Attention-based Mixture Density Recurrent Networks for History-based Recommendation

Tian Wang
eBay
New York, New York
twang5@ebay.com

Kyunghyun Cho
New York University
New York, New York
kyunghyun.cho@nyu.edu

Musen Wen
eBay
San Jose, California
mwen@ebay.com

ABSTRACT

Recommendation system has been widely used in search, online advertising, e-Commerce, etc. Most products and services can be formulated as a personalized recommendation problem. Based on users' past behavior, the goal of personalized history-based recommendation is to dynamically predict the user's propensity (online purchase, click, etc.) distribution over time given a sequence of previous activities. In this paper, with an e-Commerce use case, we present a novel and general recommendation approach that uses a recurrent network to summarize the history of users' past purchases, with a continuous vectors representing items, and an attention-based recurrent mixture density network, which outputs each mixture component dynamically, to accurate model the predictive distribution of future purchase. We evaluate the proposed approach on two publicly available datasets, MovieLens-20M and RecSys15. Both experiments show that the proposed approach, which explicitly models the multi-modal nature of the predictive distribution, is able to greatly improve the performance over various baselines in terms of precision, recall and nDCG. The new modeling framework proposed can be easily adopted to many domain-specific problems, such as item recommendation in e-Commerce, ads targeting in online advertising, click-through-rate modeling, etc.

KEYWORDS

recommender system, personalization, mixture density network, recurrent neural network, deep learning

1 INTRODUCTION

Recommender systems have become a critical part that powers search, online advertising and e-commerce industry, etc., which generate hundred of billions of dollars of revenue. In online advertising, the recommendation system tries to recommend the best targeted ads to shown to each user; in e-Commerce search, it helps each buyer to obtain the best items of interest from extremely large inventories that generally at tens of billions size. When you come to any e-Commerce website, without starting search, based on your recent view, click and purchase activities, an extremely critical task from the recommendation system is to recomend "items you might like" that will eventually lead to an max probability of shopping event happens.

Among the most popular techniques are matrix factorization (MF) based models, e.g. [14, 17, 24] which decompose a user–item matrix into user and item matrices. Such an approach treats recommendation as a matrix imputation (completion) problem, where missing entries in the original matrix are estimated by the dot product between corresponding user and item factors. Despite their popularity in recommender systems, MF-based models have their limitations. First, MF-based models aim at reconstructing user history, instead of predicting future behaviors. The underlying assumption is that user preference is *static* over time. Second, most of such approaches omit ordering information in a user history. The purchase of a diaper is generally happened before purchasing an infant-powder, e.g. To address these issues, an increasing number of recent works have started to treat user behaviors as sequential events, and predict future events based on history [see, e.g., 9, 11, 27].

Recurrent neural networks (RNNs) are one of the most widely used techniques for sequence modelling [see, e.g., 1, 22, 26]. These RNNs have recently been considered for a history-based recommendation system [see, e.g., 11, 27, 28]. In the implicit feedback scenario, where binary user-item interactions are recorded, the previously works mostly formulate recommendation to a classification task. For an large-scale recommender system, such as in e-Commerce, it contains billions of items, it raised a very big challenge to scalability - both in training time and online prediction. For example, in the work by De Boom et al. [9], they demonstrated such an approach is capable to recommend relevant items to users on their dataset containing only less than 6 million songs. From the business side, for example, user's purchase interest is sequentially correlated or effective within time window. For example, a user who just purchased a bundle of diapers will most probably not purchase it again within a week or for her next immediate purchase. Thus, sequential modeling with RNNs is a nature choice for historical-based recommendation.

On the other hand, it has been observing that in real-world, most underlying data distributions are indeed very dynamic - sparsity, time-changing, multi-modal (mixture of distributions) of predictive distribution in stead of single family of distribution [5], etc. Very recently, the dynamic vector representation, to capture time-changing semantics, was proposed [2] and explored, where in their work, it shown that using Kalman Filter to model sematic vector transition can improve the representation of words and downstream applications.

In this work, we propose a very general modeling framework and use e-Commerce recommendation as an specific use case. Specifically, we present an approach to history-based recommendation by modeling the conditional distribution of future items' vectors. The proposed model uses a recurrent network to summarize user history, and an attention-based recurrent mixture density network, which generates each component distribution in a mixture network sequentially. In this way, we both addressing the time-varying and complex data distribution simultaneously: a multi-modal conditional distribution, which much better approximates the real-world distribution, is modeled sequentially through the Recurrent Neural Network (RNN)'s nature.

The proposed model is then evaluated on two open source datasets - MovieLens-20M [10] and RecSys15 [1]. Experimental results shows that our proposed model improves recall, precision, and nDCG significantly, compared to various other methods. Further, a a comprehensive analysis is conducted to show that by increasing the number of mixture components improves recommendations by better capturing multi-modality in user behavior, which is in accordance to real-world assumptions of multi-modality of propensity distributions in various user behaviors (click, purchase, etc.)

The proposed model explores a new direction of recommender systems by solving predictive (mixture) distribution estimation problem with continuous itemuser representation. We conclude with some directions for future future research.

2 ITEM RECOMMENDATION

2.1 Recommendation Framework

For easy illustration, we start with formulating the problem as an item recommendation under the context of e-Commerce shopping. In the implicit feedback setting, a user's behavior is recorded as a sequence of interacted items, which can be a one of various actions, such as searching, clicking, viewing and purchase. For simplicity, we focus on the clicking and viewing behavior. We frame the task of recommendation as a sequence modelling problem with the goal of predicting future clickview propensity directly.

Given a splitting index t and a user behavior sequence $S = \{s_1, s_2, ..., s_L\}$, the sequence S can be split into the history part $S_{<t} = \{s_1, ..., s_{t-1}\}$ and the future part $S_{\geq t} = \{s_t, ..., s_L\}$. A recommender system, parameterized by $\mathbf{w}$, aims at modelling the probability of clicking and viewing future items conditioned on historical items $P(S_{\geq t}|S_{<t}, w)$ that she has interacted with (i.e. clicked and viewed). To be concise, we omit $\mathbf{w}$ in our notation and assume that the items in $S_{\geq t}$ are independent. Thus the conditional probability of future items being view given history can be written as a product of individual conditional probability for each item,

$$P(S_{\geq t}|S_{<t}) = \prod_{i=t}^{L} P(s_i|S_{<t})$$

2.2 Classic Count-based Approach

The conditional probability $P(s_i|S_{<t})$ can be approximated by n-gram conditional probability

$$P(s_i|S_{<t}) \approx P(s_i|S_{t-1}^{t-(n-1)}), \tag{1}$$

[1]http://2015.recsyschallenge.com/

where

$$S_{t-1}^{t-(n-1)} = \{s_{t-1}, s_{t-2}, ..., s_{t-(n-1)}\}$$

represents the past $n-1$ viewed items. This is particular true in e-Commerce setting because shopping has temporal correlations.

An n-gram statistics table can then be constructed to record the number of occurrence for each item n-gram in the training corpus. Based on this, the approximated conditional probability can be expressed as

$$P(s_i|S_{t-1}^{t-(n-1)}) = \frac{c(s_i, s_{t-1}, s_{t-2}, ..., s_{t-(n-1)})}{\sum_j c(s_j, s_{t-1}, s_{t-2}, ..., s_{t-(n-1)})},$$

where $c(\cdot)$ is the count in the training corpus. When n equals two, such setting is a similar variant of item-to-item collaborative filtering [19], where the temporal dependency among items is ignored.

Conditioned on a seed item s_j in the user history, item-to-item collaborative filtering recommends item $\hat{s}$ having the highest co-occurrence probability

$$\hat{s} = \arg\max_s P(s|s_j).$$

The bi-gram statistics table contains the number of occurrence for each item pair $c(s_i, s_j)$, where $s_i \in S_{\geq t}, s_j \in S_{<t}$. One can estimate item-to-item conditional probability by

$$P(s_i|s_j) = \frac{c(s_i, s_j)}{\sum_k c(s_k, s_j)} = \frac{c(s_i, s_j)}{c(s_j)} \tag{2}$$

With pairwise conditional probability, $P(s_i|S_{t-1}^{t-(n-1)})$ can be approximated with $P(s_i|s_k)$, where s_k is a randomly sampled item from previous $n-1$ viewed items: $s_k \in S_{t-1}^{t-(n-1)}$.

To smooth the estimation, we take the average of the approximated probability as our final estimation

$$P(s_i|S_{t-1}^{t-(n-1)}) \approx \frac{1}{n-1} \sum_{k=t-1}^{t-(n-1)} P(s_i|s_k) \tag{3}$$

3 ATTENTION-BASED MIXTURE DENSITY RECURRENT NETWORKS

In this section, we introduce our proposed the attention-based mixture density recurrent networks to tackle the sequential recommendation problem.

The major limitation of such count-based method is data sparsity, as a large number of n-grams do not occur in the training corpus. In e-commerce, items will be sold out very quickly as well - the inventory changed quickly. To address the data sparsity issue in count-based language model, Bengio et al. [4] proposed neural language model, in which each word is represented as a continuous vector.

In our approach, we take a similar idea by representing each item s_i using a continuous vector $\mathbf{v}_i$ representation. Unlike earlier works using continuous representation as input only, and modeling the user behavior as a classification task in discrete space, our proposed approach instead choose to model user behavior as the probability density function over item embedding space $f_p(\mathbf{v}_i|S_{<t})$. Items with the highest likelihoods are recommended accordingly. By doing so, our recommender system could further leveraging the external information contained in the embedding space, and potentially

avoid the performance degradation in large-scale recommender system with classification as is shown in De Boom et al. [9].

In following subsections, we discuss our approach via three sequential components.

3.1 Item Representation

Item embedding is in itself a research problem. In our setting, we build up item embedding in a way similar to continuous skip-gram model [21] by treating a user's viewing history as a sentence, and each item as a word. Under this setting, the distance between two items in embedding space could be explained by their co-occurrence chance in a sequence. In other words, the closer the distance is, the higher chance two items have to be viewed by the same user.

Once we have the pre-trained embedding, we hold it fixed during the modeling training.

An item embedding matrix $\mathbf{E}$ can be built, where each row is the representation of the item. We denote that for each item s_i, its d-dimensional vector representation $\mathbf{v}_i$ could be written as

$$\mathbf{E}(s_i) = \mathbf{v}_i \in \mathbb{R}^d.$$

3.2 Historical Interaction Representation

A user's interaction history is described as a sequence of viewed items $S_{<t} = \{s_1, ..., s_{t-1}\}$. In this paper, we propose and experiment with three alternatives to represent user history.

Continuous Bag-of-Items Representation (CBoI). The first proposed method is to bag all the items in $S_{<t}$ into a single vector $\mathbf{s} \in [0,1]^{|\mathbf{E}|}$. Any element of $\mathbf{s}$ corresponding to the item existing in $S_{<t}$ will be assigned the frequency of that item, and otherwise 0. User history representation $\mathbf{p}$ is calculated by multiplying item embedding matrix $\mathbf{E}$ from left with $\mathbf{s}$:

$$\mathbf{p} = \mathbf{E}^\top \mathbf{s}.$$

We call this representation $\mathbf{p}$ a continuous bag-of-items (CBoI). In this approach, the ordering of history items does not affect the representation.

Recurrent Representation (RNN). Recurrent neural networks (RNN) have become one of the most popular techniques for modelling a sequence with ordering information. Long short-term memory units [LSTM, 13] and gated recurrent units [GRU, 8] are the two most popular variants of RNNs. In this paper, we start with GRUs, which have the following updating rule:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \\
\mathbf{u}_t &= \sigma(\mathbf{W}_u \mathbf{x}_t + \mathbf{U}_u(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\
\mathbf{h}_t &= (1 - \mathbf{u}_t) \odot \mathbf{h}_{t-1} + \mathbf{u}_t \odot \tilde{\mathbf{h}}_t,
\end{aligned} \tag{4}$$

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

Once we map each item to its distribute vector representation, we have a sequence of the item vectors

$$V_{<t} = \{\mathbf{E}(s_1), ..., \mathbf{E}(s_{t-1})\} = \{\mathbf{v}_1, ..., \mathbf{v}_{t-1}\}$$

After converting each item into a vector representation, the sequence of item vectors $V_{<t}$ is fed into a recurrent neural network. We initialize the recurrent hidden state as 0. For each item in the history, we get

$$\mathbf{z}_l = \phi(\mathbf{v}_l, \mathbf{z}_{l-1}), \tag{5}$$

for $l = 1, ..., t-1$, where ϕ is GRU recurrent activation function defined in Eq. (4).

With $Z_{<t} = \{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$, recurrent user representation $\mathbf{p}$ is computed by

$$\mathbf{p} = \frac{\sum_{i=1}^{t-1} \mathbf{z}_i}{t-1}.$$

Attention-based Representation (RNN-ATT). Inspired by the success of attention mechanism in machine translation [1], the proposed method incorporates attention mechanism into recurrent history representation when using with sequential generation later described in Eq. (9). After $Z_{<t}$ is generated following the same way mentioned in Eq. (5), we use a separate bidirectional recurrent neural network to read $V_{<t}$, and generate a sequence of annotated vectors $A_{<t} = \{\mathbf{a}_1, ..., \mathbf{a}_{t-1}\}$. For a mixture vector $\mathbf{m}_l$, attention-based history representation $\mathbf{p}_l$ is calculated as

$$\mathbf{p}_l = \sum_{i=1}^{t-1} \alpha_{l,i} \mathbf{z}_i, \tag{6}$$

where the attention weight $\alpha_{l,i}$ is computed by

$$\alpha_{l,i} = \frac{\exp(\text{score}(\mathbf{a}_i, \mathbf{m}_{l-1}))}{\sum_{j=1}^{t-1} \exp(\text{score}(\mathbf{a}_j, \mathbf{m}_{l-1}))}. \tag{7}$$

In Eq. (7), $\mathbf{m}_{l-1}$ is the hidden state for the $(l-1)$-th step from the recurrent neural network in sequential generation defined in Eq. (9), and $\text{score}(\mathbf{a}_j, \mathbf{m}_{l-1})$ function defines the relevance score of the j-th item with respect to $\mathbf{m}_{l-1}$.

3.3 Mixture Density Network

A mixture density network [MDN, 6] is used to formulate the likelihood of an item vector $\mathbf{v}_i$ conditioned on user history $S_{<t}$ (represented by $\mathbf{p}$) as a linear combination of kernel functions

$$f_p(\mathbf{v}_i|\mathbf{p}) = \sum_{j=1}^{m} \alpha_j(\mathbf{p})\phi_j(\mathbf{v}_i|\mathbf{p})$$

where m is the number of components used in the mixture. Each kernel is a multivariate Gaussian density function,

$$\phi_j(\mathbf{v}_i|\mathbf{p}) = \frac{\exp(-\frac{1}{2}(\mathbf{v}_i - \boldsymbol{\mu}_j(\mathbf{p}))^T \Sigma_j(\mathbf{p})^{-1}(\mathbf{v}_i - \boldsymbol{\mu}_j(\mathbf{p})))}{\sqrt{|2\pi\Sigma_j(\mathbf{p})|}}. \tag{8}$$

In order to reduce the computation complexity, the covariance matrix Σ_j is assumed to be diagonal, containing only entries for element-wise variances. Under this model, for the i-th mixture component, we need to estimate following parameters

$$\boldsymbol{\mu}_i(\mathbf{p}) \in \mathbb{R}^{d_{\text{emb}}},$$
$$\text{diag}(\Sigma_i(\mathbf{p})) \in \mathbb{R}^{d_{\text{emb}}}_{>0},$$
$$\alpha_i(\mathbf{p}) \in \mathbb{R}_{>0}.$$

Bishop [6] proposed to use a feed-forward network to generate all components *simultaneously*. We adopt the same technique in our setup.

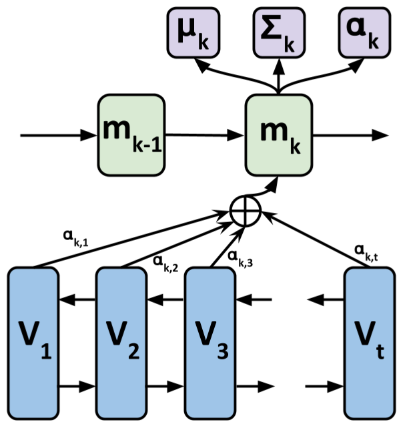


Figure 1: Architecture of sequential generation with attention-based history representation

After a user history is encoded into a single user representation $\mathbf{p} \in \mathbb{R}^{d_{\text{hidden}}}$ as explained in Sec. 3.2, the parameters for the i-th mixture are generated by

$$\begin{aligned} \boldsymbol{\mu}_i(\mathbf{p}) &= \tanh(\mathbf{W}_{\mu i}\mathbf{p}), \\ \text{diag}(\Sigma_i(\mathbf{p})) &= \log(\exp(\mathbf{W}_{\Sigma i}\mathbf{p}) + 1), \\ \alpha_i(\mathbf{p}) &= \frac{\exp(\mathbf{W}_{\alpha i}\mathbf{p})}{\sum_j \exp(\mathbf{W}_{\alpha j}\mathbf{p})}, \end{aligned}$$

where $\mathbf{W}_{\mu i} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{hidden}}}$, $\mathbf{W}_{\Sigma i} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{hidden}}}$, and $\mathbf{W}_{\alpha i} \in \mathbb{R}^{1 \times d_{\text{hidden}}}$.

Noticing that in this approach, each component is now independent to the rest components when being generated. In order to make new components aware of what has been generated, we propose further a new approach to generate components *sequentially*.

For a mixture density network with k components, a recurrent neural network iterates k steps. And in each iteration, it takes the history representation $\mathbf{p}$ as input and generates the parameters for one mixture component.

Suppose at the l-th step, the l-th component's parameters are calculated as

$$\begin{aligned} \mathbf{m}_l &= \phi(\mathbf{p}, \mathbf{m}_{l-1}), \\ \boldsymbol{\mu}_l &= \tanh(\mathbf{W}_{\mu}\mathbf{m}_l), \qquad (9) \\ \text{diag}(\Sigma_l) &= \log(\exp(\mathbf{W}_{\Sigma}\mathbf{m}_l) + 1), \end{aligned}$$

where ϕ is a GRU activation function, $\mathbf{W}_{\mu} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{hidden}}}$ and $\mathbf{W}_{\Sigma} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{hidden}}}$ are trainable weights shared among all mixtures.

After all $\{\mathbf{m}_i\}_{i=1}^{k}$ are generated, the mixture weight α_l for the l-th mixture component is calculated by

$$\alpha_l = \frac{\exp(\mathbf{W}_{\alpha}\mathbf{m}_l)}{\sum_j \exp(\mathbf{W}_{\alpha}\mathbf{m}_j)},$$

where $\mathbf{W}_{\alpha} \in \mathbb{R}^{1 \times d_{\text{hidden}}}$.

Alternatively with the attention-based history representation described in Eq. (6), $\mathbf{p}$ is replaced by $\mathbf{p}_l$ at the l-th iteration in Eq. (9).

At the l-th step, for annotated vectors $A_{<t} = \{\mathbf{a}_1, ..., \mathbf{a}_{t-1}\}$, attention weight $\alpha_{l,i}$ is computed through Eq. (7), where dot product is used as scoring function:

$$\alpha_{l,i} = \frac{\exp(\mathbf{a}_i^T \mathbf{m}_{l-1})}{\sum_{j=1}^{t-1} \exp(\mathbf{a}_j^T \mathbf{m}_{l-1})}.$$

Attention-based recurrent representation $\mathbf{p}_l$ is computed by

$$\mathbf{p}_l = \sum_{i=1}^{t-1} \alpha_{l,i}\mathbf{z}_i,$$

where $\mathbf{z}_i$ is the output from recurrent representation calculated in Eq. (5). Parameters for the l-th component is then generated by

$$\begin{aligned} \mathbf{m}_l &= \phi(\mathbf{p}_l, \mathbf{m}_{l-1}) \\ \boldsymbol{\mu}_l &= \tanh(\mathbf{W}_{\mu}\mathbf{m}_l), \\ \text{diag}(\Sigma_l) &= \log(\exp(\mathbf{W}_{\Sigma}\mathbf{m}_l) + 1). \end{aligned}$$

The architecture described above for sequential generation with attention-based history representation is illustrated in Fig. 1. The attention mechanism allows a model to automatically select items relevant to each mixture component in the user history.

4 RELATED WORK

4.1 A History-based Recommender System with a Recurrent Neural Network

RNNs were first used to model a user history in recommender systems by Hidasi et al. [11]. In this work, a RNN was used to model previous items and predict the next one in a user sequence. Tan et al. [27] improved the recommender system performance on a similar architecture with data augmentation. To better leverage item features, Hidasi et al. [12] introduced a parallel RNN architecture to jointly model user behaviors and item features. Wu et al. [28] proposed a new architecture using separate recurrent neural networks to update user and item representations in a temporal fashion.

There are two major differences in the proposed approach from the previously mentioned work. First, our work frames the task of implicit-feedback recommendation as density estimation in a continuous space rather than classification with a discrete output space. Second, unlike most of the earlier works, where the whole systems were trained end-to-end, the proposed model leverages an external algorithm to extract item representation, allowing the system to cope with new items more easily.

More recently, De Boom et al. [9] proposed a history-based recommender system with pre-trained continuous item representations as a regression problem. In their work, a recurrent neural network read through a user's history, as a sequence of listened songs, and extracted a fixed-length user taste vector, which was later used to predict future songs.

The major difference between the proposed work and the work by De Boom et al. [9] is the assumption on the number of modes in the distribution of user behaviors. The proposed model considers the mapping from history to a future behaviour as a probability distribution with multiple modes, unlike their work in which such a distribution is assumed to be unimodal. We do so by using a variant of mixture density network [6] to explicitly model user behavior. Their approach can be considered as a special case of the proposed model that degenerated to a single mixture component.

4.2 Continuous Item Representation

Inspired by the recent advance in word representation learning [21], various methods have been proposed to embed an item in a distributed vector that encodes useful information for recommendation. Barkan and Koenigstein [3] learned item vectors using an approach similar to Word2Vec [21] by treating each item as a word without considering the order of items. Liang et al. [18] jointly factorized a user-item matrix and item-item matrix to obtain item vectors. In the work by Liu et al. [20], a vector was learned for each pin in Pinterest using a Word2Vec inspired model.

In this paper, we treat item representation as a static prior knowledge, which could be extracted with external knowledge, or trained with any other models. Such setting enables the use of unrestricted architecture of item representation, and has potential to incorporate new items with content information into the existed recommender system. Under this setup, the proposed model focuses on utilizing the existed embedding space to model user behavior.

5 EXPERIMENTAL SETTINGS

5.1 Models

To illustrate the effectiveness of the newly proposed method, we experiment with the model under multiple settings. The following models are experimented and compared:

(1) CBoI-FF-*n*
(2) RNN-FF-*n*
(3) RNN-RNN-*n*
(4) RNN-ATT-RNN-*n*,

where *n* denotes the number of mixture components in a mixture density network. We experiment with four *n*'s: 1, 2, 4, and 8. The number of mixture components is used to explore the approximation to the real-data distribution, which we don't know and don't assume before hand.

Notice that, when *n* is equal to 1, a mixture model can only output a unimodal Gaussian distribution. This is similar to the work by De Boom et al. [9], where regression can be viewed as an unimodal Gaussian with an identity covariance matrix.

Further, to evaluate the effectiveness of the proposed model, we compare the new models with following benchmark models as baseline:

Recently Viewed Items (RVI). recommends items that a user has viewed in the history, ranked by viewed item's recency. Although this technique is not a collaborative filtering based method, it is widely used as a personalized recommendation module in production systems, and has demonstrated competitive performance. In the work by Song et al. [25], a similar approach (*Prev-day Click*) was adopted for temporal news recommendation, and outperformed all MF-based methods in their experiment.

Item-to-Item Collaborative Filtering (Item-CF). uses an item as a seed, and rank recommended items by their co-occurrence probability. In our use case, where seed is a sequence of items, we approximate its conditional probability as described in Eq. (3).

Implicit Matrix Factorization (IMF). [14] decomposes user-item interaction matrix into user and item matrices. For each user, items are ranked by the dot product between user vector and item vector. We use the implementation by *implicit* package[2]. The model is fit using history and future sequences in a training set, and history sequences in validation and testing sets.

All mixture density network models use 256 as d_{hidden}, and are trained using Adam [16] to maximize the log-likelihood

$$\mathcal{L}(\theta) = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{i=t}^{L^k}\log(f_p(s_i^k|S_{<t}^k,\theta)}{L^k - t + 1}$$

where K is the number of user sequences in the training set, and L^k is the length of the k-th sequence.

In all RNN-based models, one-layer GRU with 256 hidden units is used. We early-stop training based on F1@20 on the validation set, and report metric on the test set using the best model according to the validation performance.

For implicit matrix factorization, we perform grid search over the number of factors on the validation set, and report the metric using the best model on test set.

Item embeddings are trained using the continuous skip-gram model from *FastText* package [7], with the item embedding dimension d_{emb} set to be 100 and window size to be 5. All sequences in the training set are used for embedding learnings. After training, each item vector is normalized by l_2 norm.

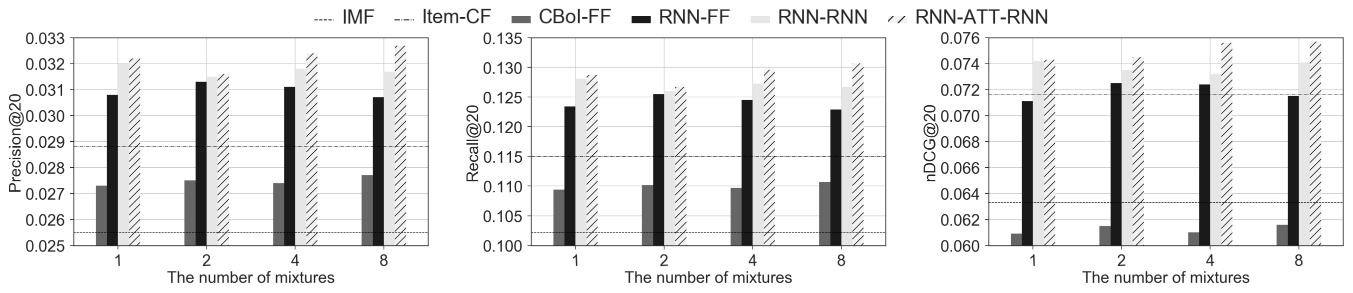
5.2 Datasets

To demonstrate the model's general utility, effectiveness and allow for re-producibility without relying on our proprietary e-Commerce data, we evaluate the proposed model on two publicly available datasets.

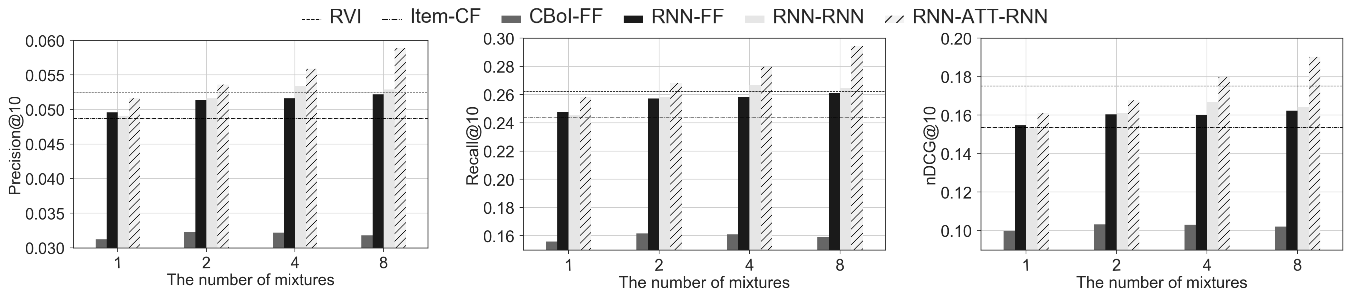
MovieLens-20M. [10] is a classic explicit-feedback collaborative filtering dataset for movie recommendation, in which (user, movie, rating, timestamp) tuples are recorded. We transform MovieLens-20M into an implicit-feedback dataset by only taking records having ratings greater than 4 as positive observations. User behavior sequences are sorted by time, and those containing more than 15 implicit positive observations are included. Each last viewed 15 movies by each user are split into 10 and 5, as history and future respectively. As the nature of this dataset, there is no duplicate items in the user sequence. After preprocessing, 75,962 sequences are kept. 80%, 10%, and 10% of sequences are randomly split into training, validation, and test sets, respectively. A movie vocabulary is built using the training set, containing 16,253 unique movies.

RecSys15. [3] is an implicit feedback dataset, containing click and purchase events from an online e-commerce website. We only work with the training file in the original dataset, and keep the click events with timestamps. We filter sequences of length less than 15, and use final 2 clicks as future, and the first 13 clicks as history. We do not filter out duplicate items, and as a result the same item could appear in both history and target parts. After preprocessing, we are left with 168,202 sequences. 80%, 10%, and 10% of sequences are randomly split into training, validation, and test sets, respectively. An item vocabulary is built only using items in the training set, leaving us with 32,117 unique items.

[2]https://github.com/benfred/implicit
[3]http://2015.recsyschallenge.com/



(a) MovieLens-20M



(b) RecSys15 [4]

Figure 2: Precision, Recall, and nDCG with varying number of mixture components on (a) MovieLens-20M and (b) RecSys15.

5.3 Metric

Various metrics can be used to evaluate the performance of a recommender system. In this paper, we concentrate on precision, recall, and nDCG. Higher values indicate better performance under these metrics.

We denote top-k recommended items by $R_k = \{r_1, r_2, ..., r_k\}$, where the items are ranked by the recommendation system, and target items by $T = \{t_1, t_2, ..., t_n\}$.

Precision@k. calculates the fraction of top-k recommended items which are overlapping with target items.

$$\text{Precision@k} = \frac{|R_k \cap T|}{k}$$

Recall@k. calculates the fraction of target items which are overlapping with top-k recommended items.

$$\text{Recall@k} = \frac{|R_k \cap T|}{|T|}$$

nDCG@k. computes the quality of ranking, by comparing the recommendation DCG with the optimal DCG [15]. In implicit feedback datasets, relevance scores for items in the target set are assigned to 1. DCG@k is calculated as

$$\text{DCG@k} = \sum_{i=1}^{|R_k|} \frac{\mathbb{I}(r_i \in T)}{\log_2(i+1)}.$$

The optimal DCG is calculated as

$$\text{DCG}_{\text{optimal}} = \sum_{i=1}^{|T|} \frac{1}{\log_2(i+1)}.$$

nDCG@k is calculated as

$$\text{nDCG@k} = \frac{\text{DCG@k}}{\text{DCG}_{\text{optimal}}}.$$

6 RESULT

Table 1 summarizes the results of our experiments, where different architectures are sorted by the metric in each column. As MovieLens has no duplicate items in a sequence, RVI is not used on that dataset. From the result on MovieLens, we first observe that the proposed RNN-ATT-RNN-4,8 model consistently outperformed the other methods in all metrics by large margins. Second, we see that CBoI-FF is the worst performing neural network model, regardless of the number of components used. Compared with that, RNN-FF model, using the same simultaneous generation but recurrent history representation, substantially improves the performance (e.g. from 0.0283 to 0.0350 for Precision@10 using 4 components). Third, comparing between the two baseline models, Item-CF outperforms IMF by a good margin across all metrics.

On RecSys15, besides the similar trends we see from MovieLens, there are several new observations. First, RVI outperforms all the models except for the RNN-ATT-RNN-{2,4,8} on Precision@10 and Recall@10. This result is in line with Song et al. [25]. They also observed the competitive performance from using previous day's clicks on news recommendation. Secondly, IMF is the worst performing model on this dataset. We conjecture that this is because that IMF only recommends items a user has not interacted with, while in clicking streams like RecSys15, items in the history are likely to reappear in the future.

To better understand the effect of mixture components on various model architecture, we group by the number of component used across various models and visualize the result in Fig. 2. We observe that RNN-ATT-RNN-*n* achieves most visible improvement as the number *n* of mixture components increases. We also find out that for all architectures with mixture density network, using two mixtures always achieves better result compared with using one mixture. However, unless the attention mechanism is used, we see diminishing improvements with more components.

The experiments have revealed that it is clearly beneficial to capture the multi-modal nature of user behavior in a recommender system. This is however only possible with the right choice of user representation and right mechanism for generating mixture parameters. In these experiments, our novel approach, the attention-based recurrent history representation combined with the sequential generation, was found to be the best model on both datasets. We have further concluded that the user preference is not static across time, and it is beneficial to model the user history as a sequence rather than a set.

In recent years, deep learning technique has been intensively adopted in building the recommendation system, such as Ni et al. [23], Zhou et al. [30] and Zhou et al. [29]. It will be interesting to see and compare the mixture density network method to these models under various use cases. This will be an interesting exploration area in future research.

7 CONCLUSION & FUTURE WORK

In this paper, we proposed a general and effective model to construct a recommender system by generating the joint probability density function for future item vectors. The proposed model combines recurrent user history representation with a mixture density network, where a novel attention-based recurrent mixture density has been proposed to output each mixture component sequentially. Experiments on two publicly available datasets, MovieLens-20M and RecSys 15, have demonstrated significant improvement in recall, precision, and nDCG compared to various strong baselines. Our proposed model shows big advantage of modelling the multi-modal nature of the predictive distribution in any recommendation system.

In this work, we start an exploration of the new modeling framework. There are several areas in which more research needs to be done to improve the overall performance. First, to better understand our model, thorough analysis on the learned mixture components and the attention weights can be conducted. Second, we propose using pre-trained embeddings using *word2vec*, which leads to embeddings that learn the distributional, user-behavior based properties of items. One way to extend our model is to incorporate content-based attributes into the item embeddings we use, and create a hybrid recommender system. Last, we have found our models was effective in our e-Commerce domain applications, and in this paper analyzed experimental results on two publicly available datasets. An widely exploration and use of the proposed method for online advertising (such as ads targeting), personalized search, shared economics (hotel, ride recommendation), etc. would be very interesting topics, and undoubtedly a potentially very rewarding exercise.

[4]Implicit Matrix Factorization resulted in the score of 0.0176, 0.0878, and 0.0402 respectively for precision@10, recall@10, and nDCG@10, and is not shown in here

ACKNOWLEDGMENTS

TW sincerely thanks Tommy Chen, Andrew Drozdov, Daniel Galron, Timothy Heath, Alex Shen, Krutika Shetty, Stephen Wu, Lijia Xie, and Kelly Zhang for helpful discussions and insightful feedbacks. KC thanks support by eBay, TenCent, Facebook, Google and NVIDIA, and was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI). MW and TW want to thank Alan Lu for research and development discussions and support.

Model	P@10	Model	P@20	Model	R@10	Model	R@20	Model	nDCG@20	Model	nDCG@20
CBoI-FF-4	0.0283	IMF	0.0255	CBoI-FF-4	0.0567	IMF	0.1022	CBoI-FF-1	0.0422	CBoI-FF-1	0.0609
CBoI-FF-1	0.0286	CBoI-FF-1	0.0273	CBoI-FF-1	0.0573	CBoI-FF-1	0.1094	CBoI-FF-8	0.0424	CBoI-FF-4	0.0610
CBoI-FF-8	0.0286	CBoI-FF-4	0.0274	CBoI-FF-2	0.0573	CBoI-FF-4	0.1097	CBoI-FF-4	0.0426	CBoI-FF-2	0.0615
CBoI-FF-2	0.0289	CBoI-FF-2	0.0275	CBoI-FF-8	0.0578	CBoI-FF-2	0.1102	CBoI-FF-2	0.0426	CBoI-FF-8	0.0616
IMF	0.0301	CBoI-FF-8	0.0277	IMF	0.0603	CBoI-FF-8	0.1107	IMF	0.0492	IMF	0.0633
Item-CF	0.0337	Item-CF	0.0288	Item-CF	0.0674	Item-CF	0.1150	RNN-FF-1	0.0514	RNN-FF-1	0.0711
RNN-FF-1	0.0343	RNN-FF-8	0.0307	RNN-FF-1	0.0686	RNN-FF-8	0.1229	RNN-FF-2	0.0524	RNN-FF-8	0.0715
RNN-FF-2	0.0348	RNN-FF-1	0.0308	RNN-FF-2	0.0695	RNN-FF-1	0.1234	RNN-FF-8	0.0525	Item-CF	0.0716
RNN-FF-4	0.0350	RNN-FF-4	0.0311	RNN-FF-4	0.0700	RNN-FF-4	0.1245	RNN-RNN-4	0.0526	RNN-FF-4	0.0724
RNN-RNN-4	0.0350	RNN-FF-2	0.0313	RNN-RNN-4	0.0701	RNN-FF-2	0.1255	RNN-FF-4	0.0528	RNN-FF-2	0.0725
RNN-FF-8	0.0351	RNN-RNN-2	0.0315	RNN-FF-8	0.0702	RNN-RNN-2	0.1260	RNN-RNN-1	0.0534	RNN-RNN-4	0.0732
RNN-RNN-1	0.0352	RNN-ATT-RNN-2	0.0316	RNN-RNN-1	0.0705	RNN-RNN-8	0.1267	RNN-RNN-2	0.0535	RNN-RNN-2	0.0735
RNN-RNN-2	0.0352	RNN-RNN-8	0.0317	RNN-RNN-2	0.0705	RNN-ATT-RNN-2	0.1267	RNN-ATT-RNN-1	0.0536	RNN-RNN-8	0.0741
RNN-ATT-RNN-1	0.0356	RNN-RNN-4	0.0318	RNN-ATT-RNN-1	0.0712	RNN-RNN-4	0.1273	RNN-RNN-8	0.0542	RNN-RNN-1	0.0742
RNN-ATT-RNN-2	0.0356	RNN-RNN-1	0.0320	RNN-ATT-RNN-2	0.0712	RNN-RNN-1	0.1281	Item-CF	0.0544	RNN-ATT-RNN-1	0.0743
RNN-RNN-8	0.0358	RNN-ATT-RNN-1	0.0322	RNN-RNN-8	0.0717	RNN-ATT-RNN-1	0.1287	RNN-ATT-RNN-2	0.0544	RNN-ATT-RNN-2	0.0745
RNN-ATT-RNN-8	0.0363	RNN-ATT-RNN-4	0.0324	RNN-ATT-RNN-8	0.0727	RNN-ATT-RNN-4	0.1296	RNN-ATT-RNN-8	0.0548	RNN-ATT-RNN-4	0.0756
RNN-ATT-RNN-4	0.0365	RNN-ATT-RNN-8	0.0327	RNN-ATT-RNN-4	0.0731	RNN-ATT-RNN-8	0.1307	RNN-ATT-RNN-4	0.0552	RNN-ATT-RNN-8	0.0757

(a) MovieLens-20M

Model	P@10	Model	P@20	Model	R@10	Model	R@20	Model	nDCG@10	Model	nDCG@20
IMF	0.0176	IMF	0.0127	IMF	0.0878	IMF	0.1267	IMF	0.0402	IMF	0.0523
CBoI-FF-1	0.0312	CBoI-FF-1	0.0215	CBoI-FF-1	0.1559	CBoI-FF-8	0.2146	CBoI-FF-1	0.0996	CBoI-FF-1	0.1215
CBoI-FF-8	0.0318	CBoI-FF-8	0.0215	CBoI-FF-8	0.1592	CBoI-FF-1	0.2148	CBoI-FF-8	0.1020	CBoI-FF-8	0.1229
CBoI-FF-4	0.0322	CBoI-FF-4	0.0217	CBoI-FF-4	0.1610	CBoI-FF-4	0.2166	CBoI-FF-4	0.1031	CBoI-FF-4	0.1241
CBoI-FF-2	0.0323	CBoI-FF-2	0.0217	CBoI-FF-2	0.1616	CBoI-FF-2	0.2171	CBoI-FF-2	0.1032	CBoI-FF-2	0.1242
Item-CF	0.0487	RVI	0.0273	Item-CF	0.2435	RVI	0.2728	RNN-RNN-1	0.1531	RNN-FF-2	0.1242
RNN-RNN-1	0.0491	RNN-FF-1	0.0320	RNN-RNN-1	0.2453	RNN-FF-2	0.3198	Item-CF	0.1536	RVI	0.1793
RNN-FF-1	0.0496	RNN-FF-2	0.0320	RNN-FF-1	0.2478	RNN-FF-1	0.3201	RNN-FF-1	0.1547	RNN-RNN-1	0.1833
RNN-FF-2	0.0514	RNN-RNN-1	0.0321	RNN-FF-2	0.2571	RNN-RNN-1	0.3212	RNN-FF-4	0.1601	RNN-FF-1	0.1839
RNN-ATT-RNN-1	0.0516	RNN-RNN-2	0.0324	RNN-RNN-2	0.2581	RNN-RNN-2	0.3235	RNN-FF-2	0.1603	Item-CF	0.1867
RNN-RNN-2	0.0516	RNN-FF-4	0.0324	RNN-ATT-RNN-1	0.2582	RNN-FF-4	0.3237	RNN-ATT-RNN-1	0.1612	RNN-FF-4	0.1875
RNN-FF-4	0.0516	RNN-FF-8	0.0327	RNN-FF-4	0.2582	RNN-FF-8	0.3269	RNN-RNN-2	0.1612	RNN-RNN-2	0.1886
RNN-FF-8	0.0522	RNN-ATT-RNN-1	0.0331	RNN-FF-8	0.2612	RNN-RNN-8	0.3306	RNN-FF-8	0.1623	RNN-FF-8	0.1906
RVI	0.0524	RNN-RNN-8	0.0331	RVI	0.2621	RNN-ATT-RNN-1	0.3313	RNN-RNN-8	0.1643	RNN-ATT-RNN-1	0.1913
RNN-RNN-8	0.0529	Item-CF	0.0334	RNN-RNN-8	0.2645	RNN-RNN-4	0.3337	RNN-RNN-4	0.1666	RNN-RNN-8	0.1925
RNN-RNN-4	0.0534	RNN-RNN-4	0.0334	RNN-RNN-4	0.2668	Item-CF	0.3342	RNN-ATT-RNN-2	0.1676	RNN-RNN-4	0.1943
RNN-ATT-RNN-2	0.0536	RNN-ATT-RNN-2	0.0336	RNN-ATT-RNN-2	0.2682	RNN-ATT-RNN-2	0.3362	RVI	0.1751	RNN-ATT-RNN-2	0.1962
RNN-ATT-RNN-4	0.0559	RNN-ATT-RNN-4	0.0344	RNN-ATT-RNN-4	0.2797	RNN-ATT-RNN-4	0.3439	RNN-ATT-RNN-4	0.1796	RNN-ATT-RNN-4	0.2054
RNN-ATT-RNN-8	0.0589	RNN-ATT-RNN-8	0.0358	RNN-ATT-RNN-8	0.2944	RNN-ATT-RNN-8	0.3578	RNN-ATT-RNN-8	0.1903	RNN-ATT-RNN-8	0.2140

(b) RecSys15

Table 1: Recall, Precision and nDCG on MovieLens and RecSys15. Results are sorted by each metric with the best model on the bottom.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Robert Bamler and Stephan Mandt. 2017. Dynamic Word Embeddings. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17)*. JMLR.org, 380–389.

[3] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 1–6.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[5] AndrÃĺ Berchtold and Adrian E. Raftery. 2002. The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time Series. *Statist. Sci.* 17, 3 (2002), 328–356.

[6] Christopher M Bishop. 1994. Mixture density networks. (1994).

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[8] 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).

[9] Cedric De Boom, Rohan Agrawal, Samantha Hansen, Esh Kumar, Romain Yon, Ching-Wei Chen, Thomas Demeester, and Bart Dhoedt. 2017. Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales. *Multimedia Tools and Applications* (2017), 1–23.

[10] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

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

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 241–248.

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

[14] 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.

[15] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

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

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

[18] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 59–66.

[19] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[20] 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.

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

[22] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.

[23] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining (KDD '18)*.

[24] 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.

[25] 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*. ACM, 909–912.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[27] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.

[28] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 495–503.

[29] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. (2018).

[30] 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 &#38; Data Mining (KDD '18)*.