IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 26, 2020, accepted July 13, 2020, date of publication July 20, 2020, date of current version July 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010508

A Variational Autoencoder Mixture Model for Online Behavior Recommendation

MINH-DUC NGUYEN[1], (Member, IEEE), AND YOON-SIK CHO[2], (Member, IEEE)
[1]Department of Software Convergence, Sejong University, Seoul 05006, South Korea
[2]Department of Data Science, Sejong University, Seoul 05006, South Korea

Corresponding author: Yoon-Sik Cho (yscho@sejong.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant 2018R1C1B5045931113, and in part by the Institute for Information & Communications Technology Promotion (IITP) Grant funded by the Korean government (MSIP) under Grant 20200005120012003.

ABSTRACT Online behavior recommendation is an attractive research topic related to social media mining. This topic focuses on suggesting suitable behaviors for users in online platforms, including music listening, video watching, e-commerce, to name but a few to improve the user experience, an essential factor for the success of online services. A successful online behavior recommendation system should have the ability to predict behaviors that users used to performs and also suggest behaviors that users never performed before. In this paper, we develop a mixture model that contains two components to address this problem. The first component is the user-specific preference component that represents the habits of users based on their behavior history. The second component is the latent group preference component based on variational autoencoder, a deep generative neural network. This component corresponds to the hidden interests of users and allows us to discover the unseen behavior of users. We conduct experiments on various real-world datasets with different characteristics to show the performance of our model in different situations. The result indicates that our proposed model outperforms the previous mixture models for recommendation problem.

INDEX TERMS Online behavior recommendation, mixture model, variational autoencoder.

I. INTRODUCTION

Recommending online user behavior is an essential component in many online platforms to improve the user experience. These platforms aim to predict behaviors such as listening to a song, watching a video, purchasing a product, that users are more likely to perform in the future and then suggest that behaviors to users. We can also understand behavior recommendation as suggesting consumed items for users. Thus, in this paper, we may use both term ''behavior'' and ''item'' with the same meaning.

With the rapid development of online service platforms, online behavior recommendation has become more crucial and attracted many scholars. Many aspects and methods for this problem have been discussed, including recommending behaviors that users used to perform (repeat behaviors) and behaviors that users never perform (new behaviors). In [1], [2], the authors suggested that repeat behaviors contribute significantly to future behaviors of users. In contrast, authors in [3] indicated that suggesting useful new behavior will decide the success of the recommendation system. In this paper, we focus on building a model that can predict well both repeat and new behaviors for users.

The mixture model [4], a probabilistic model, is an effective approach to address this problem. In mixture models, a datapoint is generated from one of multiple sources with different characteristics. Mixture weights are used to identify the impact of each source or component on the data. When applying mixture models to behavior recommendation, we assume that the behaviors of users are affected by multiple factors with different impact levels, for example, the users' habits or their interests. Multinomial mixture model [5], Adaptive mixtures model [6], Hybrid generative model [7] are some recent works that investigate this problem by proposing a mixture model consisting of the user's behavior history, the popularity of behavior or geographic distance as their components.

We extend this idea to develop a mixture model to predict future behaviors of users with a better performance. In our mixture model, one component reflects the habits of users

The associate editor coordinating the review of this manuscript and approving it for publication was Juan A. Lara.

based on their behavior history, and the other component represents the latent preference of users that is obtained through variational autoencoder. Variational autoencoder (VAE) [8] is a deep neural network architecture that aims to learn the lower-dimensional representation of data and generate the data from this representation. This model was originally introduced for image processing problems, such as image denoising, but then was applied to other domains such as topic modeling, and also behavior recommendation. Since VAE is a probabilistic model, it is also convenient to integrate this model into our mixture model. We named our model as ''Variational autoencoder mixture model''.

Our contributions can be summarized as follows:

- We propose a mixture model for online behavior recommendation based on VAE. The first component in the model represents the habits of users that are obtained through their behavior history. We named this component as ''user-specific preference''. The second component is ''latent group preference'', which is obtained through variational autoencoder.
- We apply some adjustments to VAE and achieve some promising results compared to recent models.

The remainder of this paper is structured as follows. In section II, we provide a summary of related works. Section III introduces our model. Section IV shows the experiments. Section V presents our conclusion.

II. RELATED WORKS

Repeat behaviors, or items, can be understood as the behaviors that users have performed in the past, while new behaviors are those that users never perform before. The importance of predicting repeat and new behaviors for users has been discussed in many studies [1]–[3], [9], [10]. The authors in [1], [2], [9] indicated that repeat behaviors account for a major proportion in the future behaviors of users. Many studies on the repeat behavior of users have been conducted to analyze the repeat behavior pattern from various domains, such as video watching [11], music listening [12], website re-visitation [13]. In [3], the authors suggested that predicting useful new items for users will lead to the success of recommendation systems. Predicting new behaviors can be a more challenging task, compared to predicting the repeat ones, because the new behaviors are not explicit.

To capture repeat behavior, multinomial distribution is widely used in previous studies [5], [7]. The multinomial distribution reflects the probability that a user perform any behavior and can be used to predict future repeat behaviors. Another approach is focusing on predicting repeat behavior based on sequential data or temporal features. In this way, the behaviors of a user are considered as a sequence of events, where an event is an occurrence that this user performs a behavior. An event might be associated with a time point or not. Studies on sequential or temporal data are usually developed based on Markov model [14]–[17] or Recurrent neural network [18]–[21]. This is an interesting approach; however, sequential data and temporal features can sometimes be unavailable, and models based on these data usually require a high cost in time and computing for training.

Content-based filtering and collaborative filtering are two ubiquitous traditional approaches for new behavior prediction. Content-based filtering methods [22] make recommendations by considering information about users and behaviors and finding behaviors whose description is matched with users' profiles. Collaborative filtering methods, such as Matrix factorization [23], Non-negative matrix factorization [24], Probabilistic matrix factorization [25], do not require the information about users and behaviors; instead, these methods consider behaviors of all users to find users with a similar behavioral pattern. New behaviors are suggested to users with an assumption that users who have similar behavioral patterns may perform similar behaviors or interact with similar types of items. Content-based methods may make recommendations with better accuracy than collaborative filtering methods because information about users and items are usually accurate; however, this information can be unavailable in many online platforms. In this case, collaborative filtering is useful because it only considers interactions between users and items, such as listening to a song, purchasing a product, which can be obtained conveniently.

In real-world online services, people tend to perform both repeat and new behavior; making the prediction task of both types a focal point for all platforms. A mixture model [4] that can combine different factors to predict both repeat and new behavior can be a simple but effective approach. Mixture models are probabilistic models assuming that data are generated by multiple components through a random process related to probabilistic variables. The weights of components are learned through Expectation-Maximization (EM) algorithm [26]. When applying mixture models to online behavior prediction, we can assume that the behaviors of users are affected by many factors. In [5], the authors developed a multinomial mixture model of history consumption of user and popularity of items to predict future consumption with different types of datasets, including location, music, or topic. The authors in [6] combined the locations history, the population pattern, the geographical constraint, and other social contexts to predict future locations.

We extend this line of work by developing a mixture model based on variational autoencoder (VAE) [8]. VAE is an extension of autoencoder [27], a deep neural network architecture that learns to compress input data into a lower-dimensional representation and reconstruct the data from this representation. The main difference between VAEs and standard autoencoders is that VAEs try to learn probability distributions representing the data. Thus, VAEs are generative models that can generate new data from the original dataset. Originally, VAEs and autoencoders are designed for image processing tasks such as image denoising [28], but the studies on these models have spread to various domains, including document modeling [29], [30] and also behavior recommendation [31], [32]. The application of VAEs in

recommendation problem is expected because the main idea of this model is related to lower-dimensional representation, a technique that have been employed in many recommendation methods, such as matrix factorization [23], probabilistic matrix factorization [25], non-negative factorization [24]. When employing lower-dimensional representation technique, these models consider the data from all users and learn the hidden relationships in the data. The user is represented as a vector over some latent dimensions, that can be understood as user's latent preference. This approach helps us to predict potential unseen behaviors for users based on their latent preference, which can not be done if we only consider the history of each user independently.

Mult-VAE [31] is one of the earliest work that applies VAEs to recommendation problem. In general, Mult-VAE is similar to original VAE, except that Mult-VAE uses a multinomial distribution while the original VAE uses Gaussian distribution. This work also proposes a procedure for selecting β, a parameter that controls the effect of regularization. The results in this work indicate that VAE outperforms traditional matrix factorization methods in recommendation task with remarkable margins. A big advantage of VAEs comparison over matrix factorization methods is that we can conveniently make predictions for users by simply feeding history of users into model thanks to the neural network architecture. In matrix factorization methods, we usually have to perform some optimizations when making a prediction for a new user. VAE is also more effective than matrix factorization methods in dealing with large-scale datasets. These advantages make VAEs more useful and attractive in real-world applications. Some studies [32], [33] extended Mult-VAE and make some changes, such as new prior, new architecture, new training procedure, to improve the performance. We also notice that autoencoders are also applied to recommendation problems, which can be found in [34]. However, VAEs usually have better results in prediction compared to autoencoders and can avoid many issues that autoencoders suffer, such as overfitting or growth of parameters.

Because VAEs are applied in various domains, the number of studies on VAEs increases remarkably in recent years. A common problem of VAEs that many researchers try to tackle is posterior collapse, a problem when the training procedure falls into the trivial local optimum of the objective function. When the posterior collapse happens, the generative model learns to ignore the latent variables, and the latent representations of the data becomes meaningless. A solution for this problem is adjusting the regularization term of the objective function in VAEs, which has been shared by [31], [35]. Another approach is changing the training procedure for VAEs, which has been suggested in [32], [36]. Because updating the encoder is usually more complex and challenging than updating the decoder, it requires more epochs to achieve a good performance. Therefore, the encoder (inference network) are updated multiple times for each update of the decoder (generative network). Since VAE is a deep neural network, many studies also investigate the different architectures of this model. Authors in [37] proposed a new structured inference model that shares the information between the inference network and the generative network. Generative Skip Models [38] employed skip connections to increase the mutual information between the observations and the latent variable. Another direction is investigating different distribution priors for VAEs, which have been studied in [39]–[41]

In this paper, we develop a mixture model that contains a VAE as a component. The other component is obtained by transforming the behavior history of users to a multinomial distribution over behaviors. We also apply an adjustment to the architecture of VAE, which have not been conducted for recommendation problems. The detail of our model will be discussed in the next section.

III. PROPOSED METHODS

Let U be the sets of users and I are the sets of items, then x_u is an I-dimension vector that represents behaviors of user u and x_{ui} reflects the occurrence frequency of behavior taken by user u on item i. Given the behavior history of each user over some time periods, we want to predict the consumption of that user in a future time period. There are two aspects of prediction that we consider in this study. Firstly, we want to predict behaviors that users used to perform in the past. We denote this kind of behavior as ''repeat behavior'' or ''repeat item''. Secondly, we also want to suggest behaviors that users have not performed before, which can be considered as ''new behavior'' or ''new item''.

To address these two problems, we propose a mixture model based on VAE named ''Variational Autoencoder Mixture Model'' (VAE-MM). We assume that the future behaviors of the user are influenced by two factors: (1) ''user-specific preference'' and (2) ''latent group preference''. While the user-specific preference is obtained by transforming the count data to multinomial distribution over items, the latent group preference in our proposed model is obtained through VAE. Figure 1 illustrates the architecture of our variational autoencoder mixture model.

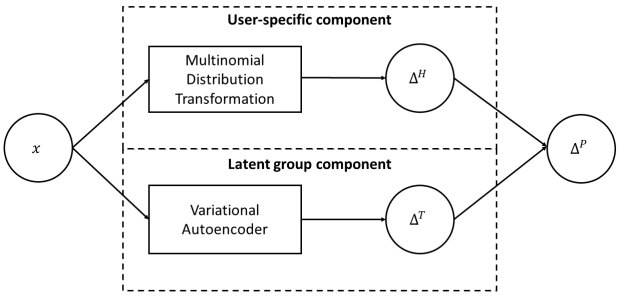


FIGURE 1. Variational autoencoder mixture model.

An important point of our mixture model is to learn the mixture weights of user-specific preference component and latent group preference component for users. We use EM algorithm to learn the mixture weights of these

components. We will describe the details of each component and the EM-algorithm in our model in the next sections.

A. MIXTURE MODEL

We denote the final prediction as Δ^P, while the prediction from user-specific preference component is denoted as Δ^H and Δ^T represents for latent group preference component. Δ^P, Δ^H and Δ^T are multinomial distributions over behaviors, indicating the probability that a user will perform a behavior in comparison with all other behaviors. This approach is more suitable than only predicting whether a user will perform a behavior or not because it is unreal for a user to have the same level of interest for all behaviors. In addition, multinomial distribution is a straightforward approach to represent the habits of users based on their behavior history.

Since we have two behavior multinomial distributions Δ^H and Δ^T from the two components, we can calculate the final probability that an user will interact with an item in the future. In mixture models, each component has a mixture weight corresponding to the impact of each component. Since our mixture model have two components: (1) The user-specific preference component and (2) The latent group preference component, we can understand that the mixture weights of two components show the impact of habits and latent interests in the future behaviors of users.

We denote π as the weight of the user-specific preference component, then the weight of latent group preference component is $1 - \pi$. Δ_u^H is the probability that a user u selects any items based on the history (or user-specific preference) and Δ_u^T is the probability that a user u selects any item based on latent group preference component. The probability Δ_u^P that user u performs any behavior is calculated as:

$$\Delta_u^P = \pi_u \Delta_u^H + (1 - \pi_u)\Delta_u^T \tag{1}$$

In our model, the weight π_u is specific for each user. We believe this is more suitable than a global weight over all user because each user has a different preference of their choices. Some users may be more affected by their habits, while others are more likely to choose an item from their latent interests.

To learn the mixture weights π for each user, we use the Expectation Maximization algorithm (EM algorithm). For the E-step, we calculate the probability λ_j of a single behavior j generated by the user-specific preference component Δ^H as:

$$\lambda_j = \frac{\pi_u p(j|\Delta_u^H)}{\pi_u p(j|\Delta_u^H) + (1 - \pi_u)p(j|\Delta_u^T)} \tag{2}$$

The M-step updates the mixing weights after summing the responsibilities for all points and normalizing the total component responsibility to sum to one:

$$\pi_u = \frac{\sum_{j=1}^{I} x_{uj}\lambda_j}{\sum_{j=1}^{I} x_{uj}} \tag{3}$$

We repeat this process until the number of iterations reaches some limit, or the likelihood of the validation data x^V for the mixture model converge. The likelihood of the validation data for the mixture model is:

$$lp(x_u^V|\Delta_u^H, \Delta_u^T, \pi_u) = \prod_{j=1}^{I}(\pi_u p(j|\Delta_u^H) + (1 - \pi_u)p(j|\Delta_u^T))^{x_{uj}} \tag{4}$$

where Δ^H and Δ^T are obtained by feeding only training data into the mixture model.

B. USER-SPECIFIC PREFERENCE COMPONENT

The first component in our mixture model is the user-specific preference, which represents for the habits of users based on their behavior history. We name this component as "user-specific" because the habits are specific for each user and only can be obtained by considering the history of each user.

In our mixture model, we use multinomial distribution to model the habits of users. We denote the vector that represent the habit of user u over all items as Δ_u^H based on their behavior history. Δ_u^H is the probability that user u will select item i based on their habits, which is calculated as:

$$\Delta_{ui}^H = \frac{x_{ui}}{\sum_i^I x_{ui}} \tag{5}$$

The more users interacts with an item in the past, the higher probability they will chose it in the future. Despite its simplicity, multinomial distribution is well-suited to model the habits of users based on their behavior history. It has been used to model the behavior history of users in [5], [31].

C. LATENT GROUP PREFERENCE COMPONENT

In our mixture model, we use VAE as the latent group preference component. We name this component as "latent group preference" because VAE in this component uses the behaviors from all users to learn the hidden relationships in the datasets. Firstly, we give a description of Mult-VAE, a general VAE for recommendation, then describe the adjustments we have applied to this model.

1) MULT-VAE

Mult-VAE is one of the earliest work that apply VAE for recommendation. In Mult-VAE, for each user u, the generative process starts by sampling a k-dimensional latent variable z_u from a Gaussian distribution with mean μ_u and variance σ_u^2. The latent variable z_u is transformed through a non-linear function $f_\theta(.) : R^k \rightarrow R^I$. The output of this transformation is normalized by a softmax function to produce a probability vector of multinomial distribution. The behavior history x_u of user u is assumed to have been drawn from this vector:

$$\begin{aligned} lz_u &\sim \mathcal{N}(\mu_u, \sigma_u^2) \\ \rho_u &= softmax(f_\theta(z_u)) \\ x_u &\sim Mult(n_u, \rho_u) \end{aligned} \tag{6}$$

To learn this generative model, we have to estimate the parameter θ. However, it is intractable to calculate the posterior distribution $p(z_u|x_u)$ directly. We use variational inference to address this problem by optimizing the evidence lower bound (ELBO):

$$rCl\mathcal{L}_\beta(x_u;\theta,\phi) = \mathbb{E}_{q_\phi(z_u|x_u)}[logp_\theta(x_u|z_u)] - \beta.KL(q_\phi(z_u|x_u)||p_\theta(z_u)) \quad (7)$$

This equation resembles the ELBO of original VAE, except the hyperparameter β that controls the strength of the regularization term.

Figure 2 illustrates the architecture of the original VAE and Mult-VAE with one hidden layer in both encoder and decoder. In general, Mult-VAE is similar to original VAE, which uses Gaussian distribution as prior. The main difference between Mult-VAE and original VAE is the activation function used in the output layer. In original VAE, a sigmoid function is used to produce an output that contains values of 0 and 1. Sigmoid function is used in the original VAE because VAE is originally trained to reconstruct image data, which are represented by vectors of 0 and 1. When applying VAE to a recommendation problem, the purpose is to predict the probability that a user will perform any behaviors; thus, a softmax function that can produce a multinomial probability distribution over behaviors is a more suitable choice. In this case, the multinomial distribution output of Mult-VAE is not exactly the ''reconstruction'' of the input x since the input x is usually count data or binary data. For each user u, to obtain the reconstruction, we simply sample $x'_u \sim Mult(n_u, \rho_u)$ where n_u is the number of events that users u perform any behavior and ρ_u is the multinomial distribution representing user u.

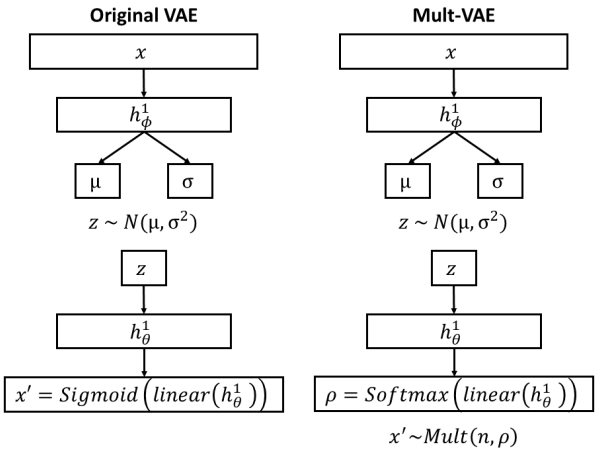


FIGURE 2. A basic architecture of original VAE and Mult-VAE with one hidden layer in both encoder and decoder. The non-linear transformation function between layers can be ReLU, tanh or sigmoid.

2) ARCHITECTURE AND ADJUSTMENTS

In this section, we describe how we modified the Mult-VAE. Figure 3 illustrates the architecture of VAE employed in our mixture model. We name this VAE as Skip-Multinomial Variational Autoencoder (SkipMult-VAE).

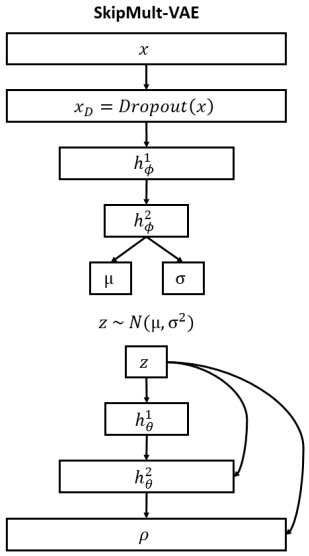


FIGURE 3. Architecture of Skip-Multinomial Variational Autoencoder (SkipMult-VAE) employed in VAE-MM.

a: DROPOUT LAYER

Dropout layer [42] is a simple technique that has been applied widely in neural network models. The idea of dropout layer is very simple: We add a layer to randomly disable some neurons in the previous layer in each training iteration. The number of disabled neurons is determined by the dropout rate, which is usually smaller than 0.5. The biggest advantage of the dropout layer is to prevent over-fitting, a common issue of neural network. By randomly deactivating some neurons, dropout layer forces the model to learn more accurate functions. In Mult-VAE [31], the authors also use a dropout layer for the input data. The results indicate that using a dropout layer helps the model to improve the prediction results. Thus, we also follow this approach in our proposed model.

b: SKIP CONNECTIONS

Another important change we made in the VAE in our mixture model is the skip connection [38]. Skip connection was proposed to avoid the posterior collapse, a common problem in VAE. When we use a VAE with many layers in the decoder, the model tend to ignore the regularization term in the objective function but still can reconstruct the data with high accuracy, leading to a poor latent representation of the data. The idea of skip connection is adding paths between latent variable z with different layers in the model, which allows sharing information and enforcing the strong link between latent variable z and the observation x. The experiment in [38] indicates that using skip connection in VAE allows the model to achieve the same performance but reduce that collapse and learn a better representation. One advantage of using skip connection is that we do not have to change the objective function, which makes it convenient to implement.

Denoting the hidden layer l of the decoder as $h_\theta^{(l)}$, our decoder with skip connection can be summarized as below:

$$\begin{aligned} h_\theta^{(1)} &= f_\theta^{(0)}(z) \\ h_\theta^{(l+1)} &= g^{(l)}(f_\theta^{(l)}(h_\theta^{(l)}), z) \text{ for } l = 1 \dots L-1 \\ \rho &= g(f_\theta(h_\theta), z) \end{aligned} \tag{8}$$

$f_\theta^{(l)}$ is the transformation function at the hidden layer l of the decoder, which is the same as in original VAE. $g^{(l)}$ is a function to combine hidden layer $l-1$ and the latent variable z. Each layer is the combination of the previous layer and latent variable z. In this experiment, we use $g^{(l)}$ as a simple function as below:

$$g^{(l)}(f_\theta^{(l)}(h_\theta^{(l)}), z) = \Omega(f_\theta^{(l)}(h_\theta^{(l)}) + W_z^{(l)} z), \tag{9}$$

where Ω is a nonlinear function, such as sigmoid, ReLU or tanh. At the output layer, Ω is a softmax function. W is the learned weight that is used to for the transformation of hidden variable z.

c: CHOOSING HYPERPARAMETER β

In Mult-VAE [31], the authors proposed a method called KL annealing to choose the best hyperparameter β. The hyperparameter β is set to 0 at the begining of training process and then is increased gradually to 1. The authors recorded the best value for β. The hyperparameter β decides the effect of regularization term in the objective function of VAE. When $\beta = 1$, the model is an original VAE. When $\beta = 0$, the model became an autoencoder. Many studies [32], [43] focused on selecting and changing β during the training process. For example, the authors in [43] proposed a procedure called cyclical annealing that increase hyperparameter β from 0 to 1 in a cycle and then immediately drop β to 0 before increasing again. The authors in [32] chose different β for each user. Although these studies have some achievements, we decide to set β to a static value in our experiment. The results in these studies show that the effect of changing β depends on the characteristics of the datasets, and it usually requires more time and effort to compute the value of β. Since our model focus on a more general mixture model that integrate VAE as a component, we leave investigating β for future work.

3) PREDICTION

We can simply make predictions by feeding the user's behavior history x_u to the trained VAE. VAE transforms the input data into a latent representation z_u by the encoder q_ϕ and generates the multinomial distribution output ρ through decoder p_θ. The latent representation z_u is constructed by taking only the mean μ instead of sampling from $\mathcal{N}(\mu, \sigma^2)$. The output ρ of Mult-VAE is also the prediction generated through the latent group preference component. We denote this prediction as Δ^T.

Using VAE in the latent group preference component brings us a big advantage when making predictions for a new user. On the other hand, for other recommendation methods, such as Matrix factorization or Latent Dirichlet Allocation, we usually have to perform an optimization with the new user's data to make predictions. This process requires some extra efforts to obtain the results, which is not suitable for industrial applications. Besides, VAE allows simple and efficient prediction. These reasons encourage us to integrate VAE as a component in our mixture model.

IV. EXPERIMENTS AND RESULTS

A. DATASETS

We conduct experiments using real-world datasets.[1] We omit the details of these datasets, which can be easily found in [5], [7]:

- `Gowalla` dataset: `Gowalla` is a location-based networking site where people share places that they have visited. The items of this dataset are physical locations. `goNYloc` and `goSFloc` are two sub-datasets which contains only location from New York and San Francisco, respectively. We filter our all places that have less than 5 visits and users who visit any location less than 10 times.
- `Twitter` dataset: Similar to `Gowalla`, this dataset is a location-based dataset that contain places from tweets of user on Twitter. We extract two sub-datasets from this datasets: `twOCloc`, that contains the locations in Orange County and `twNYloc`, that contains the locations in New York. Locations which have less than 3 interactions and users who have tweet less than 5 different days are ignored.
- `lastfm`: `lastfm` dataset consists of the music listening records from lastfm.com. In this dataset, items are artists whom users listen to. We remove any artists that have less than 100 songs and is listened by under 50 different users.
- `redditS`: `redditS` dataset contains data from Reddit, a popular social network with approximately 1.5 billion visits per month. Reddit consists of many subreddits with different topics, where users can follow and give comments. The items in this datasets are subreddits that users follow to. This dataset only contains subreddits that have more than 1000 subscribers and users who gave comments more than 1000 times since 2015.

We split the data in each dataset into three sub-datasets: training, validation, and test, similar to [5].

Table 1 summarizes the general characteristics of datasets after preprocessing. The first column contains the name of the datasets. The second column illustrates the number of users and items. The third column is the number of non-zero user-item pairs. The fourth column shows the total value of all user-item pairs, which we called as the number of events.

The fifth column contains the average repetition rate per item for each dataset. This value is obtained by calculating the average number of occurrences of each item in all users and then calculating the mean value of all items. We only use non-zero user-item pairs to calculate this value.

[1] https://archive.ics.uci.edu/ml/datasets/Repeat+Consumption+Matrices

TABLE 1. General Characteristics of Datasets Matrices: Matrix Size (Total Users and Items), Number of User-Item Pairs, Total Events and Average Repetition per Item $\bar{n}$.

	U x I size	# Pairs	# Events	Avg.Repetition
redditS	20k x 17k	391k	562k	8.3
lastfm	668 x 2439	277k	10m	28.9
goSFloc	1k x 7k	49k	86k	2
goNYloc	633 x 7k	24k	40k	1.8
twOCloc	13k x 11k	65k	291k	7.3
twNYloc	14k x 10k	167k	455k	4.2

1) CHARACTERISTICS

Table 2 provides a summary of each dataset. The first column shows the number of unique items that each user interacts with. The second column indicates the average chance that a user will repeat any behavior, which mean that the count value of non-zero user-item pair is greater than 1.

TABLE 2. Statistics in training data across all datasets: Average unique items per user and User-item pairs repetition ratio.

	Avg. Number of unique items per user	User-item pairs repetition
redditS	18.9	62%
lastfm	378	73%
goSFloc	34.9	17%
goNYloc	26.9	19%
twOCloc	8.8	43%
twNYloc	9.6	32%

lastfm has the highest numbers in both columns. It is expected when listening to an artist requires much lower energy than other kinds of activities, such as visiting a place or writing a comment, and users tend to listen to an artist more than one time. It is interesting when twOCloc and twNYloc have a low number of average unique items while goSFloc and goNYloc have a much higher value, but the repetition rate is opposite although all these datasets are location-based. This may due to the characters of each dataset. While users in twOCloc and twNYloc rarely explore new places but usually visit their familiar locations, users in goSFloc and goNYloc like to explore new places but never come back. In redditS, the average number of unique items is higher than twOCloc and twNYloc but lower than goSFloc and goNYloc while the repetition rate is much higher than all location-based dataset. We can understand this because the items in redditS relate to some ''topics'' and users usually interact with topics that they like more than one times, and commenting in an online platform costs less energy than visiting a physical locations.

Table 3 summarizes the average number of unique items per user and the average proportion of new items from each user in the test dataset. An item in a test dataset is considered as a new item for a user if it does not appear in the training set and validation set of that user. If not, that item is a repeat item.

TABLE 3. Statistics in test data across all datasets: Average unique items per user and New item ratio.

	Avg. Number of unique items per user	New items
redditS	7.9	30%
lastfm	72.8	19%
goSFloc	9.3	63%
goNYloc	8.1	68%
twOCloc	1.8	33%
twNYloc	1.6	50%

The ratios of new items in location-based dataset are higher than other datasets as expected because people are more likely to listen to a familiar artist or comment in a familiar topic than re-visit a place. The high rate of new items makes the prediction for location-based datasets more challenging than others. The diversity of these dataset allows us to validate the performance and the adaptation ability of models with various real-world scenarios.

B. EVALUATION METHODS

In our experiments, we use different metrics to evaluate the performance of our model. These metrics are all based on ranking of items. For each user, we sort the predicted items with respect to Δ^P_{ui}. Recall@k is used to evaluate the ability of model to assign a high rank to items. Recall@k is calculated as:

$$\text{Recall@}k = \frac{1}{N_{test}} \sum_u \sum_j \frac{x_{uj} \boldsymbol{I}((rank(u,j) \leq k)}{\sum_{j'} x_{uj'}} \quad (10)$$

This metric measures what fraction of items in test dataset were ranked in the top k by our model for user u. For user u, given the rank of all items, if the rank $rank(u; j)$ of item j in the test dataset is in the top k predicted behavior, the accuracy is 1; otherwise, it is 0. We denote the function for assigning accuracy by $\boldsymbol{I}((rank(u; j) \leq k)$. The notation n_{uj} corresponds to the number of occurrences of user u on item j. We denote N_{test} as the total number of users in the test dataset.

We also use the average rank of all items to evaluate the model. This metric is computed as below:

$$\text{AverageRank} = \frac{1}{\sum_u \sum_{j'} x_{uj'}} \sum_u \sum_j x_{uj} rank(u,j) \quad (11)$$

Average rank gives us an overall view of the prediction performance. While Recall@k only focuses on high-rank items, average rank considers every candidate, which is useful if we want to evaluate the predictive performance of novel items that do not appear in the history of the user. In our experiments, we set k to 100, which is similar to the metrics of the previous studies on this problem [5], [7].

We also use Normalized discounted cumulative gain (NDCG) to evaluate the model. This metric focuses on the order of items. To obtain NDCG, we calculate the discounted cumulative gain (DCG) for predicted items of each user, and then normalize this value by dividing by the best possible DCG, or ideal DCG (IDCG) of each user. In our experiments, NDCG is calculated as below:

$$DCG_u = \sum_{j}^{I} \frac{x_{uj}}{log_2(rank(u,j)+1)}$$
$$NDCG = \frac{1}{N_{test}} \sum_{u} \frac{DCG_u}{IDCG_u} \quad (12)$$

We also compute the Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) curve to provide a holistic view. The ROC illustrates the performance of models with different thresholds and allows us to see the difference between models conveniently.

C. EXPERIMENT SETUP

In our experiments, we first train the model with training dataset and evaluate with validation data to learn the best parameters, and then combine training and validation data and feed them to the model to make prediction and evaluate the results with test data.

For VAE in latent group preference component, we set the dimensions of the latent representation K to 200. In the encoder, we use 2 hidden layers with dimension of 2000 and 1000. In the decoder, the dimensions of hidden layers are 1000 and 2000 and skip connections are applied. We find that adding more layers beyond two layers does not achieve better performance. Tanh nonlinearity is used as the activation function between each layer. We add a dropout layer after the input layer with dropout rate 0.5. We train VAE by Adam with batch size of 64. We train for 50 epochs on all datasets.

We use two mixture models for behavior recommendation as the baseline models and keep the same setting as the original studies:

- **Multinomial Mixture Model (MMM)** [5]: MMM is a mixture model that contains two components: ''Individual'' component reflecting the individual's past behaviors, and ''Population'' component representing the broader population (popularity) of behaviors.
- **Hybrid Generative Model (HGM)** [7]: HGM is a recent mixture model based on Latent Dirichlet Allocation (LDA) [44]. In addition to the two components representing the user's history behavior and behavior popularity as MMM, HGM employs a component to reflect the hidden preferences of users that are obtained through LDA.

MMM and HGM are two state-of-the-art mixture models for behavior recommendation. The experiments in these studies show that these models outperform the traditional model in recommending repeat and new behavior for users. Thus, we only use these models as the baseline. Since two baseline models and our VAE-MM all employ the behavior history of users as a component, the other components in each model will affect the predictive performance between these models.

D. RESULTS

1) OVERALL PERFORMANCE

Table 4 summarizes the results in three different metrics for all items in the test datasets. Our proposed model consistently outperforms the baseline models for most datasets in all metrics. The only exception is NDCG for `lastfm`, `goSFloc` and `goNYloc` when we have a slightly smaller value, which could stem from the characteristic of these datasets.

TABLE 4. Recall@100, average rank, AUC and NDCG across different data sets. Higher scores are better for Recall@100, AUC and NDCG. Lower scores are better for average rank. Best-performing methods indicated in bold font.

		Recall@100	Avg.Rank	AUC	NDCG
redditS	MMM	0.823	202	**0.99**	0.679
	HGM	0.834	188	**0.99**	**0.682**
	VAE-MM	**0.856**	**114**	**0.99**	**0.682**
lastfm	MMM	0.689	170	0.93	**0.636**
	HGM	0.689	151	0.94	**0.636**
	VAE-MM	**0.699**	**131**	**0.95**	0.631
goSFloc	MMM	0.486	922	0.87	0.474
	HGM	0.504	857	0.88	**0.479**
	VAE-MM	**0.515**	**727**	**0.90**	0.473
goNYloc	MMM	0.454	1265	0.81	0.430
	HGM	0.467	1181	0.82	**0.436**
	VAE-MM	**0.469**	**1005**	**0.85**	0.434
twOCloc	MMM	0.758	345	0.96	0.597
	HGM	0.788	240	0.97	0.607
	VAE-MM	**0.817**	**176**	**0.98**	**0.611**
twNYloc	MMM	0.629	560	0.95	0.456
	HGM	0.646	495	0.95	0.467
	VAE-MM	**0.679**	**393**	**0.96**	**0.474**

We further investigate the results for repeat and new items separately in the following sections. An item in the test dataset is considered as a new item for a user if it does not appear in the training set and the validation set of that user. Otherwise, that item is a repeat item.

2) REPEAT BEHAVIOR PREDICTION PERFORMANCE

Table 5 displays the values of three metrics for repeat items in all datasets. Since all three models employ consumption history of user as a component, there is a slight increase in the results for repeat items of our model. However, NDCG of VAE-MM for `redditS` and `lasttm` is marginally lower than MMM and HGM. Because NDCG focuses on the order of items, the order of items has higher impacts on the results than other metrics.

The high values for repeat items of all metrics also point out that models tend to assign a high rank to repeat items.

TABLE 5. Recall@100, average rank, AUC and NDCG on the repeat items across different data sets. Higher scores are better for Recall@100, AUC and NDCG. Lower scores are better for average rank. Best-performing methods indicated in bold font.

		Recall@100	Avg.Rank	AUC	NDCG
redditS	MMM	**0.999**	**7**	**0.99**	**0.777**
	HGM	**0.999**	**7**	**0.99**	**0.777**
	VAE-MM	**0.999**	**7**	**0.99**	0.772
lastfm	MMM	0.780	97	0.96	**0.679**
	HGM	0.781	85	**0.97**	0.678
	VAE-MM	**0.784**	**75**	**0.97**	0.668
goSFloc	MMM	0.966	29	**0.99**	0.596
	HGM	0.968	28	**0.99**	0.603
	VAE-MM	**0.975**	**28**	**0.99**	**0.611**
goNYloc	MMM	0.973	29	**0.99**	0.554
	HGM	0.967	28	**0.99**	0.561
	VAE-MM	**0.988**	**24**	**0.99**	**0.569**
twOCloc	MMM	0.999	7	**0.99**	0.604
	HGM	0.999	7	**0.99**	0.608
	VAE-MM	**0.999**	**6**	**0.99**	**0.613**
twNYloc	MMM	0.999	5	**0.99**	0.441
	HGM	0.999	5	**0.99**	0.449
	VAE-MM	**0.999**	**5**	**0.99**	**0.456**

TABLE 6. Recall@100, average rank, AUC and NDCG on the new items across different data sets. Higher scores are better for Recall@100, AUC and NDCG. Lower scores are better for average rank. Best-performing methods indicated in bold font.

		Recall@100	Avg.Rank	AUC	NDCG
redditS	MMM	0.216	1721	0.88	0.189
	HGM	0.281	1599	0.88	0.202
	VAE-MM	**0.373**	**938**	**0.93**	**0.223**
lastfm	MMM	0.011	1003	0.59	0.362
	HGM	0.011	861	0.65	0.369
	VAE-MM	**0.023**	**750**	**0.71**	**0.376**
goSFloc	MMM	0.123	1910	0.73	0.213
	HGM	0.154	1776	0.75	**0.219**
	VAE-MM	**0.177**	**1501**	**0.79**	0.213
goNYloc	MMM	0.153	2401	0.64	0.193
	HGM	**0.170**	2241	0.67	**0.197**
	VAE-MM	**0.170**	**1909**	**0.72**	0.196
twOCloc	MMM	0.238	1844	0.79	0.074
	HGM	0.327	1278	0.85	0.083
	VAE-MM	**0.418**	**925**	**0.89**	**0.085**
twNYloc	MMM	0.237	1692	0.84	0.089
	HGM	0.269	1496	0.86	0.094
	VAE-MM	**0.337**	**1186**	**0.88**	**0.098**

Because repeat behaviors account for a large percentage of behaviors in the test datasets, the impact of new behavior prediction on overall result is implicit. Thus, considering only repeat items is not enough to evaluate models and we need to investigate the new items prediction.

3) NEW BEHAVIOR PREDICTION PERFORMANCE

Table 6 shows that VAE-MM outperforms all baseline models for most datasets in all metrics in new behavior prediction task. An exception is Recall@100 and NDCG for goNYloc, where the Recall@100 of HGM and VAE-MM for this dataset is equal and NDCG of VAE-MM for goNYloc is slightly lower than HGM. As we discuss above, the data in goNYloc is too sparse and too random, making the prediction of new items more difficult. Another reason is that VAE-MM does not employ the popularity of items, which may explain the appearance of some random items. However, average rank and AUC are improved considerably, showing that VAE-MM can deal with sparse data better than the baseline models. For goSFloc, NDCG is also slightly lower but Recall@100, average rank and AUC improve remarkably. The results from goSFloc and goNYloc have a similar pattern because these two are the sub-datasets of Gowalla datasets and share some characteristics.

The improvements in Recall@100 for twNYloc and twOCloc are much higher than those in goSFloc and goNYloc, although all four datasets are from location-based services. This difference can be explained when considering the characteristics of twNYloc and twOCloc datasets. These datasets are not as sparse as goSFloc and goNYloc; thus, VAE-MM can explore more hidden behaviors. The improvement in redditS is also remarkable in all three metrics. Since the items in redditS are sub-reddits, which are usually related to some semantic topics and have strong latent relationships, the increase in the performance of VAE-MM is predictable. The improvements in twNYloc, twOCloc and redditS show that VAE is more effective when the hidden relationships exist in data.

Recall@100, average rank, AUC and NDCG for lastfm improve considerably compared to the baseline models. We also notice that NDCG of VAE-MM on all items and repeat items in lastfm is lower than NDCG of other models but NDCG on new items is higher. According to table 3, the number and ratio of repeat items in lastfm is by far the highest over all datasets, and users in lastfm tend to repeat their behavior several times instead of exploring the new artists. This leads to a trade-off between exploiting repeat behaviors and exploring new behaviors in lastfm: if models suggest only repeat behaviors for high rank, NDCG on all items will obtain a high value; however, if models recommend more unseen behaviors, NDCG on new items increases but that on repeat and all items decreases. This is a limitation of NDCG when it emphasizes the impact of high-rank items; thus, average rank and AUC are necessary to show the comprehensive performance of models.

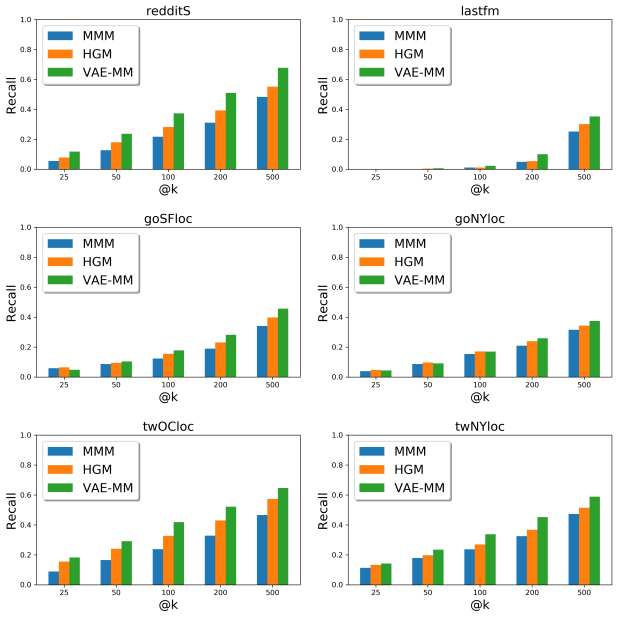


FIGURE 4. Recall@k of MMM, HGM and VAE-MM with k varied from 25 to 500 for new items across all datasets. The results for MMM are in blue (left), the results for HGM are in orange (middle) and the results for VAE-MM are in green (right).

We illustrate Recall@k for new items across all datasets with different values of k in figure 4. In general, VAE-MM outperforms baseline models constantly for all datasets. The only exception is Recall@k for `goSFloc` and `goNYloc`. When k is small, the performance of VAE-MM is slightly worse than baseline models. When we set k to higher values, the performance of VAE-MM is significantly better. `goSFloc` and `goNYloc` are sparse and random; thus, the integration of the popularity of items, which VAE-MM lacks, allows MMM and HGM to achieve some better results. When k increases, the latent group preference component in VAE-MM helps this model to outperform the baselines. We also notice that Recall@k for `lastfm` is closed to zero when k is small because of the high number of repeat items in `lastfm`.

Figure 5 illustrates the ROC curves of three models for new item predictions. The gap between VAE-MM and the baseline models is significant, which shows that VAE-MM outperforms consistently. We believe this improvement comes from the integration of the latent group preference component using VAE in VAE-MM. This component makes the recommendation from the individual's hidden representation that is learnt through VAE. Because the latent representation reflects the latent interests of users, which is heterogeneous to all user, VAE can suggest different but suitable unseen items for each user. In contrast, MMM only uses the popularity of items to predict new items, leading to a same suggestion of new items across all users in the datasets. The latent group preference component, which is missing in MMM, allows VAE-MM to outperform MMM in new items recommendation task because it is unrealistic that all users will interact with same sets of new items in the future. Similar to VAE-MM, HGM also employs a latent group preference component to model the hidden interests of users. The latent group preference component in HGM is based on LDA, instead of VAE which is used in VAE-MM. The results in our experiments indicate that VAE has a better performance than LDA in new items recommendation task. A reason for this result could be that Gaussian distribution used in VAE suits the characteristics of the datasets more than Dirichlet distribution.

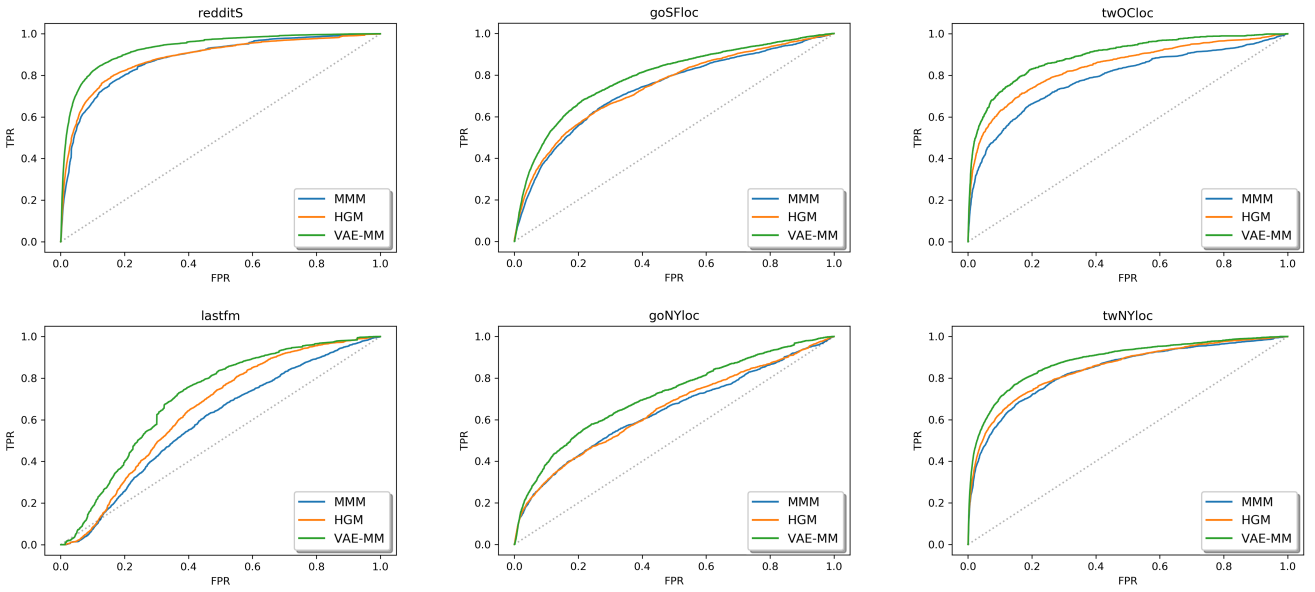


FIGURE 5. ROC curves of MMM, HGM and VAE-MM for new items across all datasets.

The deep neural network architecture of VAE could also benefit VAE-MM because it can learn accurate transformation functions. It allows us to try different settings and change the architecture or objective function conveniently to achieve a better performance.

V. CONCLUSION AND DISCUSSION

In this paper, we proposed a mixture model based on variational autoencoder (VAE) to address online behavior recommendation problem. Our Variational autoencoder mixture model (VAE-MM) considers two factors as its main components: (1) the user-specific preference (2) the latent group preference generated through VAE. While the user-specific preference reflects the behavior history of users, the latent group preference represents the hidden interests. We improved the predictive performance by applying adjustments to the VAE, such as dropout layer or skip connection. We conducted the experiment on various online behavior datasets across different domains, including location, music and community. We used Multinomial mixture model [5] and Hybrid Generative Model [7], the state-of-the-art mixture models, as the baseline for comparison. In our experiments, VAE-MM considerably outperformed the baseline models. These improvements come from the integration of VAE as a component in our mixture model.

For future work, we plan to further investigate the integration of other deep neural network models into VAE-MM. Another potential direction is incorporating additional side information to improve performance. In our experiments, we also find that some VAE techniques, such as beta annealing or aggressive training, do not improve the performance explicitly and are very data-sensitive; thus, it is necessary to further study VAE techniques for online behavior recommendation problem.

REFERENCES

[1] A. R. Benson, R. Kumar, and A. Tomkins, "Modeling user consumption sequences," in *Proc. 25th Int. Conf. World Wide Web (WWW)*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 519–529, doi: 10.1145/2872427.2883024.

[2] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii, "The dynamics of repeat consumption," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2014, pp. 419–430, doi: 10.1145/2566486.2568018.

[3] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004, doi: 10.1145/963770.963772.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. BerlinBerlin, Germany: Springer-Verlag, 2006.

[5] D. Kotzias, M. Lichman, and P. Smyth, "Predicting consumption patterns with repeated and novel events," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 371–384, Feb. 2019.

[6] M. Lichman, D. Kotzias, and P. Smyth, "Personalized location models with adaptive mixtures," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*. New York, NY, USA: ACM, 2016, p. 67.

[7] M.-D. Nguyen and Y.-S. Cho, "A hybrid generative model for online user behavior prediction," *IEEE Access*, vol. 8, pp. 3761–3771, 2020.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114

[9] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts, "Information re-retrieval: Repeat queries in Yahoo's logs," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*. New York, NY, USA: Association for Computing Machinery, 2007, pp. 151–158, doi: 10.1145/1277741.1277770.

[10] L. Lerche, D. Jannach, and M. Ludewig, "On the value of reminders within E-commerce recommendations," in *Proc. Conf. User Modeling Adaptation Personalization (UMAP)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 27–35, doi: 10.1145/2930238.2930244.

[11] W. Trouleau, A. Ashkan, W. Ding, and B. Eriksson, "Just one more: Modeling binge watching behavior," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1215–1224, doi: 10.1145/2939672.2939792.

[12] K. Kapoor, K. Subbian, J. Srivastava, and P. Schrater, "Just in time recommendations: Modeling the dynamics of boredom in activity streams," in *Proc. 8th ACM Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 233–242, doi: 10.1145/2684822.2685306.

[13] E. Adar, J. Teevan, and S. T. Dumais, "Large scale analysis of Web revisitation patterns," in *Proc. 26th Annu. CHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1197–1206, doi: 10.1145/1357054.1357241.

[14] M. A. Awad and I. Khalil, "Prediction of user's Web-browsing behavior: Application of Markov model," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 42, no. 4, pp. 1131–1142, Aug. 2012.

[15] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *Proc. ACM Conf. Ubiquitous Comput. (UbiComp)*. New York, NY, USA: ACM, 2012, pp. 911–918.

[16] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility Markov chains," in *Proc. 1st Workshop Meas., Privacy, Mobility (MPM)*. New York, NY, USA: ACM, 2012, pp. 3:1–3:6.

[17] J. Kiseleva, H. T. Lam, M. Pechenizkiy, and T. Calders, "Predicting current user intent with contextual Markov models," in *Proc. IEEE 13th Int. Conf. Data Mining Workshops*, Dec. 2013, pp. 391–398.

[18] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proc. RecSys*, Aug. 2017, pp. 152–160.

[19] G. Yang, Y. Cai, and C. K. Reddy, "Spatio-temporal check-in time prediction with recurrent neural network based survival analysis," in *Proc. 27th Int. Joint Conf. Artif. Intell.* Menlo Park, CA, USA: AAAI Press, Jul. 2018, pp. 2976–2983.

[20] T. Le, M. T. Vo, T. Kieu, E. Hwang, S. Rho, and S. W. Baik, "Multiple electric energy consumption forecasting using a cluster-based strategy for transfer learning in smart building," *Sensors*, vol. 20, no. 9, p. 2668, May 2020.

[21] A. H. Vo, L. Hoang Son, M. T. Vo, and T. Le, "A novel framework for trash classification using deep transfer learning," *IEEE Access*, vol. 7, pp. 178631–178639, 2019.

[22] M. J. Pazzani and D. Billsus, *Content-Based Recommendation Systems*. Berlin, Germany: Springer, 2007, pp. 325–341, doi: 10.1007/978-3-540-72079-9_10.

[23] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[24] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA, USA: MIT Press, 2001, pp. 556–562.

[25] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2007, pp. 1257–1264.

[26] M. R. Gupta, "Theory and use of the EM algorithm," *Found. Trends Signal Process.*, vol. 4, no. 3, pp. 223–296, Mar. 2011, doi: 10.1561/2000000034.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[28] D. J. Im, S. Ahn, R. Memisevic, and Y. Bengio, "Denoising criterion for variational auto-encoding framework," in *Proc. AAAI*, 2017, pp. 2059–2065.

[29] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn. (ICML)*, vol. 48. San Francisco, CA, USA: JMLR.org, 2016, pp. 1727–1736.

[30] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," in *Proc. ICLR*, 2017, pp. 1–12.

[31] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. World Wide Web Conf. World Wide Web (WWW)*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 689–698, doi: 10.1145/3178876.3186150.

[32] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko, "RecVAE: A new variational autoencoder for Top-N recommendations with implicit feedback," in *Proc. 13th Int. Conf. Web Search Data Mining*. New York, NY, USA: Association for Computing Machinery, Jan. 2020, pp. 528–536, doi: 10.1145/3336191.3371831.

[33] D. Kim and B. Suh, "Enhancing VAEs for collaborative filtering: Flexible priors & gating mechanisms," in *Proc. 13th ACM Conf. Recommender Syst.* New York, NY, USA: Association for Computing Machinery, Sep. 2019, pp. 403–407, doi: 10.1145/3298689.3347015.

[34] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for Top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 153–162, doi: 10.1145/2835776.2835837.

[35] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*. Toulon, France: OpenReview.net, Apr. 2017, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[36] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," 2019, *arXiv:1901.05534*. [Online]. Available: http://arxiv.org/abs/1901.05534

[37] C. K. Sønderby, T. Raiko, L. E. Maaløe, S. R. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 3738–3746. [Online]. Available: http://papers.nips.cc/paper/6275-ladder-variational-autoencoders.pdf

[38] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei, "Avoiding latent variable collapse with generative skip models," in *Proc. AISTATS*, 2019, pp. 2397–2405.

[39] J. M. Tomczak and M. Welling, "VAE with a VampPrior," in *Proc. AISTATS*, 2018, pp. 1214–1223.

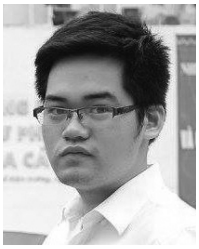
[40] S. Burkhardt and S. Kramer, "Decoupling sparsity and smoothness in the Dirichlet variational autoencoder topic model," *J. Mach. Learn. Res.*, vol. 20, pp. 1–27, 2019.

[41] Y. Miao, E. Grefenstette, and P. Blunsom, "Discovering discrete latent topics with neural variational inference," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70. San Francisco, CA, USA: JMLR.org, 2017, pp. 2410–2419.

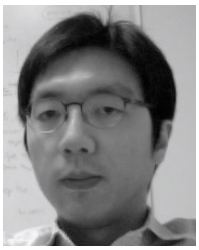
[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[43] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin, "Cyclical annealing schedule: A simple approach to mitigating KL vanishing," in *Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (Long and Short Papers)*, vol. 1. Minneapolis, MN, USA: Association for Computational Linguistics, Jun. 2019, pp. 240–250. [Online]. Available: https://www.aclweb.org/anthology/N19-1021

[44] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.



MINH-DUC NGUYEN (Member, IEEE) received the B.Sc. degree in information system from the University of Engineering and Technology, Vietnam National University, Hanoi, in 2016. He is currently pursuing the M.S. degree with Sejong University, South Korea. From 2016 to 2018, he was a Researcher Assistant and a Lecturer with the University of Engineering and Technology, Vietnam National University. His research interests include data science and social media mining.



YOON-SIK CHO (Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, South Korea, in 2003, and the Ph.D. degree in electrical engineering from the University of Southern California, USA, in 2014. He was an Academic Mentor for the RIPS Program at the Institute for Pure and Applied Mathematics, University of California at Los Angeles, and a Postdoctoral Scholar with the Information Sciences Institute, University of Southern California. He joined Sejong University, South Korea, where he is currently an Assistant Professor with the Department of Data Science. His research interests include large-scale data science, social network analysis, and cloud computing.