A Unified Recommendation Model for Features Summarization

Vito Bellini[1], Zhan Shi[2], Huseyin Yurtseven[1] and Emanuele Coviello[2]

[1]Amazon Music, Germany
[2]Amazon Music, USA

Abstract

Personalization is a key requirement for most customer experiences in a music streaming service, such as landing page, station song sequencing, or search. A common approach is to develop dedicated personalization ML models, one for each experience, that directly integrate with all the personalization signals alongside experience-specific signals. However, this is not scalable as it is costly for each product team to replicate improvements and experiment with new personalization features. It can result in inconsistencies across the various experiences and slow down adoption of new features in all the models.

We propose an approach to make it easier to consume many personalization features in multiple experiences at a low experimental cost. Specifically, we train a customer relevance score that incorporates all those personalization signals, and vend that score so that product teams can integrate it in their models instead of having to integrate with all the personalization signals directly. We validate the approach on the personalized home page of our service across different use-cases.

Keywords

Recommender Systems, Features Summarization, Model Consolidation

1. Introduction

Streaming services offer an array of experiences to let customers engage with vast catalogs of music content, including songs, albums, playlists, stations, etc. This can include visual experiences for browsing and discovering content, search, automatic stations/radios, daily playlists, and voice controlled experiences, to name a few. Whereas individual customer experiences each have unique experience optimization challenges, they all need personalization.

Recommender Systems (RSs) are designed to suggest content that is relevant to the customer and the specific experience. For instance, on the home page of a streaming service, RSs are used to select and rank the widgets to display on each slate on the page, and then to select and rank relevant content items for each widget in a personalized fashion. Recommending relevant content according to the user's preferences does not refer only to ranking but it extends to other tasks such top-1 recommendation where the RS has to select only one item to show to the user. These are two fundamentally different problems that require different techniques for effective resolution. For example learning-to-rank (LTR) algorithm with pair-wise objectives [1, 2, 3] widely adopted in ranking cannot be used for top-1 recommendation use-cases because they require pairs containing both a positive and negative samples whereas only a single positive or negative sample is available in top-1 use-case. Other differences involves the way of debiasing dataset for training ranking or top-1 use-cases. The former typically involves assumption on click-models [4, 5, 6] where the position based model is one of the most used and studied in the literature, and the latter involves propensities estimation to account for selection bias [7]. Finally, specific metrics to evaluate performances are used according to the use-case. Generally, ranking metrics are optimized for engagement on the home page while on the top-1 recommender where the user interface allows only playbacks, listening time is the gold metric to optimize. Beyond the distinction between ranking and top-1 selection, different content types entail different recommendation

MuRS 2025: 3rd Music Recommender Systems Workshop, September 22nd, 2025, Prague, Czech Republic
vitob@amazon.com (V. Bellini); shizhan@amazon.com (Z. Shi); hyurts@amazon.de (H. Yurtseven); emacov@amazon.com (E. Coviello)
0000-0003-4067-714X (V. Bellini)

requirements. For example, a playlist recommender seeks to offer a diverse yet relevant set of playlists to maximize aggregated utility, while a song recommender favors coherence to enable uninterrupted listening. Moreover, recommenders may be tailored to specific customer experience designs, such as emphasizing trending content or recent user behavior.

A common architecture for RSs in online streaming services follows a two-steps approach that consists in candidate generation and then re-ranking [8, 9]. Candidate generation determines what is eligible for an experience (for example entities matching a search query well enough, entities displayed in a visual recommender or considered for a voice request, a list of widgets to be arranged on a page, related artists to be displayed on the artist page), generally efficiently selecting a high coverage set of candidates from a large set of items (potentially millions). In the subsequent ranking step we want to optimize the ranking of these entities (or widgets with entities) in a way that is contextually relevant and personalized and meets the specific customer experience requirement of the particular widget.

It is common to develop dedicated ranking models one for each experience, trained on traffic from that experience and directly integrates with all the personalization signals plus experience-specific signals. However, the issue is that it takes a lot replicated effort for different product teams to integrate and experiment with the *same* set of personalization features. This can result in slow adoption of new or improved personalization features, inconsistencies across experiences, and reducing resources to solve other key challenges peculiar to individual products.

The observation motivating this work is that, irrespective of the actual experience, we believe that this ranking step using personalization and contextual signals is essentially the same ML problem across different experiences, with small differences that can be captured by contextual features and multi-task training. We hence propose a customer-item relevance score that is derived from a model that incorporates many personalization signals and is trained from feedback from several customer experiences. This simplifies the adoption of personalization features and reduces experimentation costs, since a product team can integrate in their ML ranking models the summary customer-item score instead of all the feature directly. In this paper, we present initial results validating the approach on a ranking use-case where we re-rank the widgets on the home page of music streaming service and on a top-1 use-case where we pick the best content from a limited pool to show in the first widget of the home page.

2. Related works

Traditional RSs require extensive manual feature engineering, with different personalization services often duplicating effort to integrate the same personalization signals. Recent work has focused on automating this process to reduce engineering overhead and improve consistency across systems.

Authors in [10] introduced AutoField, a differentiable approach for automating feature selection in deep recommender systems that reduces embedding parameters and inference time while maintaining performance. This work demonstrates how automated feature selection can create consistent feature sets across different recommendation surfaces. The challenge of feature interaction selection has been addressed through several automated approaches. Work presented in [11] proposes AutoFIS, a two-stage algorithm for automatic feature interaction selection that identifies important interactions while removing redundant ones. AutoInt, introduced in [12], uses self-attention mechanisms for automatic high-order feature interaction learning without manual design. These approaches directly support the concept of unified feature representations by automatically determining which features and interactions contribute most to overall relevance across multiple tasks. Our approach differs by providing a relevance score that eliminates the need for downstream teams to focus on feature selection and engineering.

While feature engineering automation reduces the complexity of feature selection and engineering, usually a single model is used to solve a specific task (use-case). Building separate models for each task or domain creates significant computational and manual maintenance costs as pointed out in [13, 14]. Efforts to mitigate this problem have been made in order to consolidate or unify models across use-cases. However, existing approaches require each team to integrate all personalization signals

directly into their models. This poses the risk of feature fragmentation across use-cases and the burden of feature selection and engineering. Authors in [15] introduced Multi-gate Mixture-of-Experts (MMoE), it uses multiple expert networks with task-specific gating to balance shared and task-specific learning. However, this approach still requires each product team to build and maintain their own unified model with all personalization signals, limiting its ability to reduce engineering overhead across teams. Our approach extends this concept by training a single model that produces a relevance score consumable by any downstream model, eliminating the need for teams to implement their own multi-task architectures.

Multi-domain recommendation refers to a scenario in which the RS provides recommendations in different domains where an overlapping between items across the domain exists. Conceptually similar, multi-task learning aims to learn from different tasks where tasks could be potentially very different to each other (e.g.: home page, search page, etc...) with a partial or non-existent feature overlap across the tasks. Recent works on Multi-Task and Multi-domain have shown encouraging results in leveraging knowledge learned from a task applied to another where no much training data is available or transferring knowledge from one domain to another. In [16] authors proposed to train a single model on multiple domains to be used for different tasks; this allows for compressing information in a unified model that enables knowledge transfer across domains and tasks. Multi-domain recommendation has also been studied in the setting of sequential recommendation [17], where characteristics of each domain are projected into a common domain space leveraged to transform item representation across domain spaces.

While significant progress has been made in individual components, automated feature engineering, unified architectures, and transfer learning, existing work fails to address the fundamental operational challenge in production personalization systems: the repeated engineering effort required when multiple teams integrate the same personalization signals. Our work directly addresses these gaps by proposing a unified customer relevance score that:

- Eliminates repeated feature integration: teams consume a single score instead of integrating many individual signals;
- Reduces engineering overhead: new personalization features are automatically incorporated into the unified score without requiring downstream changes;
- Maintains model flexibility: teams can integrate the relevance customer-item score into any model architecture while preserving experience-specific optimizations;

Unlike previous unified approaches that require each team to build and maintain their own multi-task models, our approach centralizes the complexity in a single, reusable component while enabling lightweight integration downstream.

3. Modeling customer relevance

In this paper we propose a customer-item relevance score that summarizes a rich set of personalization features and is trained using feedback from multiple widgets. In this section we describe the relevance model architecture and training, feature selection and engineering, and offline evaluation.

3.1. Relevance model

We want to build a model capable of representing the customer relevance towards a music entity (e.g., a song, playlist, album, etc.) and we want the predicted score to be interpretable where a high score represents high relevance for the customer towards an entity. Learning-to-rank algorithms do not fit perfectly in this scenario as they predict relative ordering instead of relevance scores. Therefore, we use reward regression which uses a point-wise loss function.

We train the relevance model with feedback collected from a variety of different widgets. Feedback $c_t(i)$ is binary with $c_t(i) \in \{0, 1\}$ and it represents a click occurring on an item i being shown to the customer for a request at time t. We frame the learning problem in a bandit setting since we

observe feedback only for the items that our production recommenders are actually showing to the customers. This introduces a form of bias in logged data called selection bias that needs to be taken into account to avoid training and evaluating a sub-optimal model. We evaluate adaptations of L2 [18] and Binary-Cross-Entropy (BCE) [19] losses that correct for such bias. We chose the former as it outperformed the latter through offline evaluations with extensive hyper-parameters optimization (HPO).

Our model is a multi-layer perceptron (MLP) trained on click feedback to predict a relevance score of an item for a given customer. The model optimizes the following objective for biased feedback as presented in [18]:

$$\mathcal{L}(w) = \sum_{t=1}^{\tau} \sum_{i} \hat{R}(i|x_t)^2 + \frac{c_t(i)}{p_t(i)}(c_t(i) - 2\hat{R}(i|x_t)) \tag{1}$$

where i denotes the item for which to compute the score for a ranking request at time t, $\hat{R}(i|x_t)$ represents a regression estimate for the item i given its feature representation x_t, $c_t(i)$ and $p_t(i)$ are the observed click and its expected examination probability respectively; where by examination probability we refer to the joint probability that the ranker ranks the item i in position k and the user observes i at rank k.

The relevance model optimizes the objective in Equation 1 using an online dataset collected under the production model. The dataset contains for each request at time t a list of items to rank where each item i is represented by a feature vector $x_t \in \mathbf{R}^d$ in the feature space $\mathcal{F} = \mathcal{F}_c \cup \mathcal{F}_p$ with $dim(c) + dim(p) = d$. The features set $\mathcal{F}_c$ represents temporal and contextual information features such as time, day, device type and others, while $\mathcal{F}_p$ represents customer preferences. The relevance model is trained only on the subset $\mathcal{F}_p$ which does not depend on contextual information. It is then used in real-time to infer relevance scores defined as $z = \hat{R}(x'_t)$ where x'_t represents the item description with features in $\mathcal{F}_p$ only and $\hat{R}(\cdot)$ is a non-linear transformation function learnt from the neural network regressor which optimizes Equation 1. The customer-item affinity score z is then used as input feature to a final ranker.

3.2. Features

Features representing customer preferences serve as input to the customer relevance model. First and foremost, we have features that directly capture the historical interactions of the customer with the music items and their artists, covering aspects such as the number of listens, likes, and follows. They also encompass non-personalized item information such as popularity or whether the item is a new release.

In order to capture user-item relationships on items that the customer had not interacted with in the past, we also leverage a collaborative filtering model that provides user and item embeddings. The model is trained on past user-item interactions. Denote X as the user-item interaction matrix, where X_{ij} indicates number of times user i has listened to item j. The goal is to learn a d-dimensional user embedding matrix U and item embedding matrix V by solving the problem:

$$\min_{U \in \mathbb{R}^{n_{\text{users}} \times d}, V \in \mathbb{R}^{n_{\text{items}} \times d}} - \sum_{i,j} X_{ij} \text{logsoftmax}_i\Big(\langle u_i, v_j \rangle\Big)$$

where $\text{logsoftmax}_i(\langle u_i, v_j \rangle) := \log\big(\exp(\langle u_i, v_j \rangle) / \sum_k \exp(\langle u_i, v_k \rangle)\big)$. In order to control computation cost, we employ random negative sampling to approximate the partition function $\sum_k \exp(\langle u_i, v_k \rangle)$. We then pass the embedding dot product $\langle u_i, v_j \rangle$ as one input feature to the customer relevance model, where u_i, v_j are learned embeddings for the user and candidate item, respectively.

We choose not to include any context features such as time-of-day in the model, this allows customer-relevance scores to be pre-computed and stored in a low-latency database for later access.

3.3. Feature selection

Feature set $\mathcal{F}_p$ is partitioned into multiple subsets $\{\mathcal{F}_p^i\}$, where each subset $\mathcal{F}_p^i$ represents a particular facet, such as customer-item historical playback, customer-artist historical playback, customer-item follows and likes and embedding dot-product. These distinct sets of features are created using separate batch-compute pipelines, which extract and process data from different data warehouses. The resulting features are then uploaded to a low-latency database to allow low latency access for real-time model inference. It's desirable to reduce the number of feature subsets used in production in order to limit engineering costs, as long as it maintains model performance. For this reason, we focus our efforts on selecting the best collection of feature subsets.

To this end, we perform a backward feature elimination on these feature subsets. During each stage, we first perform HPOs on variants models trained with different feature sets $\mathcal{S} - \mathcal{F}_p^i,\ \mathcal{F}_p^i \in \mathcal{S}$, separately, where $\mathcal{S}$ is the subsets of features determined by the previous step (at stage 1, $\mathcal{S} = \bigcup_i \mathcal{F}_p^i$). Then we train each model N times with random initialization, using the best-performing hyper-parameter. A feature subset is eliminated only if the average performance of the model without the feature subset is superior. Our findings revealed that all feature subsets, barring the entity-type indicator features, made a positive and statistically significant contribution to the model performance. Consequently, we integrated all feature subsets, except for the entity-type indicator features, into the model.

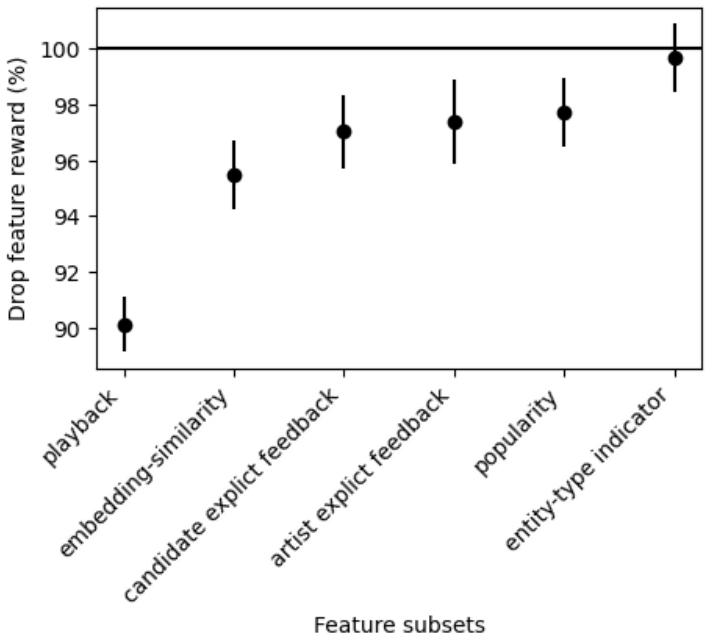

Figure 1: Ablation model performance. The horizontal line indicates the baseline reward of model trained on $\mathcal{F}_p$ and each dot indicates the average reward, in percentage, of models trained on $\mathcal{F}_p$ minus the labeled feature subset. Confidence intervals are computed using Welch's t-test with $p = 0.05$.

Notably, the embedding dot product from the user-item embedding model is often used in lieu of our proposed customer-relevance model to re-rank candidates. Nevertheless, our ablation study demonstrates that augmenting the embedding dot product with additional feature set leads to significantly improved ranking performance. Specifically, the playback feature, which includes user-item play counts within different time windows and negative feedback such as skips and stops, notably contributed to the model's performance.

3.4. Online data collection and offline evaluation methods

Offline experimentation enables us to evaluate performance and improve model architecture such as training loss function, features and hyper-parameters, with no direct impact to customer experience.

To this regard, randomized content ranking data are collected from production systems, to be used for offline experiments. The online ranker first ranks candidates, then apply a Fisher–Yates random shuffle to ranked candidates before outputting the first k candidates to be shown to the customer. The random shuffle step is critical to ensure that every candidate has a non-zero probability of being ranked at any position, thus allowing an unbiased off-policy evaluation through importance-sampling.

This logged data is then used for offline experiments. Firstly, the data is split into train/validation/test sets where we ensure that 1) validation and test sets come from the week after training period ends 2) test and validation sets are based on disjoint sets of customers, to prevent data leakage. Then we evaluate each offline-trained policy using the item-position model (IPM) [20]. Under the assumption that the likelihood of a customer interacting with the item only depends the item itself and the ranked position, and independent from other items being ranked, the IPM estimator is unbiased.

4. Experiments

In this section we describe three A/B tests we ran on production traffic. To evaluate treatment impacts over control we adopt a Bayesian framework [21, 22, 23, 24] which is broadly adopted in industry due to its advantages over frequentist methods [25]. In this paper's Bayesian analysis, we treat the result as statistically significant when the probability of positive return (PPR) exceeds 66% (i.e., 2:1 odds of positive return).

First we ran the model consolidation experiment, followed by the feature summarization experiment, and then evaluated the model on the top-1 selection task. During the A/B tests, we focused on the widgets on the home page of a music streaming service, particularly on the problem of personalized re-ranking of the items in each of the widgets. This includes item-item focused widgets (e.g., *More Like <Artist>, Because You Followed <Artist>)*, customer-item widgets for various content types (e.g., *<Songs> For You, <Stations> For You*), recent-preference widgets (e.g., *You Might Like*), as well as Top-1 recommender widget.

We ran experiments in this order because we expected that consolidating the re-ranking model would simplify the deployment and evaluation of the customer-item relevance score.

For each experiment, we run a 2-week A/B test on production traffic, randomly splitting users into different treatment groups. Since model consolidation and feature summarization are general design choices that apply to all widgets on the page, we report an engagement metric that measures the success rate of sessions on the home page[1], and we do not report widget-specific metrics.

4.1. Consolidated re-ranking model

In this experiment we used a consolidated model to rank all the content of widgets on the home page where each widget has a different feature space that takes into account the widget specific ranking strategy based on the widget's objective which can range from showing popular content or to increase discoveries. Note that in this setting, the team owning the product has to research and specify which features are included in the model.

We run a 2 weeks A/B test with the following treatments:

- **C** – current production system where each widget is ranked by an individual specialized model;
- **T1** – consolidated model based on DeepProp [3] shared across all widgets;
- **T2** – consolidated model based on [18] shared across all widgets.

We report results in Table 1. We registered a 0.44% improvement in session success rate on the home page when using the consolidated DeepProp model compared to the control setting with specialized re-rankers for each widget. Interestingly, we see that a consolidated model based on reward regression and point-wise cost function (T2) not only under-performs DeepProp, but also the control setting. We

[1]A sessions is considered successful if it results in a high-value-action from the user, including playback over a minimum threshold of time, a like on an entity, following an artist, and similar.

hypothesize this is due to the shared reward regression model failing to capture the different reward distributions associated to the different widgets (e.g., widget driving more/less traffic and engagement). On the contrary, DeepProp does not have this problem, since it learns to predict the relative ranking of items, and not their actual relevance [26].

	Engagement Metric	PPR
T1	0.44%	100%
T2	-0.32%	0.0%

Table 1
A/B test results. Engagement metric reports the relative lift over the baseline in C.

4.2. Feature summarization via customer-item score

In this experiment we trained a reward regression model as presented in [18] to regress clicks on widgets' content in the home page. The model is trained offline using logged data from production traffic. The idea is to predict whether an item inside the widget will be clicked by the user. This way, a product team can simply set its model (e.g., a consolidated DeepProp re-ranker as in Section 4.1) to consume this score as replacement of many personalization features.

To validate the idea of features summarization, we designed an experiment in which we replaced a subset $\mathcal{F}_p \subset \mathcal{F}$ of personalization features where $|\mathcal{F}_p| = 45$ with a customer-item affinity score z. We run a one-week A/B test with these treatments:

- **C** baseline that uses only features from $\mathcal{F}_0 = \mathcal{F} - \mathcal{F}_p$;
- **T1** uses all the features in $\mathcal{F}$ directly as input to the model;
- **T2** replaces $\mathcal{F}_p$ with the affinity score z, thus the features space is $\mathcal{F}_0 \cup z$.

All the treatments use a the consolidated DeepProp re-ranker from Section 4.1 (one per treatment). The difference is that T1 uses the full feature set $\mathcal{F}$ as input, T2 replaces $\mathcal{F}_p$ with a single customer-item score z, while C drops them altogether.

	Engagement Metric	PPR
T1	2.23%	100%
T2	2.12%	100%

Table 2
A/B test results. Engagement metric reports the relative lift over the baseline in C. T2 summarizes $\mathcal{F}_p$ from T1 and it retains a lift of 95% from T1.

We report results in Table 2. First, by comparing T1 to C, we see that more personalization features (i.e., adding $\mathcal{F}_p$ to $\mathcal{F}_0$) help, leading to a relative lift of 2.23% in sessions success rate. By comparing T2 to C, we also see that using the single customer-item score leads to a substantial improvement over the baseline of 2.12%. Finally, by comparing these two lifts, we see that including the single customer-item score retains 95% of the benefit of including the original features directly. In other words, using z instead of $\mathcal{F}_p$ as input features for a ranking model would consist in 95% of the original lift given by $\mathcal{F}_p$.

4.3. Content selection in Top-1 Recommender

Top-1 Recommender is a widget fixed on top position on our home page. In this widget a single content is selected from a pool of candidates and shown to the customer. Therefore, optimizing this widget is treated as selection problem instead of a ranking problem like the rest of the home page. To evaluate impact of a A/B test on this widget we use listening time instead of engagement metric. There are multiple reasons for this choice, which include the particular customer experience design of this widget.

In this experiment we leveraged the feature summarization model A/B tested in 4.2 in Top-1 Recommender on the home page. Thus, in this A/B test we evaluate the feature summarization model for content selection use-case instead of ranking.

Following the experiment design described in Section 4.2, we run a two-week A/B test with the following treatments all are powered by a linear model:

- **C** baseline that uses only features from $\mathcal{F}_0 = \mathcal{F} - \mathcal{F}_p$;
- **T1** uses all the features in $\mathcal{F}$ directly as input to the model;
- **T2** replaces $\mathcal{F}_p$ with the affinity score z, thus the features space is $\mathcal{F}_0 \cup z$.

	Listening time	PPR
T1	0.01%	53.23%
T2	0.09%	88.87%

Table 3
A/B test results. Listening time reports the relative lift over the baseline in C. T2 outperform T1 in terms of relative impact with a probability of positive return greater than 66% which we consider statistically significant.

Interestingly, results in Table 3 show that T2 using z as replacement of $\mathcal{F}_p$ from T1, outperforms the latter. This result is surprising compared to the A/B test results described in Section 4.2. We argue that in this case since T1 is powered by a linear model is unable to learn high-order feature interactions using all the features $\mathcal{F}$ directly while T2 has the advantage that z is computed by a non-linear model which indeed is able to learn high-order feature interactions in $\mathcal{F}_p$.

5. Conclusions

In this paper we presented an approach that can drastically simplify how product teams bring personalization into their experiences. We proposed a customer-item relevance score that can summarize many personalization features and is trained on feedback from different widgets. Product teams can feed this score into their recommendation models in place of directly integrating with all the personalization features. In particular, we saw that on an application on the home page of a music streaming service that integrating with the customer-items score retains most of the benefits of integrating with personalization feature directly. We argue this is a benefit for product teams, since it gives the option to simplify building personalization into a product and focus resources on challenges peculiar to the product (e.g., sequencing for automated radio stations). We showed that using a consolidated model for re-ranking the content in all widgets on the home page is beneficial in terms of engagement. We also experimented in a content selection use-case reporting positive impact on listening time for the top-1 recommender. Whereas in this work we tested the idea on the home page of our streaming service, in the future we plan to expand experimentation and deployment of the customer-item relevance score in more parts of our service that need personalization.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 89–96.

[2] Z. Hu, Y. Wang, Q. Peng, H. Li, Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm, in: The World Wide Web Conference, 2019, pp. 2830–2836.

[3] A. Agarwal, K. Takatsu, I. Zaitsev, T. Joachims, A general framework for counterfactual learning-to-rank, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 5–14.
[4] N. Craswell, O. Zoeter, M. Taylor, B. Ramsey, An experimental comparison of click position-bias models, in: Proceedings of the 2008 international conference on web search and data mining, 2008, pp. 87–94.
[5] Z. A. Zhu, W. Chen, T. Minka, C. Zhu, Z. Chen, A novel click model and its applications to online advertising, in: Proceedings of the third ACM international conference on Web search and data mining, 2010, pp. 321–330.
[6] T. Joachims, L. Granka, B. Pan, H. Hembrooke, G. Gay, Accurately interpreting clickthrough data as implicit feedback, in: Acm Sigir Forum, volume 51, Acm New York, NY, USA, 2017, pp. 4–11.
[7] O. Jeunen, Counterfactual inference under thompson sampling, arXiv preprint arXiv:2504.08773 (2025).
[8] D. Agarwal, M. Gurevich, Fast top-k retrieval for model based recommendation, in: Proceedings of the fifth ACM international conference on Web search and data mining, 2012, pp. 483–492.
[9] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al., The youtube video recommendation system, in: Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 293–296.
[10] Y. Wang, X. Zhao, T. Xu, X. Wu, Autofield: Automating feature selection in deep recommender systems, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1977–1986.
[11] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, Y. Yu, Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction, in: proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 2636–2645.
[12] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, J. Tang, Autoint: Automatic feature interaction learning via self-attentive neural networks, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1161–1170.
[13] Y. Wang, H. Guo, B. Chen, W. Liu, Z. Liu, Q. Zhang, Z. He, H. Zheng, W. Yao, M. Zhang, et al., Causalint: Causal inspired intervention for multi-scenario recommendation, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 4090–4099.
[14] Y. Wang, X. Zhao, B. Chen, Q. Liu, H. Guo, H. Liu, Y. Wang, R. Zhang, R. Tang, Plate: A prompt-enhanced paradigm for multi-scenario recommendations, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 1498–1507.
[15] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, E. H. Chi, Modeling task relationships in multi-task learning with multi-gate mixture-of-experts, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 1930–1939.
[16] A. Royer, T. Blankevoort, B. Ehteshami Bejnordi, Scalarization for multi-task and multi-domain learning at scale, Advances in Neural Information Processing Systems 36 (2023) 16917–16941.
[17] J. Hwang, H. Ju, S. Kang, S. Jang, H. Yu, Multi-domain sequential recommendation via domain space learning, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 2134–2144.
[18] M. Morik, A. Singh, J. Hong, T. Joachims, Controlling fairness and bias in dynamic learning-to-rank, in: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020, pp. 429–438.
[19] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, K. Nakata, Unbiased recommender learning from missing-not-at-random implicit feedback, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 501–509.
[20] S. Li, Y. Abbasi-Yadkori, B. Kveton, S. Muthukrishnan, V. Vinay, Z. Wen, Offline evaluation of ranking policies with click models, 2018. arXiv:1804.10488.
[21] S. Chennu, A. Maher, C. Pangerl, S. Prabanantham, J. H. Bae, J. Martin, B. Goswami, Rapid and scalable bayesian ab testing, in: 2023 IEEE 10th International Conference on Data Science and

Advanced Analytics (DSAA), IEEE, 2023, pp. 1–10.
[22] S. Kamalbasha, M. J. Eugster, Bayesian a/b testing for business decisions, in: International Data Science Conference, Springer, 2020, pp. 50–57.
[23] Y. Zhang, C. Qian, W. Bao, Bayesian a/b testing with covariates, in: 2023 IEEE International Conference on Data Mining (ICDM), IEEE, 2023, pp. 1553–1558.
[24] A. Deng, Objective bayesian two sample hypothesis testing for online controlled experiments, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 923–928.
[25] Q. F. Gronau, A. R. KN, E.-J. Wagenmakers, Informed bayesian inference for the a/b test, Journal of Statistical Software 100 (2021) 1–39.
[26] H. Li, Learning to rank for information retrieval and natural language processing, Springer Nature, 2022.