Recommending What Video to Watch Next: A Multitask Ranking System

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi
Google, Inc.
{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

ABSTRACT

In this paper, we introduce a large scale multi-objective ranking system for recommending what video to watch next on an industrial video sharing platform. The system faces many real-world challenges, including the presence of multiple competing ranking objectives, as well as implicit selection biases in user feedback. To tackle these challenges, we explored a variety of soft-parameter sharing techniques such as Multi-gate Mixture-of-Experts so as to efficiently optimize for multiple ranking objectives. Additionally, we mitigated the selection biases by adopting a Wide & Deep framework. We demonstrated that our proposed techniques can lead to substantial improvements on recommendation quality on one of the world's largest video sharing platforms.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Recommender systems**; • **Computing methodologies** → **Ranking**; **Multi-task learning**; **Learning from implicit feedback**.

KEYWORDS

Recommendation and Ranking, Multitask Learning, Selection Bias

ACM Reference Format:
Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi. 2019. Recommending What Video to Watch Next: A Multitask Ranking System. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3298689.3346997

1 INTRODUCTION

In this paper, we describe a large-scale ranking system for video recommendation. That is, given a video which a user is currently watching, recommend the next video that the user might watch and enjoy. Typical recommendation systems follow a two-stage design with a candidate generation and a ranking [10, 20]. This paper focuses on the ranking stage. In this stage, the recommender has a few hundred candidates retrieved from the candidate generation (e.g. matrix factorization [45] or neural models [25]), and applies a sophisticated large-capacity model to rank and sort the most promising items. We present experiments and lessons learned from building such a ranking system on a large-scale industrial video publishing and sharing platform.

Designing and developing a real-world large-scale video recommendation system is full of challenges, including:

- There are often different and sometimes conflicting objectives which we want to optimize for. For example, we may want to recommend videos that users rate highly and share with their friends, in addition to watching.
- There is often implicit bias in the system. For example, a user might have clicked and watched a video simply because it was being ranked high, not because it was the one that the user liked the most. Therefore, models trained using data generated from the current system will be biased, causing a feedback loop effect [33]. How to effectively and efficiently learn to reduce such biases is an open question.

To address these challenges, we propose an efficient multitask neural network architecture for the ranking system, as shown in Figure 1. It extends the Wide & Deep [9] model architecture by adopting Multi-gate Mixture-of-Experts (MMoE) [30] for multitask learning. In addition, it introduces a shallow tower to model and remove selection bias. We apply the architecture to video recommendation as a case study: given what user currently is watching, recommend the next video to watch. We present experiments of our proposed ranking system on an industrial large-scale video publishing and sharing platform. Experimental results show significant improvements of our proposed system.

Specifically, we first group our multiple objectives into two categories: 1) engagement objectives, such as user clicks, and degree of engagement with recommended videos; 2) satisfaction objectives, such as user liking a video on YouTube, and leaving a rating on the recommendation. To learn and estimate multiple types of user behaviors, we use MMoE to automatically learn parameters to share across potentially conflicting objectives. The Mixture-of-Experts [21] architecture modularizes input layer into experts, each of which focuses on different aspects of input. This improves the representation learned from complicated feature space generated from multiple modalities. Then by utilizing multiple gating networks, each of the objectives can choose experts to share or not share with others.

To model and reduce the selection bias (e.g., position bias) from biased training data, we propose to add a shallow tower to the main model, as shown in the left side of Figure 1. The shallow tower takes input related to the selection bias, e.g., ranking order decided by the current system, and outputs a scalar serving as a bias term to the final prediction of the main model. This model

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09.
https://doi.org/10.1145/3298689.3346997

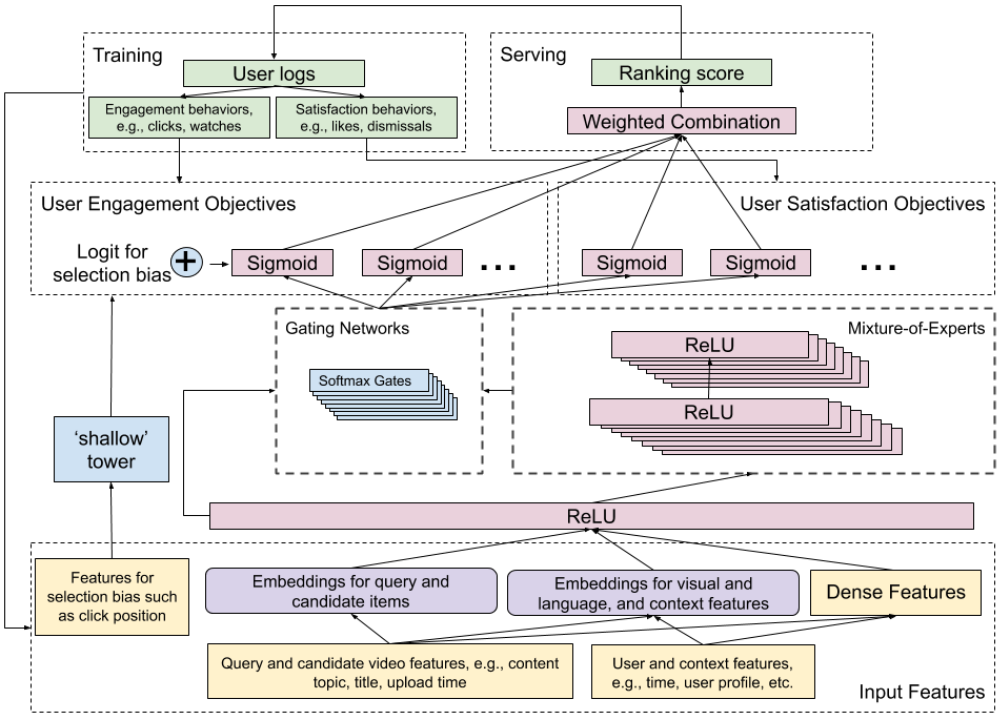

Figure 1: Model architecture of our proposed ranking system. It consumes user logs as training data, builds Multi-gate Mixture-of-Experts layers to predict two categories of user behaviors, i.e., engagement and satisfaction. It corrects ranking selection bias with a side-tower. On top, multiple predictions are combined into a final ranking score.

architecture factorizes the label in training data into two parts: the unbiased user utility learned from the main model, and the estimated propensity score learned from the shallow tower. Our proposed model architecture can be treated as an extension of the Wide & Deep model, with the shallow tower representing the Wide part. By directly learning the shallow tower together with the main model, we have the benefit of learning the selection bias without resorting to random experiments to get the propensity score [41].

To evaluate our proposed ranking system, we design and conduct offline and live experiments to verify the effectiveness of: 1) multitask learning, and 2) removing a common type of selection bias, namely, position bias. Comparing with state-of-the-art baseline methods, we show significant improvements of our proposed framework. We use YouTube, one of the largest video sharing platforms, to conduct our experiments.

In summary, our contributions are as follows:

- We introduce an end-to-end ranking system for video recommendations.
- We formulate the ranking problem as a multi-objective learning problem and extend the Multi-gate Mixture-of-Experts architecture to improve performance on all objectives.
- We propose to apply a Wide & Deep model architecture to model and mitigate position bias.
- We evaluate our approach on a real-world large-scale video recommendation system and demonstrate significant improvements.

The rest of this paper is organized as follows: In Section 2, we describe related work in building real-world recommendation ranking systems. In Section 3, we provide problem descriptions for both the candidate generation and ranking. Next, we talk about our proposed approach in two aspects, multitask learning and removing selection bias. In Section 5, we describe how we design offline and live experiments to evaluate our proposed framework. Finally, we conclude with our findings in Section 6.

2 RELATED WORK

The problem of recommendation can be formulated as returning a number of high-utility items given a query, a context, and a list of items. For example, a personalized movie recommendation system can take a user's watch history as a query, a context such as Friday night on a tablet at home, a list of movies, and return a subset of movies that this user is likely to watch and enjoy. In this section, we discuss related work under three categories: industrial case studies on recommendation systems, multi-objective recommendation systems, and understanding biases in training data.

2.1 Industrial Recommendation Systems

To design and develop a successful ranking system empowered by machine-learned models, we need large quantities of training data. Most recent industrial recommendation systems rely heavily on large amount of user logs for building their models. One option is to directly ask users for their explicit feedback on item utility. However, due to its cost, the quantity of explicit feedback can hardly scale

up. Therefore, ranking systems commonly utilize implicit feedback such as clicks and engagement with the recommended items.

Most recommendation systems [10, 20, 42] contain two stages: candidate generation and ranking. For candidate generation, multiple sources of signals and models are used. For example, [26] used co-occurrences of items to generate candidates, [11] adopted a collaborative filtering based method, [14] and [19] applied a random walk on (co-occurrence) graph, [42] learned content representation to filter items to candidates, and [10] described a hybrid approach using mixture of features.

For ranking, machine learning algorithms using a learning-to-rank framework are widely adopted. For example, [26] explored both point-wise and pair-wise learning to rank framework with linear models and tree based methods. [16] used a linear scoring function and a pair-wise ranking objective. [20] applied Gradient Boosted Decision Tree (GBDT [24]) for a point-wise ranking objective. [10] employed a neural network with a point-wise ranking objective to predict a weighted click.

One main challenge of these industrial recommendation systems is scalability. Therefore, they commonly adopt a combination of infrastructure improvements [11, 14, 19, 26] and efficient machine learning algorithms [14, 16, 17, 42]. To make a tradeoff between model quality and efficiency, a popular choice is to use deep neural network-based point-wise ranking models [10].

In this paper, we first identify a critical issue in industrial ranking systems: the misalignment between user implicit feedback and true user utility on recommended items. Subsequently, we introduce a deep neural network-based ranking model which uses multitask learning techniques to support multiple ranking objectives, each of which corresponds to one type of user feedback.

2.2 Multi-objective Learning for Recommendation Systems

Learning and predicting user behaviors from training data is challenging. There are different types of user behaviors, such as clicking [22], rating, and commenting etc. However, each one does not independently reflect true user utility. For example, a user can click an item but end up not liking it; users can only provide ratings to clicked and engaged items. Our ranking system needs to be able to learn and estimate multiple types of user behaviors and utilities effectively, and subsequently combines these estimations to compute a final utility score for ranking.

Existing works on behavior aware and multi-objective recommendation either can only be applied at candidate generation stage [3, 28, 31, 40, 45], or are not suitable for large-scale online ranking [13, 15, 38, 44].

For example, some recommendation systems [31, 45] extend collaborative filtering or content based systems to learn user-item similarity from multiple user signals. These systems are efficiently used to generate candidates. But compared to ranking models based on deep neural network, they are not as effective in providing the final recommendations [10].

On the other hand, many existing multi-objective ranking systems are designed for specific types of features and applications, such as text [38] and vision [13]. It would be challenging to extend these systems to support feature spaces from multiple modalities, e.g., text from video titles, and visual feature from thumbnails. Meanwhile, other multi-objective ranking systems that consider multiple modalities of input features cannot scale up, due their limitation in efficiently sharing model parameters for multiple objectives [15, 44].

Outside the recommendation system research area, deep neural network based multitask learning has been widely studied and explored on many traditional machine learning applications for representation learning, e.g., natural language processing [12], and computer vision [27]. While many multitask learning techniques proposed for representation learning are not practical for constructing ranking systems, some of their building blocks inspire our design. In this paper, we describe a DNN based ranking system designed for real-world recommendations and apply an extension of Mixture-of-Experts layers [21] to support multitask learning [30].

2.3 Understanding and Modeling Biases in Training Data

User logs, which are used as our training data, capture user behaviors and responses to recommendations from the current production system. The interactions between users and the current system create selection biases in the feedback. For example, a user may have clicked an item because it was selected by the current system, even though it was not the most useful one of the entire corpus. Therefore, new models trained on data generated from the current system will be biased towards the current system, causing a feedback loop effect. How to effectively and efficiently learn to reduce such biases for ranking systems is an open question.

Joachims et al. [22] first analyzed position bias and presentation bias in implicit feedback data for training learning to rank models. By comparing click data with explicit feedback of relevance, they found that position bias exists in click data and can significantly affect learning to rank models in estimating relevance between query and document. Following this finding, many approaches have been proposed to remove such selection biases, especially position bias [23, 34, 41].

A commonly used practice is to inject position as an input feature in model training and then removing the bias through ablation at serving. In probabilistic click models, position is used to learn $P(relevance|pos)$. One method to remove position bias is inspired by [8], where Chapelle et al. evaluated a CTR model using $P(relevance|pos = 1)$, under the assumption of no position bias effect for evaluation at position 1. Subsequently, to remove position bias, we can train a model using position as an input feature, and serve by setting position feature to 1 (or other fixed value such as missing value).

Other approaches try to learn a bias term from position and apply it as a normalizer or regularizer [23, 34, 41]. Usually, to learn a bias term, some random data needs to be used to infer the bias term (referred to as 'global bias', 'propensity', etc.) without considering relevance [34, 41]. In [23], inverse propensity score (IPS) is learned using a counter-factual model where no random data is needed. It is used as a regularization term in training a Rank-SVM.

In real-world recommendation systems, especially social media platforms such as Twitter [19] and YouTube [10], user behaviors and item popularities can change significantly every day. Therefore, instead of IPS based approaches, we need to have an efficient way

to adapt to training data distribution change in modeling selection biases while we are training the main ranking model.

3 PROBLEM DESCRIPTION

In this section, we first describe our problem of recommending video to watch next, then we introduce the two-stage setup of candidate generation and ranking. The rest of the paper will focus on the ranking system.

Besides the above-mentioned challenges for building ranking systems trained with implicit feedback, for real-world large-scale video recommendation problems, we need to consider the following additional factors:

- *Multimodal feature space.* In a context-aware personalized recommendation system, we need to learn user utility of candidate videos with feature space generated from multiple modalities, e.g., video content, thumbnail, audio, title and description, user demographics. Learning representation from multimodal feature space for recommendation is uniquely challenging compared to other machine learning applications. It cuts across two difficult issues: 1) bridging the semantic gap from low-level content features for content filtering; 2) learning from sparse distribution of items for collaborative filtering.
- *Scalability.* Scalability is extremely important since we are building a recommendation system for billions of users and videos. The model must be effective at training and efficient at serving. Even though ranking system scores only hundreds of candidates per query, real-world scenarios require scoring to be done in real-time, because some query and context information are only available online. Therefore, ranking system needs to not only learn representations of billions of items and users, but also be efficient during serving.

Recall that the goal of our recommendation system is to provide a ranked list of videos, given currently watching video and context. To deal with multimodal feature spaces, for each video, we extract features such as video meta-data and video content signals as its representation. For context, we use features such as user demographics, device, time, and location, etc.

To deal with scalability, similar to what was described in [10], our recommendation system has two stages, namely, candidate generation and ranking. At the candidate generation stage, we retrieve a few hundred candidates from a a huge corpus. Our ranking system provides a score for each candidate and generates the final ranked list.

3.1 Candidate Generation

Our video recommendation system uses multiple candidate generation algorithms, each of which captures one aspect of similarity between query video and candidate video. For example, one algorithm generates candidates by matching topics of query video. Another algorithm retrieves candidate videos based on how often the video has been watched together with the query video. We construct a sequence model similar to [10] for generating personalized candidate given user history. We also use techniques mentioned in [25] to generate context-aware high recall relevant candidates. At the end, all candidates are pooled into a set and subsequently scored by the ranking system.

3.2 Ranking

Our ranking system generates a ranked list from a few hundred candidates. Different from candidate generation, which tries to filter the majority of items and only keep relevant ones, ranking system aims to provide a ranked list so that items with highest utility to users will be shown at the top. Therefore, we apply most advanced machine learning techniques using a neural network architecture in ranking system, in order to have sufficient model expressiveness for learning association of features and their relationship with utility.

4 MODEL ARCHITECTURE

In this section, we describe our proposed ranking system in detail. We first provide an overview of the system, including its problem formulation, objectives, and features. Then we discuss our multi-objective setup for learning multiple types of user behaviors. We talk about how we apply and extend a state-of-the-art multitask learning model architecture called Multi-gate Mixture-of-Experts (MMoE) for learning multiple ranking objectives. At last, we talk about how we combine MMoE with a shallow tower to learn and reduce selection bias, especially position bias in the training data.

4.1 System Overview

Our ranking system learns from two types of user feedback: 1) engagement behaviors, such as clicks and watches; 2) satisfaction behaviors, such as likes and dismissals. Given each candidate, the ranking system uses features of the candidate, query and context as input, and learns to predict multiple user behaviors.

For problem formulation, we adopt the learning-to-rank framework [6]. We model our ranking problem as a combination of classification problems and regression problems with multiple objectives. Given a query, candidate, and context, the ranking model predicts the probabilities of user taking actions such as clicks, watches, likes, and dismissals.

This approach of making predictions for each candidate is a point-wise approach [6]. In contrast, pair-wise or list-wise approaches learn to make predictions on ordering of two or multiple candidates. Pair-wise or list-wise approaches can be used to potentially improve the diversity of the recommendations. However, we opt to use point-wise ranking mainly based on serving considerations. At serving time, point-wise ranking is simple and efficient to scale to a large number of candidates. In comparison, pair-wise or list-wise approaches need to score pairs or lists multiple times in order to find the optimal ranked list given a set of candidates, thereby limiting their scalability.

4.2 Ranking Objectives

We use user behaviors as training labels. Since users can have different types of behaviors towards recommended items, we design our ranking system to support multiple objectives. Each objective is to predict one type of user behavior related to user utility. For description purposes, in the following we separate our objectives into two categories: engagement objectives and satisfaction objectives.

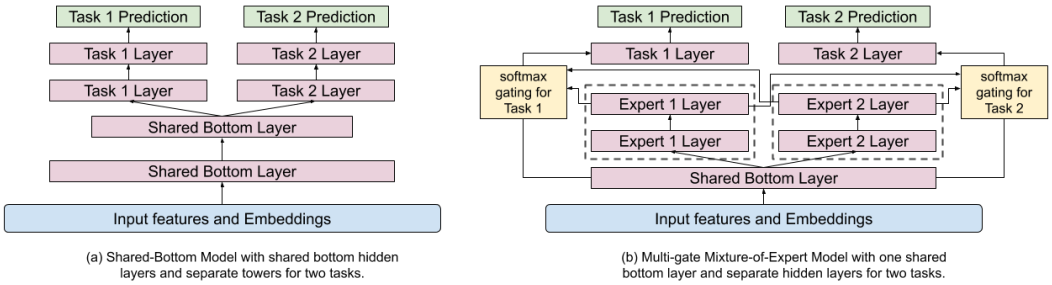

Figure 2: Replacing shared-bottom layers with MMoE.

Engagement objectives capture user behaviors such as clicks and watches. We formulate the prediction of these behaviors into two types of tasks: binary classification task for behaviors such as clicks, and regression task for behaviors related to time spent. Similarly, for satisfaction objectives, we formulate the prediction of behaviors related to user satisfactions into either binary classification task or regression task. For example, behavior such as clicking like for a video is formulated as a binary classification task, and behavior such as rating is formulated as regression task. For binary classification tasks, we compute cross entropy loss. And for regression tasks, we compute squared loss.

Once multiple ranking objectives and their problem types are decided, we train a multitask ranking model for these prediction tasks. For each candidate, we take the input of these multiple predictions, and output a combined score using a combination function in the form of weighted multiplication. The weights are manually tuned to achieve best performance on both user engagements and user satisfactions.

4.3 Modeling Task Relations and Conflicts with Multi-gate Mixture-of-Experts

Ranking systems with multiple objectives commonly use a shared-bottom model architecture [7, 10]. However, such hard-parameter sharing techniques sometimes harm the learning of multiple objectives when correlation between tasks is low [30]. To mitigate the conflicts of multiple objectives, we adopt and extend a recently published model architecture, Multi-gate Mixture-of-Experts (MMoE) [30].

MMoE is a soft-parameter sharing model structure designed to model task conflicts and relations. It adapts the Mixture-of-Experts (MoE) structure to multitask learning by having the experts shared across all tasks, while also having a gating network trained for each task. The MMoE layer is designed to capture the the task differences without requiring significantly more model parameters compared to the shared-bottom model. The key idea is to substitute the shared ReLu layer with the MoE layer and add a separate gating network for each task.

For our ranking system, we propose to add experts on top of a shared hidden layer, as shown in Figure 2b. This is because a Mixture-of-Experts layer can help to learn modularized information from its input [21]. It can better model multimodal feature space when being used directly on top of input layer or lower hidden layers. However, directly applying MoE layer on input layer will significantly increase model training and serving cost. This is because usually the dimensionality of input layer is much higher than those of hidden layers.

Our implementation of the expert networks is identical to multi-layer perceptrons with ReLU activations [30]. Given task k, the prediction y_k, and the last hidden layer h^k, the MMoE layer with n experts output for task k: $f^k(x)$, can be expressed in the following equation:

$$y_k = h^k(f^k(x)),$$
$$\text{where } f^k(x) = \sum_{i=1}^{n} g^k_{(i)}(x) f_i(x) \tag{1}$$

And $x \in \mathbb{R}^d$ is a lower-level shared hidden embedding, g^k is the gating network for task k, $g^k(x) \in \mathbb{R}^n$, $g^k_{(i)}(x)$ is the ith entry, and $f_i(x)$ is the ith expert. The gating networks are simply linear transformations of the input with a softmax layer.

$$g^k(x) = \text{softmax}(W_{g^k} x), \tag{2}$$

where $W_{g^k} \in \mathbb{R}^{n \times d}$ are free parameters for the linear transformation. In contrast to the sparse gating network mentioned in [32], where the number of experts can be large and each of the training examples only utilizes the top experts, we use a relatively small number of experts. This is set up to encourage sharing of experts by multiple gating networks and for training efficiency.

4.4 Modeling and Removing Position and Selection Biases

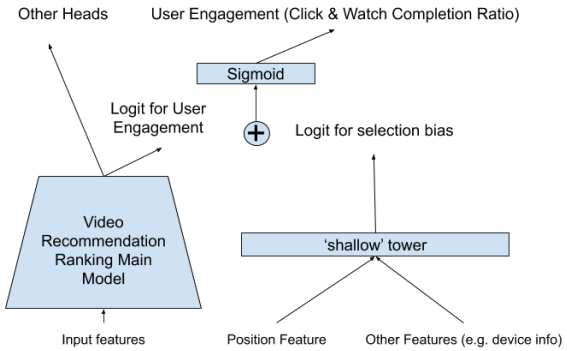

Figure 3: Adding a shallow side tower to learn selection bias (e.g., position bias).

Implicit feedback has been widely used to train learning to rank models. With a large amount of implicit feedback extracted from user logs, complicated deep neural network based model can be trained. However, implicit feedback is biased due to the fact that it is generated from the existing ranking system. Position bias, and many other types of selection biases, are studied and verified for existence in different ranking problems [2, 23, 41].

In our ranking system, where the query is a video currently being watched and candidates are relevant videos, it is common that users are inclined to clicking and watching videos displayed closer to the top of the list, regardless of their actual user utility - both in terms of relevance to the watched video as well as the users' preferences. Our goal is to remove such a position bias from the ranking model. Modeling and reducing selection biases in our training data or during model training can result in model quality gain and break the feedback loop resulted from the selection biases.

Our proposed model architecture is similar to the Wide & Deep model architecture. We factorize the model prediction into two components: a user-utility component from the main tower, and a bias component from the shallow tower. Specifically, we train a shallow tower with features contributing to selection bias, such as position feature for poistion bias, then add it to the final logit of the main model, as shown in Figure 3. In training, the positions of all impressions are used, with a 10% feature drop-out rate to prevent our model from over-relying on the position feature. At serving time, position feature is treated as missing. The reason why we cross position feature with device feature is that different position biases are observed on different types of devices.

5 EXPERIMENT RESULTS

In this section, we describe how we conduct experiments of our proposed ranking system to recommend what video to watch next on one of the largest video sharing platforms, YouTube. Using user implicit feedback provided by YouTube, we train our ranking models, and conduct both offline and live experiments.

The scale and complexity of YouTube makes it a perfect test-bed for our ranking system. YouTube is the largest video sharing platform, with 1.9 billion monthly active users [1]. The website creates hundreds of billions of user logs everyday in the form of user activities interacting with recommended results. A key product of YouTube provides the functionality of recommending what to watch next given a watched video, as shown in Figure 4. Its user interface provides multiple ways for users to interact with recommended videos, such as clicks, watches, likes, and dismissals.

5.1 Experiment Setup

As described in Section 3.1, our ranking system takes a few hundred candidates from multiple candidate generation algorithms. We use TensorFlow [2] to build the training and serving of the model. Specifically, we use Tensor Processing Units (TPUs) to train our model and serve it using TFX Servo [4] [3].

[1]https://www.youtube.com/yt/about/press
[2]https://www.tensorflow.org
[3]https://www.tensorflow.org/tfx/guide/serving

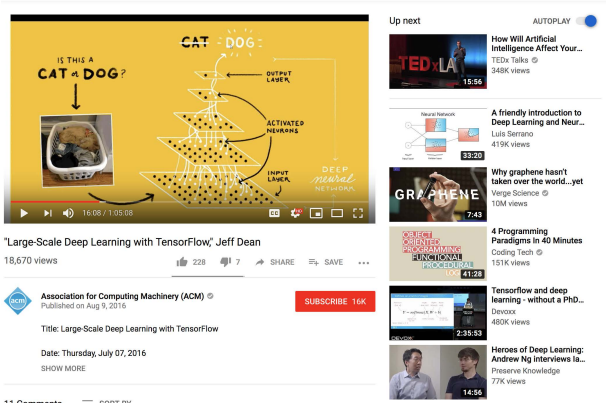

Figure 4: Recommending what to watch next on YouTube.

We train both our proposed model and baseline models sequentially. This means that we train our models by going through training data of past days following a temporal order and keep running our trainer to consume newly arriving training data. By doing so, our models adapt to the most recent data. This is critical for many real-world recommendation applications, where data distribution and user patterns change dynamically over time.

For offline experiments, we monitor AUC for classification task and squared error for regression tasks. For live experiments, we conduct A/B testing comparing with production system. We use both offline and live metrics to tune hyper-parameters such as learning rate. We examine multiple engagement metrics such as time spent at YouTube, and satisfaction metrics such as rate of dismissals, user survey responses, etc. In addition to live metrics, we also care about the computation cost of the model at serving time, since YouTube responds a substantially large number of queries per second.

5.2 Multitask Ranking With MMoE

To evaluate the performance of adopting MMoE for multitask ranking, we compare with baseline methods and conduct live experiments on YouTube.

5.2.1 Baseline Methods. Our baseline methods use the shared-bottom model architecture mentioned in Figure 2a. As a proxy, we measure model complexity by the number of multiplications inside each model architecture, because this is the main computation cost for serving the model. When comparing a MMoE model and a baseline model, we use the same model complexity. Due to efficiency concerns, our MMoE layer shares one bottom hidden layer (as shown in Figure 2b), which uses a lower dimensionality than that of the input layer.

5.2.2 Live Experiment Results. The live experiment results on YouTube are shown in Table 1. We report results on both the engagement metric which captures user time spent on watching recommended videos, and the satisfaction metric which captures user survey responses with rating scores. We compare shared-bottom model with MMoE model, using either 4 or 8 experts. From the table, we see that using the same model complexity, MMoE significantly improves both engagement and satisfaction metrics.

Model Architecture	Number of Multiplications	Engagement Metric	Satisfaction Metric
Shared-Bottom	3.7M	/	/
Shared-Bottom	6.1M	+0.1%	+ 1.89%
MMoE (4 experts)	3.7M	+0.20%	+ 1.22%
MMoE (8 Experts)	6.1M	+0.45%	+ 3.07%

Table 1: YouTube live experiment results for MMoE.

5.2.3 Gating Network Distribution. To further understand how MMoE helps multi-objective optimization, we plot the accumulated probability in the softmax gating network for each task on each expert, as shown in Figure. 5. We see that some engagement tasks share multiple experts with other engagement tasks. And satisfaction tasks tend to share a small subset of experts with high utilization, as measured by the probability of using these experts.

As mentioned above, our MMoE layer shares one bottom hidden layer, and its gating networks take input from the shared hidden layer. This could potentially make the MMoE layer harder to modularize input information than constructing MMoE layer directly from input layer. Alternatively, we let the gating networks directly take input from the input layer instead of the shared hidden layer, so that input features can be directly used to select experts. However, live experiment results show no substantial differences compared to the MMoE layer of Figure 2b. This suggests that the MMoE's gating networks of Figure 2b can effectively modularize input information into experts for task relation and conflict modeling.

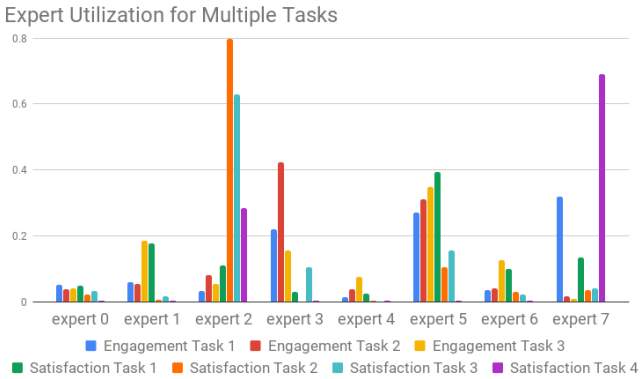

Figure 5: Expert utilization for multiple tasks on YouTube.

5.2.4 Gating Network Stability. When training neutral network models using multiple machines, distributed training strategies can cause models to diverge frequently. An example of divergences is Relu death [1]. In MMoE, the softmax gating networks have been reported [32] to have imbalanced expert distribution problem, where gating networks converge to have most zero-utilization on experts. With distributed training, we observe 20% chance of this gating network polarization issue in our models. Gating network polarization harms model performance on tasks using polarized gating networks. To solve this problem, we apply drop-out on the gating networks. By applying a 10% probability of setting utilization of experts to 0 and re-normalizing the softmax outputs, we eliminate the gating network polarization for all gating networks.

5.3 Modeling and Reducing Position Bias

One major challenge of using user implicit feedback as training data is the difficulty to model the gap between implicit feedback and true user utility. Using multiple types of implicit signals and multiple ranking objectives, we have more knobs to tune at serving time to capture the transformation from model predictions to user utility in item recommendation. However, we still need to model and reduce biases which generally exist in implicit feedback, e.g., selection biases caused by the interaction between users and current recommendation system.

Here we evaluate how we model and reduce one type of selection biases, i.e., position bias, with our proposed light-weight model architecture. Our solution avoids paying the cost of random experiments or complicated computation [41].

5.3.1 Analysis of User Implicit Feedback. To verify that position bias exists in our training data, we conduct an analysis of click through rates (CTR) for different positions. Figure 6 shows the distribution of CTR in relative scale for position 1 to 9. As expected, we see a significantly lower CTR as position gets lower and lower. The higher CTRs at higher positions are due to a combination effect of recommending more relevant items and position bias. Using our proposed approach which employs a shallow tower, we demonstrate in the following that it can separate the learning of user utility and position bias.

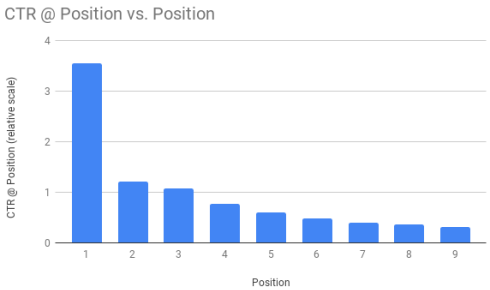

Figure 6: CTR for position 1 to 9.

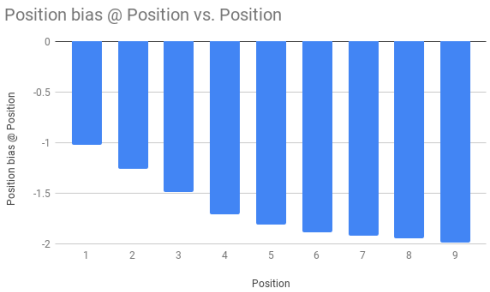

Figure 7: Learned position bias per position.

5.3.2 Baseline Methods. To evaluate our proposed model architecture, we compare it with the following baseline methods.

- Directly using position feature as an input feature: This simple approach has been widely adopted in industrial recommendation systems to eliminate position bias, mostly for linear learning to rank models.
- Adversarial learning: Inspired by the broad adoption of adversarial learning in domain adaptation [37] and machine learning fairness [5], we use a similar technique to introduce an auxiliary task which predicts position shown in training data. Subsequently, during the back propagation phase, we negate the gradient passed into the main model, to ensure that the prediction of the main model does not rely on position feature.

5.3.3 Live Experiment Results. Table 2 shows the live experiment results of our proposed method and baseline methods. We see that our proposed method significantly improves engagement metrics by modeling and reducing the position bias.

5.3.4 Learned Position Biases. Figure 7 shows learned position bias for each position. From the figure, we see that the learned bias is smaller for a lower position. The learned biases estimate the propensity scores using biased implicit feedback. Running through model training with enough training data enables us to learn to reduce position biases effectively.

Method	Engagement Metric
Input Feature	-0.07%
Adversarial Loss	+0.01%
Shallow Tower	+0.24%

Table 2: YouTube live experiment results for modeling position bias.

5.4 Discussion

In this section, we discuss a few insights and limitations which we have learned from the journey of developing and experimenting our ranking system.

5.4.1 Neural Network Model Architecture for Recommendation and Ranking. Many recommendation system research papers [18, 43] extended model architectures originally designed for traditional machine learning applications, such as multi-headed attention for natural language processing and CNN for computer vision. However, we find that many of these model architectures, which are suitable for representation learning in specific domains, are not directly applicable to our needs. This is due to:

- *Multimodal feature spaces.* Our ranking system relies on multiple sources of features, such as content feature from query and items, and context features. These features span from sparse categorical space, to natural language and image, etc. It is challenging to learn from a mixture of feature spaces.
- *Scalability and multiple ranking objectives.* Many model architectures are designed to capture one type of information, such as feature crosses [39] or sequential information [35]. They generally improve one ranking objective but may hurt others. Additionally, applying a combination of complicated model architectures in our system can hardly scale up.
- *Noisy and locally sparse training data.* Our system requires training embedding vectors for both items and queries. However, most of our sparse features follow a power-law distribution and have high variances on user feedback. For example, a user may or may not click a recommended item with same query given a slightly different context which cannot be captured in our system. This creates a great deal of difficulty in optimizing embedding space for tail items.
- *Distributed training with mini-batch stochastic gradient descent.* We rely on a large neural network model with powerful expressiveness to figure out the feature association. Since our model consumes a large amount of training data, we have to use distributed training, which itself comes with intrinsic challenges.

5.4.2 Tradeoff between Effectiveness and Efficiency. For real-world ranking systems, efficiency affects not only serving cost, but also user experiences. An overly complicated model, which significantly increases the latency in generating recommended items, can decrease user satisfaction and live metrics. Therefore, we generally prefer a simpler and more straight-forward model architecture.

5.4.3 Biases in Training Data. Besides position bias, there are many other types of biases. Some of these biases may be unknown and unpredictable, for example, due to our system's limitations in extracting training data. How to automatically learn and capture known and unknown biases in training data is a longstanding challenge requiring more research.

5.4.4 Evaluation Challenge. Since our ranking system uses mostly user implicit feedback, offline evaluation indicating how well each of our prediction tasks performs does not necessarily transfer to live performance. In fact, often times we observe misalignment between offline and online metrics. Therefore, it is preferable to choose an overall simpler model so that it can generalize better to online performance.

5.4.5 Future Directions. In addition to MMoE and removal of selection bias described above, we are improving our ranking system along the following directions:

- Exploring new model architecture for multi-objective ranking which balances stability, trainability and expressiveness. We have observed that MMoE increases multitask ranking performance by flexibly choosing which experts to share. There is more recent work which further improves model stability without hurting prediction performance [29].
- Understanding and learning to factorize. To model known and unknown biases, we want to explore model architectures and objectives which automatically identify potential biases from training data and learn to reduce them.
- Model compression. Motivated by the need to reduce serving cost, we are exploring different types of model compression techniques for ranking and recommendation models [36].

6 CONCLUSION

In this paper, we started with the description of a few real-world challenges in designing and developing industrial recommendation systems, especially ranking systems. These challenges include the presence of multiple competing ranking objectives, as well as implicit selection biases in user feedback. To tackle these challenges, we proposed a large-scale multi-objective ranking system and applied it to the problem of recommending what video to watch next. To efficiently optimize multiple ranking objectives, we extended Multi-gate Mixture-of-Experts model architecture to utilize soft-parameter sharing. We proposed a light-weight and effective method to model and reduce the selection biases, especially position bias. Furthermore, via live experiments on one of the world's largest video sharing platforms, YouTube, we showed that our proposed techniques have led to substantial improvements on both engagement and satisfaction metrics.

REFERENCES

[1] Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).

[2] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 474–482.

[3] Deepak Agarwal, Bee-Chung Chen, and Bo Long. 2011. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 609–617.

[4] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. Tfx: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1387–1395.

[5] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. 2017. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075* (2017).

[6] Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*. 89–96.

[7] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[8] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. ACM, 1–10.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[11] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 293–296.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Humaira Ehsan, Mohamed A Sharaf, and Panos K Chrysanthis. 2016. Muve: Efficient multi-objective view recommendation for visual data exploration. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 731–742.

[14] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1775–1784.

[15] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 278–288.

[16] Antonino Freno. 2017. Practical Lessons from Developing a Large-Scale Recommender System at Zalando. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 251–259.

[17] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and online evaluation of news recommender systems at swissinfo. ch. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 169–176.

[18] Qi Gu, Ting Bai, Wayne Xin Zhao, and Ji-Rong Wen. 2018. A Neural Labeled Network Embedding Approach to Product Adopter Prediction. In *Asia Information Retrieval Symposium*. Springer, 77–89.

[19] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. 2013. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 505–514.

[20] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[21] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[22] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)* 25, 2 (2007), 7.

[23] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 781–789.

[24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.

[25] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed Chi, and John Anderson. 2018. Efficient training on very large corpora via gramian estimation. *arXiv preprint arXiv:1807.07187* (2018).

[26] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 583–592.

[27] Mingsheng Long and Jianmin Wang. 2015. Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117* 2 (2015).

[28] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 4–12.

[29] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed Chi. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning. *AAAI* (2019).

[30] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. 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*. ACM, 1930–1939.

[31] Xia Ning and George Karypis. 2010. Multi-task learning for recommender system. In *Proceedings of 2nd Asian Conference on Machine Learning*. 269–284.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[33] Ayan Sinha, David F Gleich, and Karthik Ramani. 2016. Deconvolving feedback loops in recommender systems. In *Advances in Neural Information Processing Systems*. 3243–3251.

[34] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.

[35] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H Chi. 2019. Towards Neural Mixture Recommender for Long Range Dependent User Sequences. *arXiv preprint arXiv:1902.08588* (2019).

[36] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2289–2298.

[37] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7167–7176.

[38] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 165–174.

[39] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[40] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. 2016. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems* 104 (2016), 145–155.

[41] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 115–124.

[42] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual discovery at pinterest. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 515–524.

[43] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.

[44] Xiaojian Zhao, Guangda Li, Meng Wang, Jin Yuan, Zheng-Jun Zha, Zhoujun Li, and Tat-Seng Chua. 2011. Integrating rich information for video recommendation with multi-task rank aggregation. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 1521–1524.

[45] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user topic interest profiles by behavior factorization. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1406–1416.