-1$ is the number of failures. For an intuitive understanding, in the CTR predictions, it means $\alpha+\beta-2$ impressions for a user-item pair where $\alpha-1$ clicks and $\beta-1$ unclicks are observed. If the click probability of the user-item pair is r, the probability of the above observations would be proportional to $r^{\alpha-1}(1-r)^{\beta-1}$. The mean of the distribution μ is determined by the ratio α and the sum $\alpha+\beta$, where an α higher than the β means that the mean is closer to 1 than to 0, and vice versa. We further have:

$$\mathbb{E}[r] = \mu = \frac{\alpha}{\alpha+\beta}, \tag{4}$$

where $\mathbb{E}[r]$ reflects the expectation of the click probability for the user-item pair. The conjugacy between Binomial and Beta enables us to estimate the users' preference at each time period given the historical observations with Bayesian approaches. The distribution of the click observations related to the underlying preference distribution can then be expressed as smoothing the personalized distribution from the empirical prior distribution, which will be described in Eq. (8) in detail. In the following section, we will leverage the personalized distributions as the augment labels to break the ties and address the data sparsity caused by the binary preference.

4 METHODOLOGY

Our goal is to estimate a continuous augment ranking score to break the ubiquitous binary ties. We first leverage Bayesian smoothing to estimate the Bayesian probabilities according to the historical observations in Sec. 4.1. Furthermore, we combine the estimated probabilities with binary labels as augmented preference scores, which enables us to extend the preference judgment pairs definition in Sec. 4.2. Since the augmented score is continuous, it breaks the ties from binary labels. Finally, we jointly optimize the ranking and calibration in a multitask learning framework in Sec. 4.3. Specifically, we design a global context ranking loss that compares arbitrary user-item pairs instead of items for the same user, which improves the model performance further.

4.1 Learning Beta Distribution for Click Inference

With the behavior modeling in Sec. 3.2, we aim to estimate the beta distribution for the user behavior, which is used as the augment label with Bayesian approaches. The main idea is to arrange the historical data chronologically and gradually update the personalized beta distribution for each user (item) with Bayesian smoothing from an initial distribution.

Data formulation. As the user's feedback is gradually collected from the CTR systems' online serving process, it naturally forms a Bernoulli trial sequence. Let $\mathcal{V}_u = \{v|(u,v,y) \in \mathcal{D}\}$ be the interacted item set of user u induced by $\mathcal{D}$. Similarly, we can also denote $\mathcal{U}_v = \{u|(u,v,y) \in \mathcal{D}\}$ as the interacted user set for item v. Admittedly, in click behavior modeling, the click rate r should depend on the specific user-item pair. Unfortunately, historical data is often very sparse (*i.e.,* a user u always has only been exposed to a few items in the entire item set $\mathcal{V}$). This makes it difficult to estimate the beta distribution for each user-item pair. Therefore, we estimate the click probability distribution for each user u and each item v independently. Intuitively, this can be understood as modeling whether a user is more willing to click on items and whether an item is more attractive for users to click on.

Given the collected dataset $\mathcal{D}$, we first reorganize it chronically based on K periods of time, *i.e.,* $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$, where the $\mathcal{D}_k$ is the sub-dataset of the k-th period. The period here can be a period of time (such as a day or a week), or in extreme cases, it can be each timestamp. Without loss of generalization, the methods for modeling users and items are the same. Therefore, we only introduce the click probability distribution estimation of the item in detail here. Let $I^v = \left(I_1^v, I_2^v, \cdots, I_K^v\right)$ be a sequence of impression sequence for item v with length K, and $C^v = \left(C_1^v, C_2^v, \cdots, C_K^v\right)$ be a sequence of clicks for item v. Here $I_k^v = |(u,v,y^{u,v})|$, where $(u,v,y^{u,v}) \in \mathcal{D}_k$, and $C_k^v = \sum_{(u,v,y^{u,v})\in\mathcal{D}_k} y^{u,v}$. Next, we detail how to leverage the arranged sequence to update the personalized beta distribution for item v.

Initializing the beta distribution. In real applications, to obtain a more accurate estimation, especially for the users/items with rare historical impression records, we can initialize the beta distribution by learning from all the historical data.

Let $\mathrm{Beta}(\alpha_0,\beta_0)$ denote the initialized beta distribution as the prior distribution for all items. The α_0 is the average click number among all items, and β_0 is the average unclicked impression number. Then the initialized distribution is given by:

$$\alpha_0 = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{U}_v} y^{u,v},\ \beta_0 = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{U}_v}(1-y^{u,v}), \tag{5}$$

where $|\mathcal{V}|$ the number of item and $\mathcal{U}_v = \{u|(u,v,y) \in \mathcal{D}\}$ is the interacted user set for item v. Note that it's better to use the datasets from $\mathcal{D}_0$ where the 0-th period means that all available historical impressions before $\mathcal{D}_1$. However, if the $\mathcal{D}_0$ is unavailable, we can simply leverage the statistics from current $\mathcal{D}$.

Updating the personalized distribution. Given the beta distribution $\mathrm{Beta}(\alpha^v,\beta^v)$, we can derive the likelihood that we observe click sequence C_k^v given the impression sequence for I_k^v as:

$$P\left(C^v \mid I^v;\alpha^v,\beta^v\right) = \prod_{k=1}^{K}\frac{\Gamma(\alpha^v+\beta^v)}{\Gamma\left(I_k^v+\alpha^v+\beta^v\right)}\frac{\Gamma\left(\alpha^v+C_k^v\right)}{\Gamma(\alpha^v)}\frac{\Gamma\left(I_k^v-C_k^v+\beta^v\right)}{\Gamma(\beta^v)}, \tag{6}$$

where the $\Gamma(\cdot)$ is the gamma function [2]. To investigate what is the optimal α^v,β^v, we take the partial derivatives of the log-likelihood function with respect to α^v and β^v as:

$$\begin{aligned}\frac{d\log P\left(C^v \mid I^v;\alpha^v,\beta^v\right)}{d\alpha^v} &= \sum_{k=1}^{K}[\psi(\alpha^v+\beta^v)-\psi\left(I_k^v+\alpha^v+\beta^v\right)\\ &\quad +\psi\left(\alpha^v+C_k^v\right)-\psi(\alpha^v)],\\ \frac{d\log P\left(C^v \mid I^v;\alpha^v,\beta^v\right)}{d\beta^v} &= \sum_{k=1}^{K}[\psi(\alpha^v+\beta^v)-\psi\left(I_k^v+\alpha^v+\beta^v\right)\\ &\quad +\psi\left(I_k^v-C_k^v+\beta^v\right)-\psi(\beta^v)],\end{aligned} \tag{7}$$

where $\psi(x)=\frac{\Gamma'(x)}{\Gamma(x)}$ is the Psi (digamma) function which can be calculated quickly with Bernardo's algorithm [4].

Using the fixed-point iteration method [49], we can derive the update formulas for α^v and β^v for N iterations:

$$\begin{aligned}\alpha_n^v &= \alpha_{n-1}^v\frac{\sum_{k=1}^{K}\left[\psi\left(\alpha_{n-1}^v+C_k^v\right)-\psi(\alpha_{n-1}^v)\right]}{\sum_{k=1}^{K}\left[-\psi(\alpha_{n-1}^v+\beta_{n-1}^v)+\psi\left(I_k^v+\alpha_{n-1}^v+\beta_{n-1}^v\right)\right]},\\ \beta_n^v &= \beta_{n-1}^v\frac{\sum_{k=1}^{K}\left[\psi\left(I_k^v-C_k^v+\beta_{n-1}^v\right)-\psi(\beta_{n-1}^v)\right]}{\sum_{k=1}^{K}\left[-\psi(\alpha_{n-1}^v+\beta_{n-1}^v)+\psi\left(I_k^v+\alpha_{n-1}^v+\beta_{n-1}^v\right)\right]},\end{aligned} \tag{8}$$

where the $n\in[1,2,\cdots,N]$ is the iteration index. The iteration process continues until convergence or the maximum number of iterations is reached. We treat the output distribution $\text{Beta}(\alpha^v,\beta^v)$ as the final personalized distribution for item v's click probability.

Getting personalized posterior probabilities. For the posterior probability estimation, under the prior personalized probability $\text{Beta}(\alpha^v,\beta^v)$, as in the last period we observe C_k^v clicks after I_k^v impressions, then the posterior distribution is $\text{Beta}(\alpha^v+C_k^v,\beta^v+I_k^v-C_k^v)$. According to the expectation in Eq. (4), we can easily obtain $P_{\text{clicked}}^v=\frac{\alpha^v+C_k^v}{I_k^v+\alpha^v+\beta^v}$. Similarly, we can compute the beta distribution for user u clicks and the corresponding probability P_{click}^u.

We can use the estimated posterior probability as augmented labels to break ties in binary labels. This approach provides a continuous score that can effectively break ties in binary click-through rate prediction tasks, enhancing the model's ranking performance while maintaining its calibrated outputs.

4.2 Constructing Extended Preference Pairs

Based on the estimated posterior probabilities, we now can extend the Definition 1 of *preference pair* even when (v,v') has the same binary preferences. Firstly, we utilize an aggregation function to readout from the estimated probability of user clicks P_{click}^u and the probability of the item being clicked P_{clicked}^v, and finally obtain a scalar as the final posterior probability as:

$$P_{\text{agg}}=\texttt{agg}(P_{\text{click}}^u,P_{\text{clicked}}^v), \tag{9}$$

where the aggregation function $\texttt{agg}(\cdot)$ could be any function such as the maximum or the average function.

The final augmented preference score z is then obtained by summing the pooled probability and the original click label:

$$z=P_{\text{agg}}+y. \tag{10}$$

We can clearly see that even if the pair $(u,v),(u,v')$ has the same binary preferences, it is less likely that they also have the same posterior probabilities at the same time. In this way, we are able to extend the definition of the preference pair in Section 3.1 as follows:

Definition 2 (Extended preference judgement pair). *An extended preference judgement pair is define as:* $(u,v)\succcurlyeq(u,v')$*, where* $z^{u,v}$ *is greater than* $z^{u,v'}$*.*

In summary, our proposed BBP ranking framework is able to construct pairs even with the same preferences (*i.e.,* 1 vs. 1 or 0 vs. 0). Although our model does not explicitly output a ranking score, this approach is consistent with our goal of having user items with higher ranking scores (*i.e.,* clicked or with higher posterior probability) associated with a higher predicted output probability.

4.3 Beyond Binary Preference Ranking

This ranking-based approach enables us to go one step further and perform global context ranking. In other words, when building the dataset, we do not necessarily have to consider the same context (such as the clicked item and the unclicked item of the same user). Instead, we can rank the correlations between any user and item pairs. This further improves the number of training samples and the training flexibility of our model, as we no longer need to construct a loss function based on the same user. By incorporating global context ranking, our BBP framework can better generalize and adapt to various ranking scenarios, enhancing its performance in real-world CTR prediction tasks.

During training, we construct a global context ranking loss. Specifically, in each batch of a training set with a size of B, we compute the difference between their scores and obtain the label of the partial order relationship:

$$\Delta z^{(u,v)\succcurlyeq(u',v')}=z^{(u,v)}-z^{(u',v')}, \tag{11}$$

where (u,v) and (u',v') are two user item pairs and $z^{(u,v)}$ and $z^{(u',v')}$ are their corresponding scores.

We then define an indicator function $\mathbb{I}(\cdot)$ that returns one if the difference is greater than 0, and 0 otherwise:

$$\mathbb{I}(\Delta z^{(u,v)\succcurlyeq(u',v')})=\begin{cases}1, & \text{if } \Delta z^{(u,v)\succcurlyeq(u',v')}>0\\ 0, & \text{otherwise}\end{cases}. \tag{12}$$

We can also compute the difference of the corresponding model output:

$$\Delta s^{(u,v)\succcurlyeq(u',v')}=\sigma(f(\mathbf{x}^{u,v})-f(\mathbf{x}^{u',v'})) \tag{13}$$

After that, we check whether the partial order relationship (difference) between the model's output $f(\mathbf{x}^{u,v})$ is consistent with the partial order relationship label. The calculation is given by:

$$\mathcal{L}_{\text{rank}}=-\frac{1}{M}\sum_{(u,v)\succ(u',v')}^{M}\log\left(\sigma\left(\Delta s^{(u,v)\succcurlyeq(u',v')}\cdot\mathbb{I}(\Delta z^{(u,v)\succcurlyeq(u',v')})\right)\right), \tag{14}$$

where $M=\binom{B}{2}$ is the size of preference pairs, B is the batch size, and $\sigma(\cdot)$ is the sigmoid function. We compare all possible $\binom{B}{2}$ different user item pairs' partial order $(u,v)\succ(u',v')$ in this batch.

Following the SOTA methods [3, 44], we optimize the binary cross entropy loss per batch as the calibration loss $\mathcal{L}_{\text{cal}}$:

$$\mathcal{L}_{\text{cal}}=-\frac{1}{B}\sum_{(u,v,y)}y^{u,v}\log\sigma(s)^{u,v}+(1-y^{u,v})\log(1-\sigma(s)^{u,v}). \tag{15}$$

As a multi-task framework, the final loss is the weighted sum of the calibration loss and the ranking loss:

$$\mathcal{L}_{\text{total}} = \lambda \mathcal{L}_{BCE} + (1 - \lambda)\mathcal{L}_{\text{rank}}, \tag{16}$$

where $\lambda \in [0, 1]$ is the weighting factor hyper-parameter. Empirically, we find that λ is a stable hyper-parameter that is free from time-consuming fine-tuning. We provide a TensorFlow-style pseudo code of the Beyond Binary Preference (BBP) training paradigm in Algorithm 1.

To better build the intuition of our BBP and facilitate further application and research, we provide the source code of BBP on https://github.com/AlvinIsonomia/BBP.

4.4 Computational Complexity Analysis

We provide in detail computational complexity analysis in both theory and deployed systems running time.

Firstly, let us have N records in our dataset. As shown in Sec. 4.1, the pre-computing has three steps. The first step is averaging the whole dataset, so the complexity is $O(N)$. The second step uses the fixed-point iteration method to update the personalized beta distribution. Note that we set a constant maximum iteration step, and the total length is also N for the impression (I) and click (C) sequence, respectively. So, the complexity is also $O(N)$. The third step estimates the posterior probabilities for all users and items. Note that the complexity for each user or item is $O(1)$, and the number of users and items is always less than the number of total records N (because records come from the combination of users and items). So, in summary, the computational complexity for our pre-computing is $O(N)$, which means our method is linearly scalable with the amount of data.

In practice, we record the time used in our large-scale industrial datasets of Tencent games. In our deployed instance, it takes around 31 minutes to pre-compute 38 million interaction records, which is fast enough to pre-compute the beta distributions for millions of records daily. Moreover, given that the training time cost is around 6 7 hours, the pre-computing takes only less than 8% of the total time (*i.e.*, pre-computing + training).

5 EXPERIMENTS ON PUBLIC DATASETS

5.1 Experimental Settings

The experiments conducted in this study on public datasets are designed to address the following three research questions:

RQ1: Does the BBP framework outperform the existing state-of-the-art (SOTA) models in CTR prediction tasks?

RQ2: Does BBP's breaking ties in CTR improve the performance?

RQ3: Does the global contexts further enhance BBP's performance?

Experimental environments. We conduct the experiments on a machine equipped with a Tesla V100 GPU with 16 GB GPU memory, 22 CPU cores, and 90 GB shared CPU memory with TensorFlow [1].

Datasets. We use three subsets of the Amazon datasets [35]: Clothing, Music, and Electronics. Following the processing in [28], we treated the samples with ratings greater than four as positive and negative. Meanwhile, we only use samples where the items have a picture and more than five comments. The features we used include image features (image embeddings extracted by the pre-trained VGG-16 [45] feature extractor), text features (review embeddings extracted by the pre-trained BERT[10]), user profile (average vector of user historical review embeddings), and item attribute (average vector of item history review embeddings).

To ensure a robust evaluation of our model, we split the dataset based on the user's ID into a 7:1:2 ratio for the training, validation, and test sets. This procedure guarantees that all users in the validation and test sets are not present in the training set, thus providing an unbiased assessment of the model's performance on unseen users. Furthermore, to avoid the risk of label leakage, we constructed Bernoulli trials only on the training set and performed Bayesian smoothing. This step ensures models' generalization and effectiveness in real-world CTR prediction tasks. By adopting this user-based splitting strategy and incorporating Bayesian smoothing, we can better understand the model's generalization capability and performance in CTR prediction tasks.

5.2 Hyper-parameter settings

We use Adam as the optimizer. We set the batch size to 256 and the learning rate to 1e-3. We clip the gradients where the norm is larger than 5. During training, for every 100 iterations, we compute the AUC on the validation set and finally report the test AUC and LogLoss with the checkpoint with the highest validation AUC score. For the AutoInt model, we use ReLU as the activation function, set the embedding dimension for each feature to 8, the number of heads to 2, the dimension of the heads to 4, the attention dropout rate to 0.1, and the number of hidden units to $[512, 256, 64]$.

Specifically, we simply set λ to 0.5 in Eq. (16) to take the balance between calibration and ranking.

Compared methods. We compare our BBP with five SOTA ranking-based methods: Bayesian Personalized Ranking (BPR) [40], Smooth-AUC (SAUC) [47], Regression Compatible Ranking (RCR) [3], Scale Calibration (SC) [51], and Joint optimization of Ranking and Calibration (JRC) [44] as the training methods. Since this paper focuses on the training paradigm, *i.e.,* loss function, we choose AutoInt [46] as our backbone model for experiments. AutoInt is a mainstream self-attention-based neural network for recommendation systems, which can find low-dimensional representations and efficient combinations of sparse, high-dimensional features. For a fair comparison, we keep the model capacities the same for all methods. JRC requires modifying the output layer to output two logits instead of the default one.

5.3 Metrics

In this study, we choose LogLoss as the calibration metric and AUC as the ranking metric. These metrics are selected because we can only obtain binary preference information in the training datasets. The equations and notation descriptions for LogLoss and AUC are as follows:

LogLoss. LogLoss measures the performance of a classification model by calculating the logarithm of the likelihood of the true labels given the predicted probabilities. For a binary classification

Algorithm 1: A TensorFlow-style pseudo code of Beyond Binary Preference (BBP) training paradigm.

```
import tensorflow as tf
import tensorflow.keras.backend as K
# B: batch size, label_list: [B, 1], score_list: [B, 1].
# Feed forward computation to get the prediction and compute BCE loss with binary click label.
logits = model(inputs)
pred = K.sigmoid(logits)
bce_loss = tf.keras.losses.BinaryCrossentropy(label_list, pred)
# Compute Beyond Binary Preference (BBP) ranking loss with pre-computed float ranking score.
BBP_matrix = pred - tf.transpose(pred)
BBP_label = tf.cast(tf.sign(score_list - tf.transpose(score_list)), dtype="float")
BBP_loss = -K.mean(K.log(K.sigmoid(BBP_matrix * BBP_label)))
totoal_loss = bce_loss + BBP_loss
```

Table 1: Statistics of the public user-item recommendation datasets.

	#Users	#Items	#Interactions
Clothing	39,387	23,033	278,651
Music	5,541	3,568	64,704
Electronics	192,403	63,001	1,688,104

problem, the LogLoss is defined as:

$$\text{LogLoss} = -\frac{1}{T_{\text{test}}} \sum_{i=1}^{T_{\text{test}}} \left[y^{u_i,v_i} \log(p^{u_i,v_i}) + (1 - y^{u_i,v_i}) \log(1 - p^{u_i,v_i}) \right], \tag{17}$$

where T_{test} is the number of test impressions, y^{u_i,v_i} is the true label of the i-th impression, and p^{u,v_i} is the predicted CTR of the i-th impression.

AUC. The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) is a widely used metric for evaluating the ranking performance of a binary classifier. The AUC is defined as:

$$\text{AUC} = \frac{1}{|T_{\text{P}}| \times |T_{\text{N}}|} \sum_{i=1}^{|T_{\text{P}}|} \sum_{j=1}^{|T_{\text{N}}|} \left\{ \mathbb{I}\left[(s^{u_i,v_i} - s^{u_j,v_j}) > 0 \right] + \frac{1}{2} \mathbb{I}\left[(s^{u_i,v_i} - s^{u_i,v_i}) = 0 \right] \right\}, \tag{18}$$

where T_{P} and T_{N} represent the positive and negative impression set for the test dataset, respectively, and $|\cdot|$ denotes the set size. s^{u_i,v_i} indicates the preference score by user u_i on item v_i. $\mathbb{I}[\cdot]$ is an indicator function that takes the value one if the condition inside the brackets is true and 0 otherwise.

5.4 Results in Public Datasets (RQ1)

To answer **RQ1** and validate the performance of different training paradigms for click-through rate (CTR) prediction, we conduct experiments on three public datasets: Clothing, Music, and Electronics. As shown in Table 2, our proposed BBP method consistently outperforms all pointwise, pairwise, and listwise alternatives across all datasets regarding LogLoss and AUC metrics.

The pairwise training paradigms, including BPR [40] and SAUC [47], struggle with calibration and achieve limited ranking performance. Their LogLoss values are significantly worse than the Binary Cross Entropy (BCE) baseline, as indicated by the red color. Since BPR and SAUC ignore the calibration loss during training, the predicted score keeps drifting and can raise a NaN (Not a Number) LogLoss for calculating log(0). Although they show some improvements in AUC compared to the baseline, their performance is still inferior to the listwise and our BBP methods. Since pairwise baselines including BPR and SAUC are not designed for calibration, we modify them by using calibrated loss. Specifically, we add the pairwise losses with a weighted BCE loss, similar to Eq. (16). We have conducted additional experiments after fine-tuning the weight of BCE loss as our BBP does. The results show that BBP also significantly outperforms calibrated pairwise competitors for all metrics.

On the other hand, listwise methods handle calibration and ranking simultaneously due to their multi-task design. They achieve better performance than the pairwise methods, with SC [51] and JRC [44] often being the best-performing alternatives to our method. However, their performance still falls short compared to BBP, as the significant p-values indicate.

With its beyond binary preference ranking, our BBP method stands out by achieving the best performance across all datasets and metrics. The LogLoss values are the lowest, and the AUC values are the highest among all methods, indicating that BBP provides the best calibration and ranking. The significant p-values further confirm the superiority of BBP over all competitors. As shown in Eq. (9), BBP can leverage different functions to aggregate the possibilities from user and item. We take the average and maximum functions as examples, namely BBP-Average and BBP-Maximum. We find that the performance with these two aggregation functions has no significant difference (p-values > 0.05).

In conclusion, the experimental results demonstrate the effectiveness of our BBP method in CTR prediction. It provides significant advantages over all SOTA competitors, especially the pairwise techniques, and sets a new benchmark for CTR prediction.

5.5 Ablation Study (RQ2 and RQ3)

We conduct an ablation study to investigate the contributions of BBP's ability to break ties and the global context ranking to its performance. We implement three variants of our proposed method, and all variants implement BCE as the calibration loss: **Case 1:** The ranking loss in Eq.(14) only compare item v from the same user u with their binary click label y. **Case 2:** The ranking loss in Eq.(14) only compare item v from the same user u with their estimated

Table 2: Comparison of overall performance for CTR prediction based on different training paradigms, with average results reported over five random seeds. LogLoss ↓ is a calibration metric where lower values are better. AUC ↑ is a ranking metric, with higher values preferable. The underlined values highlight the best performance, excluding our proposed method. The p-value is obtained through a t-test between our method and the best-performing alternative. Bold values indicate that our BBP method significantly outperforms all competitors (determined by the significance at $p < 0.05$), no matter which implemented aggregation function. Results significantly worse than the Binary Cross Entropy (BCE) baseline are denoted in red, while those significantly better are marked in green. The color significance is also based on the $p < 0.05$ threshold.

Dataset		Clothing		Music		Electronics	
Metric		LogLoss ↓	AUC ↑	LogLoss ↓	AUC ↑	LogLoss ↓	AUC ↑
Pointwise	BCE	0.6184	0.7695	0.6414	0.7246	0.6522	0.7242
Pairwise	BPR (UAI'09)	17.4783	0.7754	NaN	0.7307	NaN	0.7294
	BPR + BCE	0.6289	0.7720	0.6851	0.7286	0.6451	0.7282
	SAUC (SIGIR'22)	8.3426	0.7785	0.7292	0.7371	NaN	0.7361
	SAUC + BCE	0.6028	0.7792	0.6712	0.7338	0.6558	0.7368
Listwise	RCR (CIKM'23)	0.6523	0.7808	0.6846	0.7398	0.6540	0.7409
	SC (KDD'22)	0.5636	0.7839	0.6590	0.7418	0.6564	0.7508
	JRC (KDD'23)	0.6187	0.7886	0.6487	0.7474	0.5955	0.7447
Ours	BBP-Average	**0.5413**	**0.7965**	**0.5996**	**0.7562**	**0.5754**	**0.7590**
	BBP-Maximum	**0.5471**	**0.7942**	**0.5952**	**0.7541**	**0.5747**	**0.7566**
	p-value	1.23E-03	2.35E-02	7.79E-04	4.41E-02	3.30E-02	1.00E-03

score s in Eq.10. **Case 3 (BBP):** The ranking loss Eq.(14) compare all $u - v$ pairs in each sampled batch with their estimated score s.

To address **RQ2**, *i.e.,* to show that even without the augmentation of global contexts, the BBP framework can improve model performance with its ability to break ties using Bayesian probabilities. We can compare the average performance between Case 1 and Case 2. To further address **RQ3**, *i.e.,* to confirm that the global context ranking can further improve model performance, we can compare Case 2 with Case 3 (our BBP). As shown in Figure 3, the experimental results demonstrate that the performance of the models follows the order: Case 3 (BBP) > Case 2 > Case 1. The improvement observed between Case 3 and Case 2 is smaller than between Case 2 and 1. This indicates that the primary enhancement of BBP is due to breaking many binary ties, and the score estimate provided by the Bayesian probabilities is meaningful (RQ2). The global context also contributes to the improvement of BBP (RQ3).

Moreover, as illustrated in Algorithm 1 in the appendix, the global context enables on-the-fly ranking with naive pointwise training, making it easier to deploy. Unlike state-of-the-art methods [3, 40, 44, 47, 51], we do not need to be concerned about whether the training batch is under the same context. This flexibility allows BBP to adapt to various ranking scenarios more effectively, enhancing its performance in real-world CTR prediction tasks.

In summary, the ablation study demonstrates that BBP's ability to break ties and the incorporation of global contexts contribute to performance improvement. The primary enhancement comes from breaking ties using Bayesian probabilities. At the same time, incorporating global contexts further boosts performance and offers greater flexibility and adaptability in various ranking scenarios.

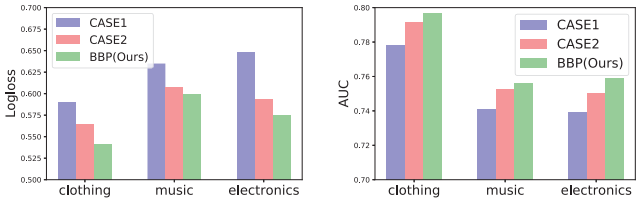

Figure 3: Ablation study results on Amazon datasets.

6 INDUSTRIAL DEPLOYMENT

6.1 In-Game Friend Recommendation

We have deployed our BBP training paradigm model and its top two competitors, SC and JRC, on two Tencent games: a mobile *battle royale game* denoted by X and a *casual party game* denoted by Y.

CTR Prediction for Friend Recommendation. The deployed methods serve for friend recommendation tasks in both real-world games. The friend recommendation is a resident module in both games X and Y. When a player u (inviter) accesses the recommendation module to find some other players to become in-game friends, u sees an ordered list of interacted non-friend players. This generates an *impression* record in the recommendation logs, and u can decide to click on a non-friend player v (invitee) or not. Inviter u can click one or multiple invitees. Once inviter u clicks on a recommended invitee v, it sends a friend request and generates a *click* record in the recommendation logs. The request requires approval; the clicked invitee v can decide to accept or reject it. If u and v successfully become in-game friends, the recommendation module generates a *success* record. In summary, the CTR prediction model is used to infer the probability that an inviter u clicks on an invitee v, and then the recommendation list is generated according to the descending order of predicted CTR.

Offline dataset collection. We gather the one-week logs of log-in players, and we consider the inviters u who clicked at least one invitee v as the positive examples in the beforehand recommendation module. We treat the clicked invitees as positive samples and the unclicked ones as negative samples. If an inviter u clicks on an invitee v at least once, the interaction is viewed as a positive sample, and we will filter out repeated $u - v$ pairs. Each dataset contains the logs of the in-game recommendation modules of all logged-in players within seven contiguous days, along with their in-game features as follows:

- Players' profile data, including all related in-game attribute data of a player, such as the game level, gender, online time, *etc.*
- Pairwise interaction features, including the interaction data between the u and v, such as the number of common friends, and the number of common matches *etc.*
- Players' social network properties (*i.e.,* the existing in-game social network properties), including the local network structure, the number of friends, *etc.*

All raw features are standardized using min-max normalization before being fed into the CTR prediction models. The statistics of games X and Y can be found in Table 3.

6.2 Offline Evaluation

We first train the models with datasets X and Y offline to evaluate their performance. We split the dataset by inviter u's id into a 7:1:2 ratio for training, validating, and testing datasets. We perform the Bayesian smoothing only on the train set like Section 5.1 to avoid label leakage. For the evaluation metrics, since we are interested in whether the inviter u would click the invitee v in the friend recommendation list, we report Hit rate@N and NDCG@N, where N is 10, 20, and 50 to show the performance of models on different positions. As summarized in Table 4, the offline evaluation results demonstrate the effectiveness of our BBP method compared to the competing methods in all metrics. This observation justifies that our method is more effective than the competitors for ranking the player list in real-world friend recommendation tasks.

6.3 Online Evaluation

To evaluate the performance of the various methods in an online setting, we additionally conduct online A/B tests on both games [19, 30–32, 53–56]. The friend recommendation module in each game is updated with the deployed models, and their performance is monitored over one week (from 2024/01/19 - 2024/01/25). We evaluate BBP, SC, and JRC in the friend recommendation task using three metrics: *(i)* Click Rate, which is the proportion of *click* user among *impression* records; *(ii)* Approval Rate, the proportion of *success* approval among *click* invitations; and *(iii)* overall Friend Rate, the proportion of *success* approval among *impression* records. The overall Approval Rate is the product of the Click Rate and the Approval Rate after invitations are sent, *i.e.*, Friend Rate = Click Rate × Approval Rate.

The online evaluation results in Table 5 demonstrate that our proposed BBP method consistently outperforms the state-of-the-art (SOTA) methods SC and JRC in terms of Click Rate, Approval Rate and Friend Rate for both games X and Y. Specifically, considering the definition of Friend Rate, the BPR we proposed can increase the number of new friends for the in-game friend recommendation module by more than 10.28% compared to SOTA methods. *The proposed BBP method has now been fully deployed into the in-game friend recommendation systems of Tencent Games X and Y.*

Table 3: Statistics of real-world game datasets.

GAME	#Users	#Interactions	#Features
X	3,240,444	38,024,787	74
Y	754,776	20,906,940	229

Table 4: The results of the offline evaluation.

GAME X	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
Baseline	0.3202	0.2422	0.3503	0.2498	0.3743	0.2547
SC	0.3224	0.2483	0.3505	0.2555	0.3735	0.2601
JRC	0.3157	0.2373	0.3460	0.2450	0.3717	0.2502
Ours	**0.3287**	**0.2581**	**0.3545**	**0.2580**	**0.3765**	**0.2624**
GAME Y	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
Baseline	0.3817	0.2678	0.4523	0.2869	0.5005	0.2944
SC	0.3879	0.2715	0.4547	0.2889	0.5016	0.2977
JRC	0.3873	0.2707	0.4550	0.2882	0.5015	0.2959
Ours	**0.3897**	**0.2728**	**0.4561**	**0.2900**	**0.5022**	**0.2985**

Table 5: The performance lift of the proposed BBP compared with the SOTA methods.

Metric	Click Rate		Approval Rate		Friend Rate	
GAME	X	Y	X	Y	X	Y
vs. SC	3.94%	2.93%	4.41%	9.26%	11.59%	18.92%
vs. JRC	10.90%	3.00%	7.73%	4.33%	22.82%	10.28%

These results indicate that the BBP method is highly effective in enhancing the friend recommendation module in both games, leading to higher user engagement and satisfaction. The superior performance of BBP in real-world scenarios further validates its effectiveness and applicability in CTR prediction and downstream recommendation tasks.

7 CONCLUSION

This paper presents the Beyond Binary Preference (BBP) training framework for CTR prediction, which addresses the limitations of insufficient training pairs caused by binary preference. BBP estimates probability distributions for users and items through Bayesian smoothing. The estimated probabilities then eliminate the ties as augmented labels. BBP also design a global context ranking loss to augment more preference pairs further during training.

We have demonstrated that the BBP framework outperforms all current state-of-the-art competitors in ranking and calibration capabilities through extensive offline experiments and online A/B tests on various user-item recommendation datasets and in-game friend recommendation scenarios. The implementation code for BBP has been provided as supplementary materials to encourage further development by the research community.

ACKNOWLEDGEMENT

This work is supported by the National Nature Science Foundation of China under Grant 62176155 and Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*. USENIX Association, 265–283.

[2] Richard A Askey and Ranjan Roy. 2010. Gamma function.

[3] Aijun Bai, Rolf Jagerman, Zhen Qin, Le Yan, Pratyush Kar, Bing-Rong Lin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2023. Regression Compatible Listwise Objectives for Calibrated Ranking with Binary Relevance. In *CIKM*. ACM, 4502–4508.

[4] Jose M Bernardo. 1976. Algorithm AS 103: Psi (digamma) function. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 25, 3 (1976), 315–317.

[5] Bryan Brancotte, Bo Yang, Guillaume Blin, Sarah Cohen Boulakia, Alain Denise, and Sylvie Hamel. 2015. Rank aggregation with ties: Experiments and Analysis. *Proc. VLDB Endow.* 8, 11 (2015), 1202–1213.

[6] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. 2020. A Stochastic Treatment of Learning to Rank Scoring Functions. In *WSDM*. ACM, 61–69.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML (ACM International Conference Proceeding Series, Vol. 227)*. ACM, 129–136.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. (2016), 7–10.

[9] Daniel Cohen, Bhaskar Mitra, Oleg Lesota, Navid Rekabsaz, and Carsten Eickhoff. 2021. Not All Relevance Scores are Equal: Efficient Uncertainty and Calibration Modeling for Deep Retrieval Models. In *SIGIR*. ACM, 654–664.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[11] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognit. Lett.* 27, 8 (2006), 861–874.

[12] Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-Scale Bayesian Click-Through rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In *ICML*. Omnipress, 13–20.

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1321–1330.

[14] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An Embedding Learning Framework for Numerical Features in CTR Prediction. In *KDD*. ACM, 2910–2918.

[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. (2017), 1725–1731.

[16] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer.

[17] Kun He, Fatih Çakir, Sarah Adel Bargal, and Stan Sclaroff. 2018. Hashing as Tie-Aware Learning to Rank. In *CVPR*. Computer Vision Foundation / IEEE Computer Society, 4023–4032.

[18] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD@KDD*. ACM, 5:1–5:9.

[19] Shixun Huang, Wenqing Lin, Zhifeng Bao, and Jiachen Sun. 2022. Influence Maximization in Real-World Closed Social Networks. *Proc. VLDB Endow.* 16, 2 (2022), 180–192.

[20] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *RecSys*. ACM, 169–177.

[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[22] Norman Lloyd Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. 1972. *Continuous multivariate distributions*. Vol. 7. Wiley New York.

[23] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *RecSys*. ACM, 43–50.

[24] Norman Knyazev and Harrie Oosterhuis. 2023. A Lightweight Method for Modeling Confidence in Recommendations with Learned Beta Distributions. In *RecSys*. ACM, 306–317.

[25] Ranganath Krishnan and Omesh Tickoo. 2020. Improving model calibration with accuracy versus uncertainty optimization. In *NeurIPS*.

[26] Tien-Duy B. Le, David Lo, Claire Le Goues, and Lars Grunske. 2016. A learning-to-rank based fault localization approach using likely invariants. In *ISSTA*. ACM, 177–188.

[27] Cheng Li, Yue Lu, Qiaozhu Mei, Dong Wang, and Sandeep Pandey. 2015. Click-through Prediction for Advertising in Twitter Timeline. In *KDD*. ACM, 1959–1968.

[28] Xiang Li, Chao Wang, Jiwei Tan, Xiaoyi Zeng, Dan Ou, and Bo Zheng. 2020. Adversarial Multimodal Representation Learning for Click-Through Rate Prediction. In *WWW*. ACM / IW3C2, 827–836.

[29] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *KDD*. ACM, 1754–1763.

[30] Wenqing Lin. 2019. Distributed Algorithms for Fully Personalized PageRank on Large Graphs. In *WWW*. ACM, 1084–1094.

[31] Wenqing Lin. 2021. Large-Scale Network Embedding in Apache Spark. In *KDD*. ACM, 3271–3279.

[32] Wenqing Lin, Feng He, Faqiang Zhang, Xu Cheng, and Hongyun Cai. 2020. Initialization for Network Embedding: A Graph Partition Approach. In *WSDM*. ACM, 367–374.

[33] Tie-Yan Liu. 2010. Learning to rank for information retrieval. In *SIGIR*. ACM, 904.

[34] Boxiang Lyu, Zhe Feng, Zachary Robertson, and Sanmi Koyejo. 2023. Pairwise Ranking Losses of Click-Through Rates Prediction for Welfare Maximization in Ad Auctions. In *ICML (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 23239–23263.

[35] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*. ACM, 43–52.

[36] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *KDD*. ACM, 1222–1230.

[37] Aditya Krishna Menon, Xiaoqian Jiang, Shankar Vembu, Charles Elkan, and Lucila Ohno-Machado. 2012. Predicting accurate probabilities with a ranking loss. (2012).

[38] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. 2021. Revisiting the Calibration of Modern Neural Networks. (2021), 15682–15694.

[39] Steffen Rendle. 2010. Factorization Machines. In *ICDM*. IEEE Computer Society, 995–1000.

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. AUAI Press, 452–461.

[41] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. ACM, 521–530.

[42] Simo Särkkä. 2013. Bayesian Filtering and Smoothing. In *Institute of Mathematical Statistics textbooks*.

[43] D. Sculley. 2010. Combined regression and ranking. In *KDD*. ACM, 979–988.

[44] Xiang-Rong Sheng, Jingyue Gao, Yueyao Cheng, Siran Yang, Shuguang Han, Hongbo Deng, Yuning Jiang, Jian Xu, and Bo Zheng. 2023. Joint Optimization of Ranking and Calibration with Contextualized Hybrid Model. In *KDD*. ACM, 4813–4822.

[45] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2015).

[46] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *CIKM*. ACM, 1161–1170.

[47] Shuang Tang, Fangyuan Luo, and Jun Wu. 2022. Smooth-AUC: Smoothing the Path Towards Rank-based CTR Prediction. In *SIGIR*. ACM, 2400–2404.

[48] Xuerui Wang, Wei Li, Ying Cui, Ruofei Zhang, and Jianchang Mao. 2011. Click-through rate estimation for rare events in online advertising. In *Online multimedia advertising: Techniques and technologies*. IGI Global, 1–12.

[49] Xinlong Weng. 1991. Fixed point iteration for local strictly pseudo-contractive mapping.

[50] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML (ACM International Conference Proceeding Series, Vol. 307)*. ACM, 1192–1199.

[51] Le Yan, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. Scale Calibration of Deep Ranking Models. In *KDD*. ACM, 4300–4309.

[52] Yanwu Yang and Panyu Zhai. 2022. Click-through rate prediction in online advertising: A literature review. *Inf. Process. Manag.* 59, 2 (2022), 102853.

[53] Shiqi Zhang, Yiqian Huang, Jiachen Sun, Wenqing Lin, Xiaokui Xiao, and Bo Tang. 2023. Capacity Constrained Influence Maximization in Social Networks. In *KDD*. ACM, 3376–3385.

[54] Shiqi Zhang, Jiachen Sun, Wenqing Lin, Xiaokui Xiao, Yiqian Huang, and Bo Tang. 2024. Information Diffusion Meets Invitation Mechanism. In *WWW*. ACM, 383–392.

[55] Shiqi Zhang, Jiachen Sun, Wenqing Lin, Xiaokui Xiao, and Bo Tang. 2022. Measuring Friendship Closeness: A Perspective of Social Identity Theory. In *CIKM*. ACM, 3664–3673.

[56] Xingyi Zhang, Shuliang Xu, Wenqing Lin, and Sibo Wang. 2023. Constrained Social Community Recommendation. In *KDD*. ACM, 5586–5596.

[57] Piplong Zhao, Ou Wu, Liyuan Guo, Weiming Hu, and Jinfeng Yang. 2016. Deep learning-based learning to rank with ties for image re-ranking. (2016), 452–456.
[58] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*. ACM, 287–294.
[59] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*. ACM, 1059–1068.
[60] Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2008. Learning to rank with ties. In *SIGIR*. ACM, 275–282.
[61] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *CIKM*. ACM, 2759–2769.
[62] Xiaofeng Zhu and Diego Klabjan. 2020. Listwise Learning to Rank by Exploring Unique Ratings. In *WSDM*. ACM, 798–806.

A EMPIRICAL STUDY ABOUT USER BEHAVIOR

A.1 The Beta Distribution Assumption

Click-through rate (CTR) is the probability of a user clicking on an item. Considering the binary nature of the clicking behavior (click or not), it can be modeled as a Bernoulli trial with an underlying probability r. To further validate our assumption about beta distribution, we conduct the Kolmogorov-Smirnov test and other data-based analyses to show that the assumption is reasonable in our deployed system.

Firstly, we conduct a Kolmogorov-Smirnov test on the active users' click rates and our estimated beta distribution. The H0 is the click rate that obeys our estimated beta distribution. The result shows that the K-S statistic is 0.01641195 and the p-value is 0.13383894. Given that the p-value > 0.05, we accept the H0 that the users' click rates obey a beta distribution.

Secondly, we provide a Quantile-Quantile plot to show the relationship between our estimated beta distribution (Theoretical Quantiles) with real-world players' behavior (Sample Quantiles).

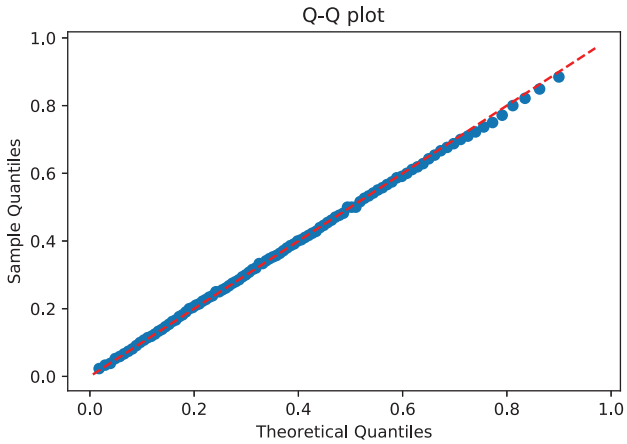

Figure 4: The Q-Q plot of user click rate in our deployed system.

As shown in Fig. 4, the Quantile-Quantile plot shows that the real-world players' behavior fits nicely with our estimated beta distribution.

A.2 The Confidence Level Distribution with Number of Training Samples

BBP may fail to estimate personalized distributions with high confidence levels for users with limited samples in the training set. Intuitively, the Bayesian smoothing drives the posterior probabilities from the initialized beta distribution to a more personalized one. If the user does not have enough samples, their posterior probabilities will be similar to the posterior probability from our initialized beta distribution. In the extreme case, if the user does not have any historical data, his posterior probability will be derived from the initialized beta distribution. As we treat impressions and clicks as visible variables of a user that obey a Binomial distribution with an underlying probability r. Our Bayesian smoothing aims to estimate personalized posterior probabilities for the Binomial distribution.

We provide further tests on the visible user behavior, *i.e.*, the click obeying the personalized distribution estimated by BBP. We test active players in our deployed system. The result shows that 70.43% of the total players obey their personalized distribution of BBP, determined by the significance at p-value > 0.05. We agree with you that the number of samples of users is important because we find that less than 30% of players fail in the test of their limited records in real-world applications. We also provide a Boxplot to show the confidence of the personalized distribution among players, illustrating that most players obey their personalized distribution.

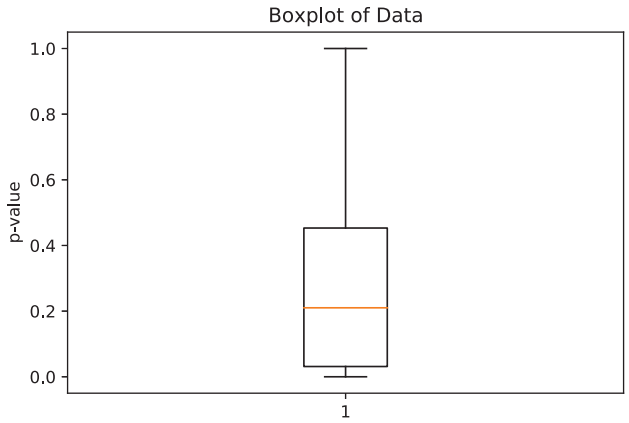

Figure 5: The confidence level distribution of user obeying the BBP's output estimation in our deployed system.

Due to the complexity of real-world deployment scenarios, it is unavoidable that some users with insufficient data (such as *cold-start* players who have just joined the game). However, BBP has already outperformed all competitors in our online and offline experiments.

B IMPACT OF HYPER-PARAMETER

Empirically, we find that λ is a stable hyper-parameter that is free from time-consuming fine-tuning. Intuitively, as we design both calibration loss in Eq. (15) and ranking loss in Eq. (14) in a sigmoid-then-log style, the scale of the two losses are similar. To show our intuition, we perform a grid search of λ in [0.1, 0.3, 0.5, 0.7, 0.9] on the Music dataset and report the average LogLoss and AUC metric *w.r.t.* the different λ.

As shown in Figure 6 and Figure 7, when we increase the λ, the total loss prefers more calibration during training, which leads to

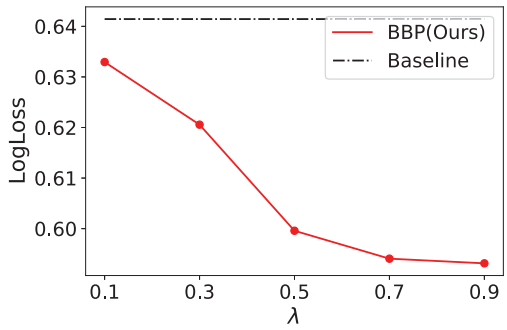

Figure 6: Calibration Performance (LogLoss) of BBP *w.r.t.* different hyper-prameter λ on the Music dataset.

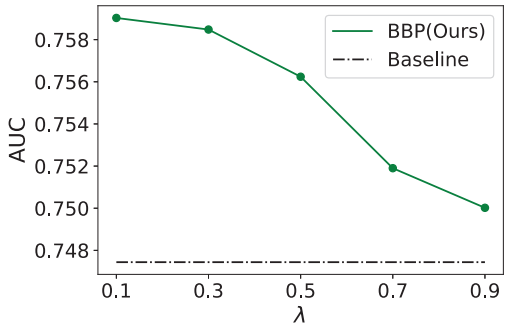

Figure 7: Ranking Performance (AUC) of BBP *w.r.t.* different hyper-prameter λ on the Music dataset.

lower LogLoss and AUC at the same time. However, in such a large range, BBP continues to outperform the best baseline (represented with dashed lines).

This analysis demonstrates that the hyper-parameter λ is stable and does not require extensive fine-tuning, which simplifies the training process and makes BBP more practical for real-world applications. Furthermore, the robust performance of BBP across different values of λ highlights its effectiveness in balancing calibration and ranking losses, resulting in superior CTR prediction performance compared to the baselines.