Personalized Embedding-based e-Commerce Recommendations at eBay

Tian Wang
twang5@ebay.com
eBay Inc.

Yuri M. Brovman
ybrovman@ebay.com
eBay Inc.

Sriganesh Madhvanath
smadhvanath@ebay.com
eBay Inc.

ABSTRACT

Recommender systems are an essential component of e-commerce marketplaces, helping consumers navigate massive amounts of inventory and find what they need or love. In this paper, we present an approach for generating personalized item recommendations in an e-commerce marketplace by learning to embed items and users in the same vector space. In order to alleviate the considerable cold-start problem present in large marketplaces, item and user embeddings are computed using content features and multi-modal onsite user activity respectively. Data ablation is incorporated into the offline model training process to improve the robustness of the production system. In offline evaluation using a dataset collected from eBay traffic, our approach was able to improve the Recall@20 metric by 8.3% over the Recently-Viewed-Item (RVI) method. This approach to generating personalized recommendations has been launched to serve production traffic, and the corresponding scalable engineering architecture is also presented. In an industrial recommender system, surface rate which is defined as the percent of user page views that result in recommendations being displayed, is an important metric. Initial A/B test results show that compared to the current personalized recommendation module in production, the proposed method increases the surface rate by ~6% to generate recommendations for 90% of item page views.

CCS CONCEPTS

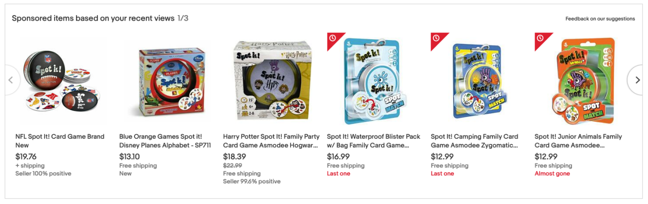
• **Computing methodologies → Learning from implicit feedback**; **Neural networks**; • **Information systems → Personalization**; **Information retrieval**; **Recommender systems**.

KEYWORDS

deep learning, personalization, recommender systems, e-commerce, cold-start

ACM Reference Format:
Tian Wang, Yuri M. Brovman, and Sriganesh Madhvanath. 2021. Personalized Embedding-based e-Commerce Recommendations at eBay. In *Proceedings of KDD '21.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '21, August 14-18, 2021, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn


Sponsored items based on your recent views 1/3
Feedback on our suggestions

NFL Spot It! Card Game Brand New
$19.76
+ shipping
Seller 100% positive

Blue Orange Games Spot it! Disney Planes Alphabet - SP711
$13.10
Free shipping
New

Harry Potter Spot It! Family Party Card Game Asmodee Hogwar...
$18.39
~~$22.99~~
Free shipping
Seller 99.6% positive

Spot It! Waterproof Blister Pack w/ Bag Family Card Game...
$16.99
Free shipping
Last one

Spot It! Camping Family Card Game Asmodee Zygomatic...
$12.99
Free shipping
Last one

Spot It! Junior Animals Family Card Game Asmodee...
$12.99
Free shipping
Almost gone

Figure 1: Screenshot of an eBay recommendations module where the user has been previously looking at games.

1 INTRODUCTION

Generating product recommendations for users is commonplace in e-commerce marketplaces. The eBay marketplace, with over 1.6 billion live items and over 183 million users, presents a unique set of challenges when it comes to generating recommendations. Traditional collaborative filtering and matrix factorization methods [1] produce poor results given the scale and extreme sparsity of eBay's user-item matrix [12]. With millions of new items listed daily, the cold start problem affects a substantial fraction of the inventory. Furthermore, over half of the live items are single quantity, that is, they can be purchased by at most one buyer. After being purchased, items are removed from the site, and no longer accessible to users. Consequently, implicit user feedback signals such as clicks and purchases are extremely sparse. In this paper, we describe how we attempt to address these unique challenges to build an effective recommender system.

Generally speaking, e-commerce recommendations may be driven purely by the shopping context or they may be personalized for a user based on a user profile. On an item page in an e-commerce marketplace, the *seed* item provides strong indication of a user's shopping mission that may be used to guide the generation of recommendations. Indeed, there are several recommender systems based on the seed item context that are deployed at eBay [3, 4, 12]. However, on other landing pages such as the homepage, such seed item context is missing. There may be other occasions as well where we need to provide personalized recommendations for the user in the absence of a seed item, and the signal for generating recommendations is primarily available user information. Such "personalized recommendations" are our primary concern in this paper. Figure 1 depicts a screenshot of an eBay recommendations module ("sponsored items based on your recent views") where the input is primarily taken from a user's activity on the marketplace.

Information about a user to generate a profile may be captured explicitly, by asking the user to fill out a survey as part of onsite registration, or implicitly, e.g. by parsing the user's shopping history. Although explicit methods directly capture the user's interests, there are several limitations with this approach: an exhaustive set of potential interests is difficult to curate, user participation tends to be low, the input can be highly incomplete, and long-term interests may not capture specific short-term shopping missions. Due to these limitations, generating a user profile of interests is commonly performed using implicit user interaction data.

In this paper, we propose to model users as embeddings based on implicitly observed user shopping behavior. Using a two-tower deep learning model architecture [8], one tower for items and one for users, users and items are represented as points in the same vector space. In order to address the data sparsity and cold-start challenges mentioned above, (i) items are represented using content-features only, and (ii) we expand the set of implicit user signals to incorporate multi-modal user onsite behaviors such as item clicking and query searching. Once trained, a k-nearest neighbor (KNN) search using a user embedding is used to generate a set of item recommendations for the user that reflect his or her implicit shopping behavior. At runtime, an additional Learning-To-Rank (LTR) model may be applied to this candidate item set in order to improve conversion, as was done in the work by Brovman et al. [4]. However this paper primarily focuses on the method for generating personalized recommendation candidate items. And since deploying a deep learning based recommendation model to a large scale dynamic industrial marketplace environment involves non-trivial engineering challenges, we also discuss details of our production engineering architecture. In summary, we contribute methods and techniques for:

(i) generating content-based item embeddings to address the cold-start problem
(ii) generating multi-modal user embeddings from various onsite events, such as item views and search queries
(iii) selectively dropping out training data to increase production model robustness
(iv) utilizing cluster-based KNN algorithm to increase recommended item diversity
(v) deployment of the model and end-to-end recommender system to eBay's large scale industrial production setting

This paper is organized in the following manner. Section 2 summarizes related work from academia as well as industry. We describe the proposed core model architecture in Section 3. The dataset as well as the offline experiments to evaluate the model are presented in Section 4. To analyze the model robustness in production environment, we conduct user data ablation analysis, and propose solutions to improve model performance. We then turn our attention to the model prediction stage in Section 5, cover retrieval as well as the production engineering architecture and discuss empirical A/B test results. Finally, we present a summary of this work and discuss future directions in Section 6.

2 RELATED WORK

The generation of personalized recommendations is a well studied problem in both academia and industry. Among the most popular techniques are matrix factorization models (e.g. [18, 22, 27]) which decompose a user–item matrix into user and item matrices, and treat recommendation as a matrix imputation problem. Despite seeing success in the Netflix competition for movie recommendation [22], traditional matrix factorization models require unique user and item identifiers, and do not perform as well in a dynamic e-commerce marketplace where existing items sell out and new items come in continuously. Utilizing content features such as the item title text becomes essential for tackling data sparsity and cold-start issues, and various methods have been proposed to address this within the matrix factorization framework. For example, Content-boosted collaborative filtering [23] uses a content-based model to create pseudo user-item ratings. Factorization machine [26] and SVDFeature [5] directly incorporate user and item features into the model.

More recently, neural networks have been used to model more complex content features and combine them in a non-linear fashion. Covington et al. [8] proposed two-tower neural networks to embed users and items separately, and applied it to the task of generating video recommendations. He et al. [16] explored the use of a non-linear affinity function to replace the dot product between the user and item embedding layers for improved model capacity. Zhu et al. [33] and Gao et al. [13] further extended the idea by using graph structures for candidate recall and scaling the non-linear affinity function for an industrial setting, for e-commerce and video recommendations respectively. Our work takes inspiration from these efforts and the practical challenges and limitations posed by the eBay marketplace.

There is a different but related line of work focusing on using neural networks for LTR, such as Deep and Wide [6] and DIN [32]. However, our work is is aimed at tackling the core candidate recall retrieval problem in an industrial setting, with the primary goal of efficiently selecting small groups of relevant items from a very large pool of candidate items. As mentioned earlier, an LTR model may be applied to this candidate item set to improve user engagement and conversion.

3 MODEL

Our proposed approach for personalized recommendations is based on training a two-tower deep learning model to generate user embeddings and item embeddings at the same time. The architecture of the model is as shown in Fig. 2, and described in detail below. We also mention the impact of adding specific model features to our primary offline model performance metric, Recall@K, described in detail in Section 4.3.

Following the work by Covington et al. [8], we model generating recommendations as a classification problem with the softmax probability:

$$P(s_i|U) = \frac{e^{\gamma(\mathbf{v}_i,\mathbf{u})}}{\sum_{j\in V} e^{\gamma(\mathbf{v}_j,\mathbf{u})}}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^D$ is a D-dimensional vector for the embedding of user U, $\mathbf{v}_i \in \mathbb{R}^D$ is a D-dimensional vector for the embedding of item s_i, γ is the affinity function between user and item, and V is all items available on eBay. As V could contain billions of items, it is infeasible to perform a full-size softmax operation. Negative sampling has to be used to limit the size of V, and we will discuss this further in Sec. 4.2. The whole model is trained to minimize the negative log-likelihood (NLL) of observed user clicks in the dataset. Next, we discuss the details of how eBay items are encoded by the model.

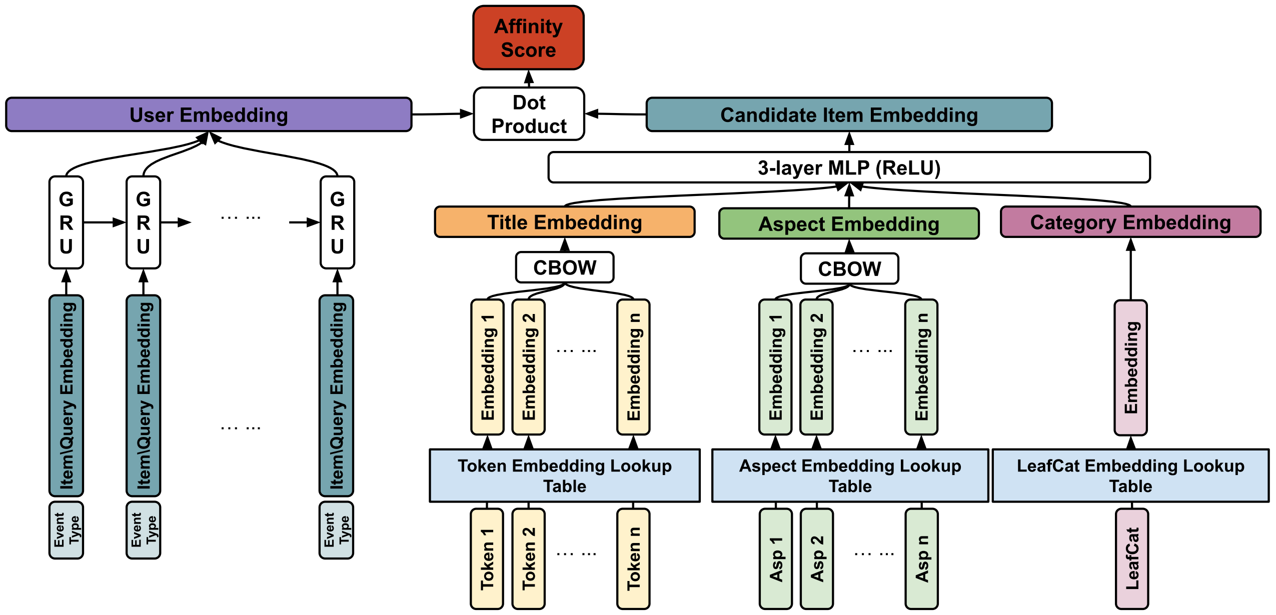


Figure 2: Model architecture with recurrent user representation.

3.1 Content-based Item Embedding

In the eBay marketplace, an item corresponds to a listing (or offer) of something for sale from a seller. In order to address the cold-start problem, an item in our model is represented not as a unique identifier (item id), but solely by using its content-based features such as item title, category (e.g mobile phone), and structured aspects (e.g brand: Apple, network: Verizon, etc.). We chose not to incorporate historical item-behavior features (e.g. historical Click-Through-Rate, Purchase-Through-Rate) in our model. These features are not applicable to cold-start items and are constantly changing by their very nature, creating additional engineering complexity for storage and retrieval when building a large-scale production system.

For title and aspect features, we tokenize and convert raw text into token embeddings with embedding size D_{text}, and use the Continuous-Bag-of-Words (CBOW) [24] approach to generate title and aspect feature representations. The vocabulary for the title feature consists of approximately 400K tokens and is gathered from eBay item titles as opposed to a generic English language corpus. This allows us to better capture the distribution of item title tokens in the eBay marketplace, which is drastically different from the traditional English language, as is demonstrated in the work by Wang and Fu [30]. Tokenization is comprised of replacing any character that is not *a-Z* or a number with whitespace, and splitting by whitespace. The vocabulary for aspect features comes from the existing production database and contains around 100K aspect tokens. For the item category feature, we index the category values and map them into an embedding space of size $D_{category}$ using a lookup table. All of the embedding tables are trained from scratch with random initialization from the standard normal distribution $\mathcal{N}(0,1)$.

After mapping all item features into a continuous space, item feature embeddings $\mathbf{z}_i$ are concatenated and passed through a MLP with L hidden layers, H hidden dimensions, and Rectified Linear Units (ReLU) [14] as the non-linear activation function, to generate a D-dimensional item embedding $\mathbf{v}_i$:

$$\begin{aligned} \mathbf{z}_i &= \text{concat}(\mathbf{z}_i^{\text{title}}, \mathbf{z}_i^{\text{aspect}}, \mathbf{z}_i^{\text{category}}), \\ \tilde{\mathbf{v}_i} &= \text{MLP}(\mathbf{z}_i). \\ \mathbf{v}_i &= \frac{\tilde{\mathbf{v}_i}}{||\tilde{\mathbf{v}_i}||} \end{aligned} \tag{2}$$

The item embedding $\mathbf{v}_i$ is normalized to unit length. We now turn our attention to the user tower part of the model.

3.2 Multi-Modal User Embedding

A user's activity on an e-commerce marketplace is not limited to only viewing items. A user may also perform actions such as making a search query, adding an item to their shopping cart, adding an item to their watch list, and so on. These actions provide valuable signals for the generation of personalized recommendations. In this work, we have attempted to create a generic framework to incorporate such "multi-modal" user activity into the model. We have chosen to start with item viewing and the search query user actions as representatives of item-based events and query-based events respectively, since these are the quintessential online shopping activities.

Item views/clicks are the most common form of implicit user feedback for an e-commerce marketplace, and generate large volumes of training data. For an item-based event z_i, we first map the corresponding item s_{z_i} to the corresponding embedding $\mathbf{v}_{z_i}$ as described in Sec. 3.1, and then concatenate it with a 4-dimensional vector $\mathbf{e}_{z_i}$ representing its event type.

User searches are a valuable signal for a recommender system as they are strong indications of explicit user interest or shopping mission. In order to encode this user action into our framework, we model each search query as a *"pseudoitem"* with the actual query text taking place of the item title, and the "dominant" query category (predicted using a separate model) taking place of the item category, and the aspects left empty. The event type embedding is concatenated to the item-based embedding. Adding this search query signal to the model resulted in a ~4% improvement in our offline validation metric, Recall@20.

We denote for each user event z_i, its corresponding vector representation $E(z_i)$ as:

$$E(z_i) = \text{concat}(\mathbf{v}_{z_i}, \mathbf{e}_{z_i}) \tag{3}$$

We explored different methods of generating a user embedding for a given user U with onsite activity $Z = \{z_1, ..., z_n\}$.

3.2.1 Continuous Bag-of-Events Representation. The first approach is to bag all the event embeddings into a single vector by averaging over all embeddings. After combining all events into a single vector, we use a MLP layer with L layers, H hidden dimension, and ReLU non-linear activation functions to generate a D-dimensional user embedding $\mathbf{u}$:

$$\begin{aligned} \tilde{\mathbf{u}} &= \text{MLP}(\frac{1}{n}\sum_{i=1}^{n} E(z_i)), \\ \mathbf{u} &= \frac{\tilde{\mathbf{u}}}{||\tilde{\mathbf{u}}||} \end{aligned} \tag{4}$$

Continuous Bag-of-Events is the simplest representation of user activity, however, in this approach, the ordering of the events does not affect the outcome.

3.2.2 Recurrent Representation. In order to integrate the ordering information of user historical events, we also experimented with using a recurrent neural network to process the sequence of event embeddings. We start with gated recurrent units [GRU, 7], which have the update rule $\mathbf{h}_t = \phi(\mathbf{x}_t, \mathbf{h}_{t-1})$ defined by:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}) \\ \mathbf{u}_t &= \sigma(\mathbf{W}_u\mathbf{x}_t + \mathbf{U}_u(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\ \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\ \mathbf{h}_t &= (1 - \mathbf{u}_t) \odot \mathbf{h}_{t-1} + \mathbf{u}_t \odot \tilde{\mathbf{h}}_t, \end{aligned} \tag{5}$$

where σ is a sigmoid function, $\mathbf{x}_t$ is the input at the t-th timestep, and $\odot$ is element-wise multiplication.

We initialize the GRU recurrent hidden state $\mathbf{l}_0$ as 0. For each event z_t in the user history Z, we feed the corresponding event embedding into the GRU cell in sequence as input in each timestep:

$$\begin{aligned} \mathbf{l}_t &= \phi(E(z_i), \mathbf{l}_{t-1}), \\ \mathbf{l}_0 &= \mathbf{0}. \end{aligned} \tag{6}$$

The D-dim user embedding $\mathbf{u}$ is generated by taking the average over output vectors from all GRU steps:

$$\begin{aligned} \tilde{\mathbf{u}} &= \frac{\sum_{i=1}^{n} \mathbf{l}_i}{n}, \\ \mathbf{u} &= \frac{\tilde{\mathbf{u}}}{||\tilde{\mathbf{u}}||} \end{aligned} \tag{7}$$

Compared to the Continuous Bag-of-Events user representation, recurrent user representation has access to the order of user activity, and in principle can better relate user relevance feedback to the user's interaction history. In our experiments, using this recurrent user representation in our model resulted in a ~5% gain in our offline Recall@20 metric.

3.3 Affinity function

The affinity function $\gamma(\mathbf{v}_i, \mathbf{u})$ between user U and item s_i is constructed by the dot product between the user and item embeddings. As user and item embeddings are normalized to have unit length ($||\mathbf{u}|| = 1$, $||\mathbf{v}_i|| = 1$), the dot product score between any pair of embeddings is constrained to have a value between -1 and 1. This essentially limits the capability of the model to distinguish positive items from negatives items. In order to address this, we added a *temperature* τ [31] term to our affinity function described in Eq. 1 as follows:

$$\gamma(\mathbf{v}_i, \mathbf{u}) = \frac{\mathbf{v}_i\mathbf{u}}{\tau}. \tag{8}$$

The temperature hyperparameter was tuned to maximize the retrieval metric, Recall@k. In our experiments, we found that τ has a large impact on the performance of the trained model. By tuning τ on the validation set, we are able to increase Recall@20 by ~150%.

4 DATASET & EXPERIMENTS

In this section, we describe the dataset we created to train our model, the importance of negative sampling during the training process, as well as the offline experiments performed to evaluate the effectiveness of the model.

4.1 Experimental Dataset

Since we treat the recommendation task as a classification problem (Eq.1), in order to train our model, we require positive and negative samples of items, where positive samples represent items relevant to the user and their shopping journey at impression time. In our e-commerce setting, an impression is defined as a single instance of an item page view by a user. eBay's e-commerce site (and mobile apps) features millions of item pages corresponding to active items, and each page contains recommendations for other items, organized into horizontal modules representing "similar items", "related items", "seller's other items", "items based on recent views" and so on. These recommendations are typically powered by module-specific recall and ranking stages. Each module presents multiple items (up to 12 on desktop web), and there may be as many as 6 such modules on each item page, distributed along the length of the page.

In order to collect positive and negative data samples, we looked at implicit user interactions with these merchandising recommendation modules on eBay's item pages, captured in the form of offline log data. Only those item page impressions that had a recorded click event on a recommendation module were considered for positive and negative data samples. Recommended items across recommendations modules that were clicked on by a user were selected as positive examples for the model target. Click events were chosen due to the volume of available data, however, other signals such as purchases may also be used. Recommended items that were not clicked on were treated as negative examples. Since clicking on a recommended item causes a new item page to be loaded, each item page impression typically resulted in one positive and multiple negative samples. As we shall discuss in the next section, the sampling strategy used for negative examples is critical to achieve good model training performance.

The data needed for the user tower was gathered over a 30 day period going back from a given page impression, and consists of up to 450 user events. All of the positive and negative items were enriched with necessary metadata about category, title, and aspects using offline tables. A typical training run would consist of around 10 million page impressions gathered from 8 days of data. A validation set with approximately 110K page impressions was collected following the end of the training data time frame, in order to avoid information leakage across training and validation sets. In order to avoid biasing the outcome towards a few users with high engagement, a given user was only allowed to contribute to one page impression in the training data and validation data. Therefore, we had 10 million unique users and 110K unique users in our training and validation datasets respectively. In order to better capture the distribution of users and their diverse shopping patterns, data was collected from logs from all of eBay's platform experiences: desktop web, mobile web, and iOS and Android native apps.

4.2 Negative Sampling

As previously mentioned, the number of available items $|V|$ on the eBay marketplace is on the scale of one billion, therefore it is infeasible to perform a full-size softmax operation as defined in Eq.1. We experimented with two approaches for sampling negative examples.

4.2.1 Observed Un-Clicked Item. In this approach, on the item page, we take the item(s) clicked on as positive, and a subset of the items that were impressed but not clicked on as our negatives. Specifically, each positive item is paired with 8 un-clicked negative items. This approach failed in our initial model training, resulting in overfitted models that were unable to generalize. The main reason for this is that on the item page, all of the impressed items from current recommendation modules are very similar to the seed item, and this leads to the effect that the model is unable distinguish positive from negative examples utilizing content-based item features.

4.2.2 In-batch Random Negative Sampling. We then experimented with using random items as negatives. Rather than randomly sampling items from the whole item pool (billions of items), we use in-batch negative sampling [17] by using the impressed but unclicked items from other training examples within the same batch as negatives. This approach gives us a less complex and more efficient sampling strategy. This approach has some similarities with a popularity-based sampling approach, as the likelihood of an item serving as a negative sample is proportional to the number of times this item is presented to a user.

4.3 Evaluation Metrics

We experimented with multiple evaluation metrics to measure model performance using our offline validation dataset. Given the similarity of our problem to the ranking problem in the information retrieval setting, we considered several metrics commonly used for ranking problems such as Normalized Discounted Cumulative Gain (NDCG), Recall@k, Precision@k, and Mean Reciprocal Rank (MRR) [11]. As mentioned in the previous section, we typically have only 1 positive in each page impression, therefore it becomes important to measure whether or not the positive recommendation is in the top k results. We therefore ultimately chose to use Recall@k as our primary evaluation metric, for $k = 1, 5, 10, 20, 40$. For P impressions, the metric is defined as:

$$Recall@k = \frac{1}{P}\sum_{i=1}^{P}\frac{\text{\# relevant items @ k}}{\text{\# total relevant items}} \tag{9}$$

For an industrial recommender system, it is important to surface the most relevant recommendations at the very top of the ranking since the user may only be shown (say) 5 recommendations via the user interface and would not engage with other recommendations.

Symbol	Hyperparameter Description	Value
D	Item/User embedding dimension	64
D_{text}	Text-based feature dimension	64
$D_{category}$	Category feature dimenstion	64
L	Number of hidden layers in MLP	3
H	Hidden dimension in MLP	64
τ	Temperature in affinity function	0.1

Table 1: Model hyperparameter settings.

4.4 Model Training

We used the PyTorch [25] deep learning framework to implement the core model. Additionally, we utilized the PyTorch-Lightning [10] framework which shortened development iterations and standardized the training loop so that it was seamless to transition the model between different CPU and GPU training environments. We chose the Adam optimizer [21] with a 0.01 learning rate. The gradient clipping parameter, set to 0.001, was essential in stabilizing the gradient in the recurrent part of the network, which spanned several hundred steps. We chose to sample 3000 negatives for each positive item, and use 600 as our batch size to maximize GPU utilization. Finally, we trained the model with 10 epochs over our data to reach convergence of the evaluation metrics. Model hyperparameters were selected considering production storage constraints and model performance on the validation set. The chosen settings are reported in Table 1.

4.5 Offline Evaluation

As an offline baseline recommendation method for comparison, we used Recently Viewed Items (RVI), which recommends items that a user has recently viewed ranked by the viewed item's recency. Although this method is simple and does not use a collaborative filtering (CF) based approach, RVI is widely used as a way of generating personalized recommendations in production systems. It is also a difficult baseline to beat in terms of generating user engagement, given that these are items the user has engaged with recently. The works of Song et al. [28] and Wang et al. [29] show approaches similar to RVI to be strong baseline methods, outperforming CF-based methods.

We evaluated our best model, which used a recurrent user representation based on item views and search query events, and the

Recall@k	RVI	Proposed Model
1	0.01	**0.02 (50%)**
5	0.06	0.06
10	0.09	0.09
20	0.12	**0.13 (8.3%)**
40	0.16	**0.18 (12.5%)**

Table 2: Offline test set evaluation results.

hyperparameters shown in Table 1. This evaluation was performed on a separate test set, which consists of 7K unique users and 10 million candidate items. The number of candidate items used for this evaluation is similar in scale to the number of candidate items typically used at prediction time in production.

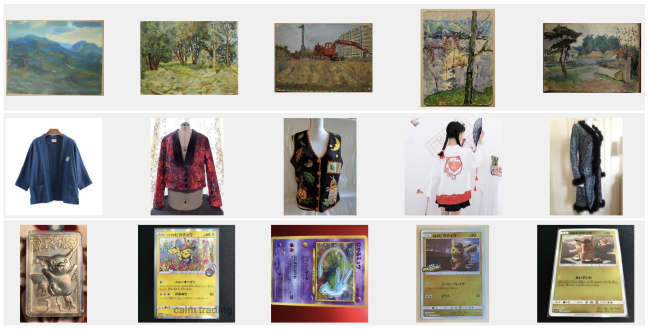


Figure 3: Sample items are grouped together by similar themes, demonstrating model quality.

Experimental results are reported in Table 2. Our method outperforms the RVI method in several Recall@k metrics that were measured. This shows that our model is able to generate appropriate personalized recommendations based on the user's current shopping mission, and potentially inspire new ones given the right user history. The RVI method, in contrast, only serves a re-targeting purpose, wherein a user is shown items they have already browsed in order to encourage re-engagement with previous shopping missions. In our approach, the multi-modal user embedding framework allows the model to incorporate various user activities seamlessly in a machine-learned manner to maximize user engagement.

Several sample items are presented in Figure 3 in order to demonstrate the model quality in a visual manner. The model generates item embeddings in a way that groups items with a similar semantic theme together. For example, the rows in Figure 3 represent item that have i) similar artistic style, ii) clothing with a similar style, and iii) Pokemon trading cards with the same character. Although the model uses text based features, we are showing the pictures of the items for brevity.

4.6 Data Ablation Analysis and Model Robustness

Stability of the model and robustness of its prediction is essential for a production environment, but is rarely studied for machine learning models that power recommender systems. We conducted an ablation analysis for our model with respect to user history data in order to study model performance under the condition where part of the user history is missing.

To understand the possible impact of missing the most recent user history at prediction time, we performed several experiments, the results of which are shown in Figure 4. First, we trained a model with the full user history present (dashed blue curve in Fig. 4), and computed predictions on a validation set while dropping different lengths of the most recent user activity (horizontal axis in Fig. 4). As can be seen by the dashed blue curve in Fig. 4, when the most recent 5-minute user activity was missing, the Recall@20 metric decreased by more than 30%, from 0.9 to 0.62. The metric degrades by as much as 50% to 0.45 when user activity within the most recent 60 minutes is missing. This creates significant performance risk for production deployment, since the model may not always have up-to-date real-time user onsite history at prediction time. To counteract this effect, we chose to train the model while dropping the most recent user activities, not random ones from the user history, in order to better align with the scenario happening in production system where there may be a gap in time between the batch model prediction output and a user impression. A user can simply be browsing the site, potentially with a new shopping mission, for some time after a batch update.

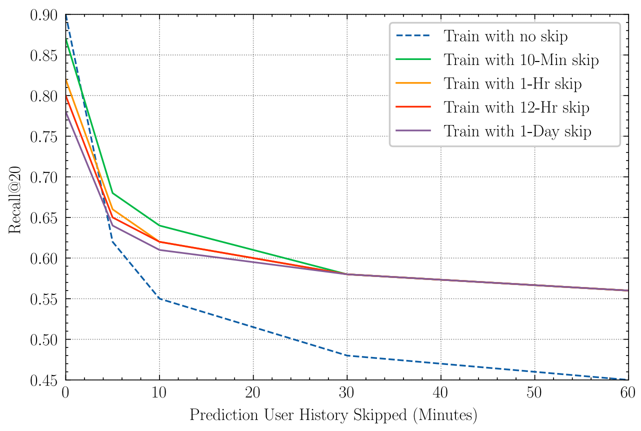


Figure 4: Model prediction performance with missing user history.

We experimented with training models by dropping some of the user onsite activity data before impression time, ranging from the most recent 10 minutes (green curve) to 1 day (purple curve). As we can see in Fig. 4, models trained using skipped user history performed better compared to the original model, when part of user history was missing at prediction time. For example, with 60 minutes of user activity missing at prediction time, all "skipped" models were able to achieve Recall@20 of 0.56, compared to 0.45 for the original model (dashed blue curve). In addition, training with more skipped history leads to a "flatter" curve, suggesting a more robust model under conditions of variable missing history. However, we also observe that model performance decreases when trained with more dropped out user history, especially across the range of 0 minute (no skipping) to 30 minutes. In order to find a balance between performance and consistency, the production model is selected as the one with the largest area under curve from Fig. 4 amongst those trained with 10-minutes of skipped user activity (green curve). This sort of user history dropout is important to consider when training a model with robust prediction expectations for a production setting.

5 MODEL PREDICTION

The previous sections focused on describing the model and offline validation testing. In this section, we will turn our attention to describing the model prediction stage, including the multitude of engineering considerations and trade-offs for building a large scale production engineering recommender system.

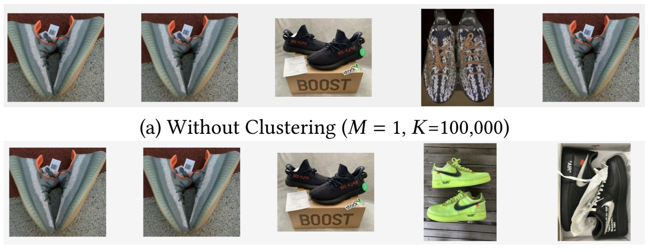
5.1 Retrieval

During the prediction stage, given a user embedding and a pool of candidate item embeddings, retrieval is conducted by the KNN search algorithm. We used the KNN implementation from FAISS [19]. For a marketplace with an enormous item-based inventory, similar items are common on the site (at the time of paper writing, searching "iphone 11" on eBay would return 3047 results). As our model only consumes content-based features (title, category, and aspects) for items, all those content-similar items would have similar embedding from our model. Utilizing the traditional KNN approach, given a user embedding, the retrieved items would be extremely overlapping in the embedding space.

To address this diversity problem, we use K-means clustering to group all candidate items into K clusters, each with a centroid c_i. At retrieval time, we try to find N candidate recall items, given a user embedding u. We first find the nearest M clusters, and in each cluster conduct a KNN search to retrieve m_i items:

$$m_i = \lceil \frac{e^{\gamma(\mathbf{c}_i,\mathbf{u})}}{\sum_{j=1}^{M} e^{\gamma(\mathbf{c}_j,\mathbf{u})}} \cdot N \rceil, \tag{10}$$

where $\gamma(\mathbf{c_j}, \mathbf{u})$ is the same affinity function defined in Eq. 1.


(a) Without Clustering (M = 1, K=100,000)

(b) With Clustering (M = 10, K=100,000)

Figure 5: Generated personalized item recommendations (a) without and (b) with clustering, where the user has been previously looking at Adidas Yeezy shoes.

Since the inventory on eBay is highly diverse, the existing product catalog does not cover all eBay items consistently. The clustering step essentially creates a "pseudo catalog" covering all items, and content-similar items could be organized into static entities. In our experiment, we found K = 100,000 provides the right level of granularity in clustering items.

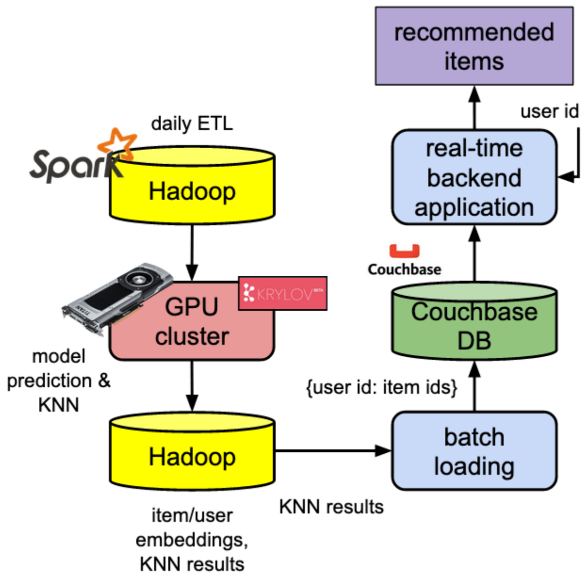


Figure 6: Production engineering architecture for model prediction.

The number of clusters, M, for retrieval is chosen by balancing between item diversity and retrieval metrics. With larger M, the retrieved items would cover more potential user interests, but with less concentration on a specific direction. With M = 1, this approach degenerates to the traditional KNN method. In our experiment, as can been shown from an example in Fig. 5, the clustering-based method generates a more diverse set of item recommendations, without losing relevancy. This technique enables controlling recommendation diversity with a simple hyperparameter, M, and can be tuned during the prediction stage separately from the training stage. In production, we found that using M = 10 and K = 100,000 provides the optimal amount of diversity of eBay item recommendations.

5.2 Production Engineering Architecture

In this section, we describe the engineering architecture used for model prediction to serve the personalized recommended items to eBay users. The production engineering architecture is depicted in Figure 6. Since a user's browsing history is constantly being updated, we recalculate the user embedding as well as the KNN results on a daily basis. Note that our model is based on content text based features, which are mostly static for any given item, so we found that we do not need to retrain the full model on a daily basis.

The prediction process is performed offline in batch mode. First, two Spark extract, transform, load (ETL) jobs generate the candidate items and the up to date user histories for all users on eBay that had activity in the last 30 days, along with the necessary metadata aggregated in Hadoop. The user behaviour data at eBay is aggregated once per day in Hadoop, so this is the reason the ETL jobs run with this daily frequency. In order to control for item popularity, we limit the candidate item set to items that have had 2 or more clicks in the past 4 days.

Next, we utilize eBay's GPU cluster, Krylov [20], to run a forward pass on the item and user towers in the trained model, in order to generate the item and user embeddings, respectively. Now, for every user embedding, a KNN search is performed on all of the item embeddings, to generate the KNN results and write them back to Hadoop. The KNN results are then loaded to a Couchbase database, which is utilized as a fast (with a latency of a few milliseconds) run-time cache, with the user id as the key, using a batch loading application. This caching approach is scalable to hundreds of millions of users, and is only limited by the Couchbase cluster capacity being used.

The eBay run-time web serving application stack is based on the Java Virtual Machine (JVM). The backend application serving recommendations is written in Scala and runs on the JVM for fast run-time performance. One of the reasons the caching architecture was chosen, is due to the latency requirements of the run-time application for serving personalized recommendations to the eBay users. Traditional approaches, such as in-memory matrix factorization, would simply not be computationally feasible at eBay data scale. In addition to enriching the recommended items with necessary metadata, the backend application can also apply a separately trained LTR model [4] in order to optimize conversion.

5.3 Online Evaluation

The best trained model was deployed to our production environment and evaluated in an online A/B test. Surface rate, defined as percent of user page views that result in recommendations being displayed, is an important metric in an industrial recommender system. For example, if 100 users view an item page and only 50 of those users are shown recommendations, the surface rate would be 50%. In general, the surface rate of a production algorithm can be below 100% due to lack of inventory, item expiration, engineering constraints, as well as business logic, among other reasons. If a user is not shown recommendations on a specific page, they cannot click, interact, or purchase those items, which is the ultimate business intent for a recommender system.

Compared to the current personalized recommendation module in production, which mainly focuses on resurfacing items a user has viewed, our model demonstrated an increase in the surface rate of ~6% to generate recommendations for 90% of item page impressions. This is mainly achieved by addressing the cold start and data sparsity problem through content-based item and user embeddings. The production RVI based algorithm is limited to using a user's item activity only, as opposed to our model which uses search activity as well. The embeddings based approach generates a much larger candidate set, which results in a higher surface rate overall. The current system, refreshed daily to capture new user activity, serves 50 million US users and will be scaled to cover all global eBay users in the near future.

6 SUMMARY AND FUTURE WORK

In this paper, we presented an approach for generating personalized item recommendations in a large scale e-commerce marketplace. A two-tower deep learning model is used to learn embeddings of items and users in a shared vector space. Items and users embeddings are learned using their content features and multi-modal onsite user activity respectively, to tackle the cold-start problem. To better address the instability of the online production environment, user data ablation is incorporated into the offline training process to generate a more robust model. Offline (improvement in Recall@20 metric by 8.3% over the RVI method) and online (improvment in surface rate of ~6% over the production algorithm) experiments have validated our approach, and shown significant improvements over baseline approaches. A personalized recommender system based on our approach has been launched to production and is now serving recommendations at scale to eBay buyers.

We are actively working to enhance the quality of the model. More types of implicit user feedback, such as "add to cart" and "watch item" should be incorporated into the model to better capture a user's shopping mission. Additionally, having only a single vector to represent a user is potentially limiting, since the user may have diverse interests and multiple ongoing shopping missions. We are working on an improved user representation that encodes parallel shopping missions, and has the capability to separate long term user interests from short term shopping missions. From the item modeling perspective, a pre-trained language understanding model (e.g BERT [9]) could be leveraged to better understand item titles. We are also working on incorporating an additional LTR model at runtime which to improve conversion metrics. This LTR model would be trained specifically for our context wherein the input is the user history and not a *seed* item.

One of the shortcomings of our current engineering implementation is the frequency of the recommendation update, which is currently limited by the need for daily batch processing. We are working on moving to a near real-time (NRT) system for generating personalized recommendations using user and item embeddings. This entails several infrastructural improvements including i) a real-time KNN service based on algorithms such as FAISS [19] and ScaNN [15] to compute distances between embedding vectors, ii) a real-time model prediction service for generating item and user embeddings (potentially using the Open Neural Network Exchange (ONNX) format [2] for transferring the trained model from the offline training to the real-time prediction environment), and iii) an event stream processing service for capturing up to date item and user actions on the eBay site. We are actively working on developing this infrastructure in the form of more general services that can be utilized for a variety of deep learning embedding-based algorithms, in addition to personalized recommendations.

ACKNOWLEDGMENTS

We would like to thank Jesse Lute for product guidance and A/B testing. We also appreciate useful feedback on the manuscript from Zhe Wu, Hongliang Yu, and Menghan Wang.

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems: The Textbook* (1st ed.). Springer Publishing Company, Incorporated.
[2] Junjie Bai, Fang Lu, Ke Zhang, et al. 2019. ONNX: Open Neural Network Exchange. https://github.com/onnx/onnx.
[3] Y. M. Brovman. 2019. *Complementary Item Recommendations at eBay Scale.* https://tech.ebayinc.com/engineering/complementary-item-recommendations-at-ebay-scale/

[4] Yuri M. Brovman, Marie Jacob, Natraj Srinivasan, Stephen Neola, Daniel Galron, Ryan Snyder, and Paul Wang. 2016. Optimizing Similar Item Recommendations in a Semi-Structured Marketplace to Maximize Conversion. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 199–202. https://doi.org/10.1145/2959100.2959166

[5] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: a toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research* 13, 1 (2012), 3619–3622.

[6] 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*. 7–10.

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

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

[9] 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).

[10] WA Falcon. 2019. PyTorch Lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

[11] Antonino Freno et al. 2015. One-Pass Ranking Models for Low-Latency Product Recommendations. In *SIGKDD* (Sydney, NSW, Australia). 10. https://doi.org/10.1145/2783258.2788579

[12] Daniel A Galron, Yuri M Brovman, Jin Chung, Michal Wieja, and Paul Wang. 2018. Deep Item-based Collaborative Filtering for Sparse Implicit Feedback. *arXiv preprint arXiv:1812.10546* (2018).

[13] Weihao Gao, Xiangjun Fan, Jiankai Sun, Kai Jia, Wenzhi Xiao, Chong Wang, and Xiaobing Liu. 2020. Deep Retrieval: An End-to-End Learnable Structure Model for Large-Scale Recommendations. *arXiv preprint arXiv:2007.07203* (2020).

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[15] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *International Conference on Machine Learning*. https://arxiv.org/abs/1908.10396

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

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

[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[20] S. Katariya and A. Ramani. 2019. *eBay's Transformation to a Modern AI Platform*. https://tech.ebayinc.com/engineering/ebays-transformation-to-a-modern-ai-platform/

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

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[23] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai* 23 (2002), 187–192.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[26] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

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

[28] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-rate deep learning for temporal recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 909–912.

[29] Tian Wang, Kyunghyun Cho, and Musen Wen. 2019. Attention-based mixture density recurrent networks for history-based recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–9.

[30] Tian Wang and Yuyangzi Fu. 2020. Item-based Collaborative Filtering with BERT. In *Proceedings of The 3rd Workshop on e-Commerce and NLP*. 54–58.

[31] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 269–277.

[32] Guorui Zhou, Xiaoqiang Zhu, Chenru 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[33] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1079–1088.