Variational User Modeling with Slow and Fast Features

Ghazal Fazelnia
ghazalf@spotify.com
Spotify

Eric Simon
eric.simon324@gmail.com
Spotify

Ian Anderson
iananderson@spotify.com
Spotify

Benjamin Carterette
benjaminc@spotify.com
Spotify

Mounia Lalmas
mounial@acm.org
Spotify

ABSTRACT

Recommender systems play a key role in helping users find their favorite music to play among an often extremely large catalog of items on online streaming services. To correctly identify users' interests, recommendation algorithms rely on past user behavior and feedback to aim at learning users' preferences through the logged interactions. User modeling is a fundamental part of this large-scale system as it enables the model to learn an optimal representation for each user. For instance, in music recommendation, the focus of this paper, users' interests at any time is shaped by their general preferences for music as well as their recent or momentary interests in a particular type of music. In this paper, we present a novel approach for learning user representation based on general and slow-changing user interests as well as fast-moving current preferences. We propose a variational autoencoder-based model that takes fast and slow-moving features and learns an optimal user representation. Our model, which we call **FS-VAE**, consists of sequential and non-sequential encoders to capture patterns in user-item interactions and learn users' representations. We evaluate FS-VAE on a real-world music streaming dataset. Our experimental results show a clear improvement in learning optimal representations compared to state-of-the-art baselines on the next item recommendation task. We also demonstrate how each of the model components, slow input feature, and fast ones play a role in achieving the best results in next item prediction and learning users' representations.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Machine learning approaches**; **Learning latent representations**;

KEYWORDS

User Modeling, Latent Representation, Music Streaming

ACM Reference Format:
Ghazal Fazelnia, Eric Simon, Ian Anderson, Benjamin Carterette, and Mounia Lalmas. 2022. Variational User Modeling with Slow and Fast Features. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498477

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498477

1 INTRODUCTION

Modern recommender systems have enabled users to navigate through extremely large inventories of items in online platforms and find their preferred content easily. To enable such a service, recommender systems need to successfully model users based on their history of interactions. They do so by collecting users feedback and behavior to reach an understanding about users' interests and tastes [2–4, 20, 23, 50]. User modeling is a fundamental building block in many recommendation algorithms, as it provides the information needed to serve individuals with the best content at any given moment [3, 27].

Recommendation systems are the core engines in many user-facing services such as online retail, news, and entertainment. In this work, we focus on music streaming services, which have become increasingly dominant in the past decade. Online music platforms can host tens of millions of tracks and serve hundreds of millions of users in real-time. Users consume music in a way that is distinct from other services [10, 39–41, 43]: songs are often short and played together sequentially; their popularity could be impacted by timely factors and seasonality effects; listening is shaped not just by general tastes but transient preferences (e.g. different times of day, discovering new content) or in conjunction with other tasks (e.g. commuting to work, doing chores, or dancing). These factors greatly contribute to the unique challenges that arise in designing successful recommendation systems for online music streaming. Hence, despite the recent advancement in user modeling for recommender systems, modeling users' music preferences and learning effective representations remains a challenging task.

In this work, we seek to take these distinctive characteristics and use them in building and improving upon modeling users interests and tastes. In particular, we aim at modeling users' both holistically and locally according to various types of information that they provide us. We account for general and historical user preferences, i.e. *slow features*, as well as their recent interests and feedback, i.e. *fast features*. By collecting and modeling these two types of features, we aim at learning users' long-term general tastes as well as recent or current preferences. This enables us to incorporate historical streaming while keeping the representation up to date with learning from recent interactions. Our modeling design and learning algorithm are built such that we can scale it to a large set of users and their past and new interactions.

We present a two-part model. The first part focuses on historical and slow-changing features that represent overall users' tastes and

preferences in music, whereas the second part represents recent consumption and interaction behavior from fast-changing features to account for current interests. We build the network architecture with both components and train together with an end-to-end learning algorithm. We propose probabilistic encoder components for each part inside a larger variational autoencoder framework [21, 36]. We then represent the user with the output of the learned encoding stage. Probabilistic approaches have shown promising performances in user modeling. They can be used to learn a generative model for representing users rather than a point estimate. This enables the system to model various aspects in tastes and interests for each individual, encompass uncertainty in representation, and generalize better to the whole population.

We illustrate our results on a real-world music streaming dataset and demonstrate how each model component contributes to achieving the best result in modeling users. Essentially we elaborate how slow and fast features play a role in the representation, and how model design and learning algorithm can capture holistic user interests as well as current preferences. In particular, we summarize our contributions in answering the following research questions.

Our contributions aim to answer the following research questions:

- **[RQ1]** Can user modeling be improved by incorporating past users' interactions as slow features as well as recent users' consumption as fast features?
- **[RQ2]** How do our learned user embedding vectors compare to other baselines? What can we learn from our learned embedding space?
- **[RQ3]** How much do the slow and fast feature components impact user representation?

We aim at investigating these research questions throughout our modeling process, training, and evaluation. We develop a model to effectively capture patterns in user interactions and embed that into their representation. We split the input features based on the pace with which they change, *slow* and *fast* features. We design the model such that it treats each feature set differently to learn user general and momentary preferences efficiently and combine them into an embedding vector. Finally, we show the advantages of our method over the baselines in the experiments section.

We summarize our main contributions as follow:

- We define slow and fast features to capture general users' preferences as well as their instantaneous tastes.
- We model slow and fast features non-sequentially and sequentially, respectively. This strategy allows for capturing distinctive information in both of these features, adjusting the model according to the nature of input features. It further enables us to efficiently incorporate a large set of input features and expand the available user information for the model without compromising the time complexity.
- We develop the model for user representations in a probabilistic framework to learn the generative process. This would help with us having a function learned through the inference process for representation rather than point estimates. Variational inference enables us to capture non-linearities in highly complex user behavior and understand the patterns.
- We design an end-to-end training for the Variational inference with and evaluate the model on a recommendation task as a proxy and compare to the state-of-the-art baselines.

2 RELATED WORKS

Recommender systems have been ubiquitous to many user-facing platforms over the past two decades. Originally, they were built by collecting implicit and explicit user feedback to make desirable personalized suggestions [2–4, 23]. Collaborative filtering is among the most successful approaches for recommendation. The assumption is that users who consume many similar items would have similar tastes and interests. Hence, matrix factorization approaches have been successful in modeling users based on their interactions with items. Recent advances in probabilistic models, variational approaches, and deep factorization models have shown promising results regarding the recommendation tasks [14, 19, 22, 28, 30, 38, 49, 51]. Mainly, they jointly learn user and item representations through optimizations for specific recommendation tasks.

Depending on other sources of information available beyond implicit or explicit feedback, recent works in recommender systems can be further categorized: content-based recommender systems rely on item-specific features and information [1]. Hybrid recommender systems combine the collaborative filtering approach with the content based modeling [9, 24]. Context-aware recommenders utilize the available contextual information for users and items to provide the best contextual recommendation [34].

More recently, deep learning approaches have greatly improved the recommendation results on various personalization tasks [50]. Session-based models have aimed at learning users' intent within a session [44]. They focus on capturing instantaneous users' interests to provide them with the best timely item match [16, 33]. Sequential recommender systems exploit the sequential nature of user-item interactions [12, 33, 44, 45]. Recurrent neural networks have been widely used to model such properties and have shown great results in various next-item recommendation settings [5, 11, 25, 35, 46]. The latter approaches are scalable and can account for a long time horizon of user interaction and consumption in learning representations. However, they lack precision when it comes to picking up on current users' tastes and timely interests. The former are great at modeling recent trends in user interests, but have limitations on large-scale datasets as it is hard to scale them up to include various features over long sequences.

Our model is based on a hybrid recurrent neural network architecture that takes in recent interaction as sequential "fast" features and aggregates over historical interaction as non-sequential "slow" features. Our framework analyzes these features separately in the initial stage. This mixed approach allows us to capture all-time user interests through slow features yet account for recent fast sequential interactions to capture current users' tastes. In addition, we could utilize a wider range of input features without jeopardizing the time complexity. We aggregate features that represent general users' tastes and treat recent interactions sequentially to learn current users' tastes.

Turning to prediction task, next-item prediction has been a widely-used choice in sequential or session-based settings [44].

However, with a very large item pool, predicting the next item could become intractable. Predicting the embedding of the next item rather than the item itself has been introduced to resolve this issue [25]. To this end, we mainly focus on learning user representation based on their fast and slow-changing features. We set the embedding of the next item to be our target within our learning process.

Variational-based approaches benefit from the non-linear probabilistic capacities which allow them to go beyond traditional linear factorization routes [28]. Probabilistic latent factor models exploit the underlying similarities among items that are consumed together [31, 32, 47]. Consequently, user representations are based on the representations of the items that they have interacted with in the past. Variational autoencoders enable us to infer the latent patterns and themes among large-scale user-item interaction datasets powered by neural networks [15, 48]. To capture complex structure in user behavior, we design a new variational autoencoder with two main components to learn from fast and slow features separately. The autoencoder is trained end-to-end to learn from both slow and fast feature sets simultaneously.

Music recommendation has been a challenging area of research in the past decade [7, 10, 40, 41]. While the classic recommendation models could be applied to the music domain, in practice many of them are not feasible due to the distinct characteristics of this domain. First, the consumption intents that tracks are listened to highly differ from other recommender systems as they are often sequential and context-dependent. Second, track lengths are shorter than movies yet higher than the time spent with or browsing an item in e-commerce. Last, users have general musical preferences which change slowly over time, yet they could have a short-time interest in a particular musical content. By incorporating both slow and fast features, our model is able to better represent these factors that make music recommendation distinct.

3 METHOD

In this section, first, we introduce and formalize our notation. Second, we introduce the various components of the model and expand them. Last, we present the model architecture and outline on the generative and inference processes.

3.1 Notations

We index users with $\{1, 2, \ldots, U\}$ and music tracks with $\{1, 2, \ldots, N\}$. For each user u at time t, we denote the sequential features as $x_u^s(t) \in \mathbf{R}^{d_s \times w}$ and non-sequential ones as $x_u^n(t) \in \mathbf{R}^{d_n}$ where d_s, w, and d_n represent the dimension of sequential features, sequence length, and the dimension of non-sequential features, respectively. We define the variable $z_u(t) \in \mathbf{R}^K$ to represent user u at time t. Finally, we use $x_u(t) = [x_u^s(t), x_u^n(t)]$ to represent the sequential and non-sequential features corresponding to u at time t. Figure 1 illustrates the input features to the model. The model takes in the recent interaction as fast features sequentially, $x_u^s(t)$. We aggregate over historical interactions, $x_u^n(t)$, and set that as the slow features for the model.

Without loss of generality and to clarify this further, we assume we have a music streaming dataset consisting of users and tracks. This dataset has a record of tracks played by users as well as the timestamp of the play. These features correspond to the fast input features that the model would take in as sequences at time t, $x_u^s(t)$. This dataset also contains aggregate features, such as total plays, total likes, skips, and restarts up to time t. These features, which are collected over a long period of time, are denoted by $x_u^n(t)$.

3.2 Model Components

To learn and train an end-to-end model with slow and fast feature, we design a *two-component* variational autoencoder to learn latent variables corresponding to each of those input components. We provide a brief overview of the specific part of the model next.

3.2.1 Recurrent Neural Networks (RNNs). The simplest recurrent neural network models a sequence of variables $y = (y_1, ..., y_T)$ with a neural network that takes as input a corresponding sequence of variables $x = (x_1, ..., y_T)$ along with internal hidden states from the network $h = (h_1, ..., h_T)$. When the model is probabilistic, this can be viewed as a joint likelihood of the data, $p(y|\theta) = p(y_1|\theta)\Pi_{t=2}^T p(y_t|y_{t-1}, \theta)$, where $p(y_t|y_{t-1}, \theta) = p(y_t|h_t)$ and $h_t = f_\theta(x_t, h_{t-1})$ for $t > 0$, often with $x_t \equiv y_{t-1}$ in the recurrent setting. The non-linear function f_θ can be a recurrent cell in a neural network. When multiple sequences are observed, a straightforward approach is to treat them as independent from the same RNN [13].

3.2.2 Variational Autoencoders. Generative modeling is a broad area of research that seeks to learn the distribution of data points. To optimally approximate the data generating process, modeling assumptions are put in place regarding the likelihood of observations given the model parameters. Model parameters are probabilistic variables drawn from a defined prior distribution. Expressive and informative prior distributions help to identify and model complex underlying structures in the data [21, 36].

Inference algorithms aim at estimating posterior distributions to complete the learning process. However, most large and high dimensional datasets need complex generative models for which the exact inference task becomes analytically implausible. Approximate inference techniques such as Markov Chain Monte Carlo (MCMC) and variational inference have been widely used to perform the approximation task [37]. While they have been very successful in a variety of applications, they remain un-scalable for large datasets. Variational autoencoders on the other hand aim at tackling this issue by amortization assumption and batch inference. Given a dataset X, the generative model is defined as $x \sim p(X|z)$ and $z \sim p(z)$. Calculating $p(z|x)$ might be impossible due to an intractable integral in the denominator of $p(z|x) = p(x|z)p(z)/p(x)$.

Variational inference approximates the true distribution with an estimation $q(z)$ where q is usually selected from a family of distribution and the optimization seeks to find a member that is the closest to the true posterior distribution [6]. KL-divergence is a standard metric to evaluate the approximate posterior. Since the posterior is unknown, calculating the divergence value remains challenging. However, a simple re-writing of this KL-divergence would result in:

$$\mathrm{KL}(q_\theta(z|x)||p(z)) = \log(p(X)) + \mathbf{E}_q[\log(q_\theta(z|x))] - \mathbf{E}_q[\log(p(x, z))]$$

Since KL-divergence is always non-negative, and $\log(p(X))$ is constant with respect to posterior parameters, taking the gradients to minimize the KL would be equivalent to taking the gradient

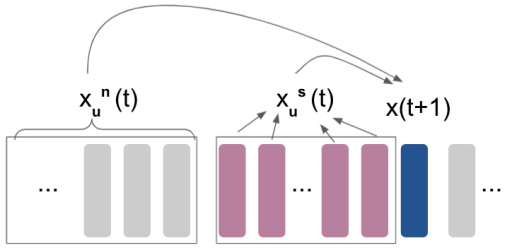

Figure 1: Input features to the model. The model takes in recent and historical interactions as fast and slow features in sequential and non-sequential order.

to maximize the rest of the terms of the equation. Therefore, in order to minimize the unknown KL term, one can maximize the Evidence Lower Bound (ELBO). The ELBO for one single data point in variational autoencoder would be:

$$\mathcal{L}_i(\theta) = \mathbf{E}_{q_\theta(z|x_i)}[\log(p(x_i|z))] - \mathrm{KL}(q_\theta(z|x_i)||p(z))$$

If we parametrize the generative process with ϕ, then:

$$\mathcal{L}_i(\theta, \phi) = \mathbf{E}_{q_\theta}[\log(p_\phi(x_i|z))] - \mathrm{KL}(q_\theta(z|x_i)||p(z)) \quad (1)$$

The final optimization target would be to maximize:

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \mathcal{L}_i(\theta, \phi)$$

This process casts the inference as an optimization problem with respect to the posterior parameters.

3.3 Model Architecture

With the building blocks of the last section, we now present the main model and outline the generative and inference processes.

3.3.1 Generation. In the generative process, we define the decoding process using the encoded state $z_u(t)$ with sequential and non-sequential components. The latent parameters are a function of a recurrent state and a non-recurrent state corresponding to the encodings for fast state and slow state features, respectively. g could be a linear concatenation function.

$$z_u(t) = g(z_u^s(t), z_u^n(t))$$

The generative process for user u at time t is:

$$\begin{aligned} x_u^s(t) &\sim \mathcal{N}(\mu_x^s(z_u^s(t)), \sigma_x^s(z_u^s(t))), \\ \mu_x^s(z_u^s(t)), \sigma^s(z_u^s(t)) &= f_\phi^{dec,s}(z_u^s(t), h_{t-1}) \\ x_u^n(t) &\sim \mathcal{N}(\mu_x^n(z_u^n(t)), \sigma_x^n(z_u^n(t))), \\ \mu_x^n(z_u^n(t)), \sigma^n(z_u^n(t)) &= f_\phi^{dec,n}(z_u^n(t)) \end{aligned} \quad (2)$$

where the prior distributions are as follows:

$$z_u^s(t) \sim \mathcal{N}(0, \mathbf{I}_K), z_u^n(t) \sim \mathcal{N}(0, \mathbf{I}_K) \quad (3)$$

Then, the joint probability distribution and the posterior approximate can be found as:

$$\begin{aligned} p(x_u(1:T), z_u(1:T)) &= \Pi_t p(x_u(t)|z_u(t))p(z_u(t)) \\ q(z_u(1:T)|x_u(1:T)) &= \Pi_t q(z_u(t)|x_u(1:t), z_u(1:t-1)) \\ h_t &= f(x_u(t), z_u(t), h_{t-1}) \end{aligned} \quad (4)$$

3.3.2 Inference. In the encoder, we define the probability distributions as follow:

$$\begin{aligned} z_u(t)|x_u(t) &\sim \mathcal{N}(\mu_z(x_u(t)), \sigma_z(x_u(t))) \\ \mu_z(x_u(t)), \sigma_z(x_u(t)) &= f^{enc}(x_u(t), h_{t-1}) \end{aligned} \quad (5)$$

The goal is to maximize the variational lower bound. For the posterior inference, we can compute the Evidence Lower Bound (ELBO) as follows:

$$\begin{aligned} \mathbf{E}_{q(z_{1:T}|x_{1:T})} = [&\sum_{t=1}^{T} \log(p(x_t|x_{1:t-1}, z_{1:t})) \\ &- \mathrm{KL}(q(z_t|x_{1:t}, z_{1:t-1})||p(z_t|x_{1:t-1}, z_{1:t-1}))] \end{aligned} \quad (6)$$

$$\begin{aligned} &\mathcal{L}(\phi, \theta, X) = \\ &\sum_u \sum_{t=1}^{N_u} \{\frac{1}{2} \sum_k (\sigma_k)(t) - 1 - \log(\sigma_k))(t) + \mu_k(t)^2) - \\ &\mathbf{E}_{\varepsilon \sim \mathcal{N}(0,1)}[\log(p_\phi(x_u(t)|z(\varepsilon, t)))]\} \end{aligned} \quad (7)$$

The main model architecture is depicted in Figure 2. Features are split into a fast and slow set and are processed by separate encoders. Sequential encoder consists of two LSTM cells with LeakyReLU activation function. Regular encoder is a two-layer feedforward network with a LeakyReLU activation. Next, they pass through another cell to produce the user representation. The decoder is a two-layer cell. We use dropout layers in the encoder and decoder to avoid overfitting and generalize better.

Reparametrization Trick. The evidence lower bound in Equation 6 is known to have challenges with respect to the gradients for optimizing ϕ in the sampling process. The *reparametrization trick* [21, 36] resolves this issue; it first samples $\varepsilon \sim \mathcal{N}(0, \mathbf{I}_K)$ and reparametrizes $z_u(t) = \mu_z(x_u(t)) + \varepsilon\sigma_z(x_u(t))$. With this process, the stochasticity in sampling is with regard to ε which makes back-propagation for optimizing ϕ plausible.

Modified VAE. The ELBO derived in Equation 1 can be interpreted as the sum of two terms. The first term, $\mathbf{E}_{q_\theta(z|x_i)}[\log(p(x_i|z)])$, is the negative reconstruction error, and the KL could be viewed as a regularization term. The role of reconstruction is to maximize the likelihood of data under the posterior approximation distribution while the KL term penalizes the posterior distributions that are far from the prior. Viewing the ELBO as a trade-off between these two terms has inspired previous works to achieve more powerful inference processes by changing the regularization weight [17, 18, 28]. In particular, β-VAE is defined as:

$$\mathcal{L}(\theta, \phi) = \mathbf{E}_{q_\theta}[\log(p_\phi(x|z))] - \beta \cdot \mathrm{KL}(q_\theta(z|x)||p(z))$$

Here, $\beta = 1$ results in the original ELBO objective. $\beta > 1$ promotes learning an efficient representation of data and pushes the regularization further. However, if we let $0 < \beta < 1$, we can weaken the influence of the KL, and as a result the prior distribution [28]. This further helps to avoid a variational collapse. Although this might lower the model's ability to generate novel user representations, it improves the reconstruction error. This enhances the performance especially on recommendation tasks, which often is the ultimate goal. To avoid exploding gradients, we set the initial ε to zero.

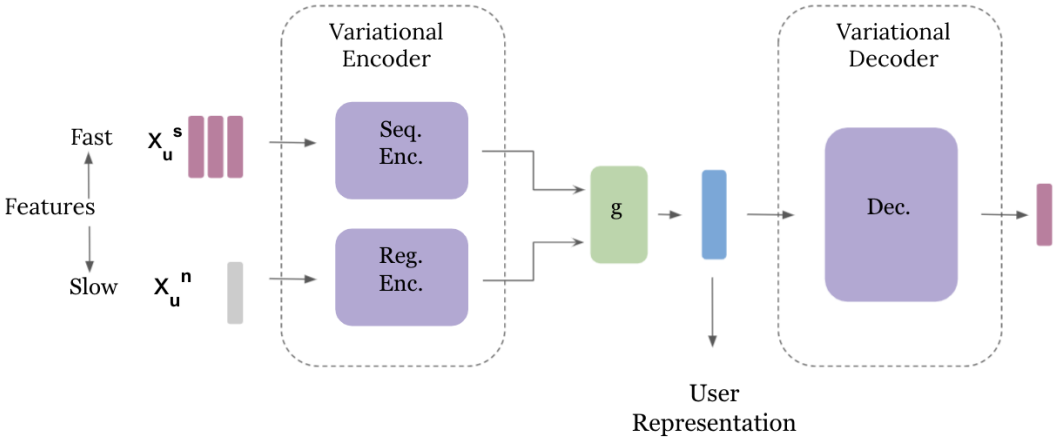

Figure 2: The overall architecture of our model, FS-VAE. Fast and slow input features are taken by the variational encoder separately. They merge after the encoders to generate the user representation before passing through a variational decoder. The decoder output is shown in concatenation.

3.4 Training Process

In Algorithm 1 we summarize the training steps using an stochastic optimization.

Algorithm 1 Training VAE for Optimizing the ELBO

Input: Matrix X of all users histories.
Initialize θ and ϕ randomly.
while not converged **do**
  Sample a batch of users $\mathcal{D}$
  For $u \in \mathcal{D}$ select x_u for all its time steps.
    Generate $\varepsilon \sim \mathcal{N}(0, \mathbf{I}_K)$
    Compute z_u for all time steps.
    Compute the gradients ∇L_θ and ∇L_ϕ.
  Compute the aggregate gradient from this batch
  Update θ and ϕ by taking the gradient step.
end while
Output: θ and ϕ

4 EXPERIMENTS

We present the experimental results performed on a real-world music dataset. We start with introducing the dataset.

4.1 Dataset

We evaluate our approach on a real-world music streaming dataset consisting of a sample of 150,000 users over a 28-day period. As we focus on learning user representation, we let each track be represented by an 80-dimensional real-valued vector acquired in a pre-processing step. These track representations are based on track co-occurrences in playlists meaning that two tracks are likely to be near each other in the embedding space if they co-occur in playlists and vice-versa. This particular embedding space has been previously shown to work well for music recommendation [29]. The same track representations are used for all the experiments.

This dataset contains the times of each listening event and aggregated users' interactions with tracks (if any). These are *i)* like, which is the total number of likes, *ii)* add to a playlist, which corresponds to the total number of tracks added to their personal playlists, *iii)* skip, which is the total number of skipped tracks, and finally *iv)* restart, which is the total number of restarts.

4.2 Exploratory Analysis

In order to gather insights for our research questions and support modeling decisions, we first present an exploratory analysis of the dataset.

In particular, we investigate the slow-moving features and explore their properties. Figure 3 shows the portion of plays in different parts of a day. As seen here, the majority of the listening events happen during the afternoon and evening hours. As previous works have noted [16, 26, 42], the time of the day could indicate users' interests, tastes, and budget in interacting with an app or listening to music. Therefore, it has a potentially significant impact on users' streaming content. As will be shown later in the experiments, including time of day as an input feature improves accuracy when predicting users' next plays.

Figure 4 shows the average portions of skips by users' throughout the day. This plot indicates the difference in skip rate which could be influenced by users' mood, engagement, satisfaction, or interest. Incorporating skips as input feature could further help the model to learn users' preferences and improve user representation [8, 16].

To better understand our slow input features, we illustrate their pairwise correlation in Figure 5. Most features tend to have a low correlation (indicated by small positive values), and no pairs seem to be negatively correlated. As expected, total plays and plays from the liked tracks have the highest pairwise correlation, which highlights the fact that users prefer to turn to their liked tracks when listening to music. The high pairwise correlation between total plays and skips is an interesting insight. Contrary to what might seem the direct effect of skipping tracks, they are correlated with

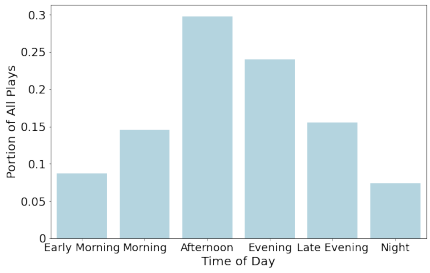

Figure 3: The portion of all tracks that are played by users over different parts of a day. Afternoon and Evening plays overall account for more than half of the plays.

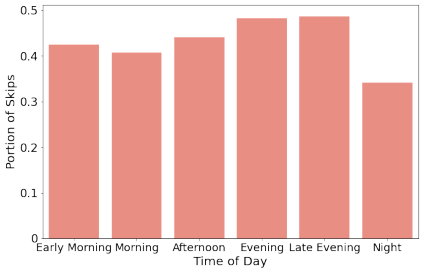

Figure 4: The portion of skips when listening to a track by users over different parts of a day. Time of a day could affect user interaction as well as engagement with the streaming track. Users tend to skip more tracks on average later in a day compared to other dayparts.

more plays overall. Skips could indicate user engagement with the streaming music and help to understand their momentary preferences. They emphasize that although the user is not interested in the current track, they would like to stay and continue listening. Similarly, restart, skips, and total plays all have high pairwise correlation which suggests that like skips, restarts indicate high user engagement and interest in continuing to listen [16]. Without having to explicitly interpret the purpose of these behavioral actions, we let the model learn patterns in users' listening history, capture users' interests, and represent them in their embeddings. We split the data into two groups. As shown in Figure 1 we aggregate over historical interactions and set them as slow features, while keeping the recent data as sequences as fast features. The model is designed to capture patterns from this data and learn a comprehensive user embedding at any time step.

4.3 Experimental Set-up

We now present the experimental results for training the model and learning user representations. Our main task is the following: given a user's listening history as a sequence of tracks along with their interaction, how well can we learn the user's representation? To measure the performance, as mentioned in Section 3, we train the model to predict the next track played by the user. A shorter

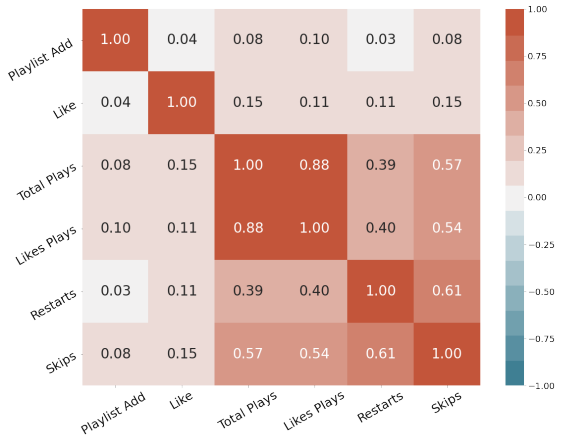

Figure 5: Correlation between slow input features. Higher values indicate a higher correlation between the features. Although most features have low pairwise correlation, some could be highly correlated, such as total plays and plays from the liked tracks.

distance in track space would be our proxy to evaluate the learned user representation.

After the training step, we evaluate the model for prediction error (L2 loss) on a held-out test dataset. In addition to that, we measure the cosine distance between the predicted track and the ground truth. Lastly, we perform an ablation study to understand the role of each part of the model in the performance. In particular, we investigate how the sequential input, sequential model architecture, and the generative process contribute to the results.

4.3.1 Training Setup. Our model is trained to maximize the ELBO as it learns the user representation in the encoding step on predicting the next played tracks by users. FS-VAE parameters are: $\{d_s = 82, w = 5, d_n = 6, K = 80\}$. The output target is an 80-dimensional next track prediction. We train the model using Adam optimizer with learning rate = 10^{-5} for 100 epochs. We let $\beta = 0.5$ to allow to weaken the influence of the prior distribution. These hyperparameters are tuned using {70%, 20%, 10%} split for training, validating, and testing and correspond to the best performing results.

4.3.2 Baselines. To evaluate the performance of our model FS-VAE, we compare with the following baselines:

- *Vanilla RNN:* This baseline takes in the sequential features and trains an RNN model to predict the next track played by a user.
- *Aggregate Sequence:* For this baseline, we aggregate the sequential features and feed them along with the slow features to the model.
- *Popularity:* We use popularity to recommend the next tracks to the users.

- *Average:* Since we are training and predicting in the track embedding space, this baseline takes the average of past tacks listened to by a user as the next track prediction.
- *LatentCross:* This RNN-based model combines the sequential features with contextual features into a deep architecture for a recommender system [5]. We modified the objective to match our ELBO loss and replaced the softmax with a feed forward cell. We fix the embeddings for tracks, and only let it train for the user embeddings.
- *JODIE:* This model learns the embeddings from sequential and contextual features [25]. We aggregated over skips to match our setting, and we fixed the track embeddings, similar to other baselines.
- *VAE Collaborative Filtering:* This is a collaborative filtering baseline that trains a variational autoencoder for recommendation tasks [28]. In order to match our settings, we set the track embeddings fixed and only trained the user matrix. We performed a grid search to pick the best β value for this experiment.

4.4 Results

In this section, we present the experimental results to evaluate our approach and answer our research questions.

4.4.1 Evaluation on Next Track Prediction. To answer our first research question **RQ1**, we evaluate our approach on the next track prediction task. We show the results in Table 1. We run each experiment 10 times and show the average results alongside the standard deviation in each row and corresponding column. L2-norm distance (L2 Distance) shows the Euclidean distance in the embedding space for the next track prediction and cosine distance shows one minus the cosine similarity in that space.

Our model FS-VAE results in significant improvement in prediction distances among the candidates. JODIE, LatentCross, and VAE-CF are the top competitors but result in significantly higher L2 distances and noticeable inferior performance cosine distances. This suggests that our model could predict the embedding of the next track played by the user with higher accuracy with lower Euclidean distance. As regions in track embedding space usually correspond to distinct music genres or types, prediction in the vicinity of the target becomes very crucial. This would be essential in an online setting where the recommender system generates recommendations for users and there is no ground truth. Our model could estimate the neighborhood in the track space that the user would be interested in, and could suggest them music tracks in that region.

4.4.2 Analyzing User Embedding Space. To further analyze FS-VAE qualitatively and investigate the user embedding space and answer our second research question **RQ2**, we monitor the learned vectors after few updates. In particular, we are interested to know how much movement our model creates in the user embedding space after several updates and how much that differs from an average baseline. Moreover, our goal is to analyze the embeddings produced by the model to collect qualitative insight on what might be contributing to the improved prediction results that the model generates.

Table 1: Prediction results on a held-out test dataset for comparing our model FS-VAE to current baselines. Prediction errors as L2 Distance and Cosine distance show the average L2 distance and cosine distance between the predicted track and the truth, respectively.

Model	L2 Distance	Cosine Distance
Vanilla RNN	7.04 ± 0.11	0.31 ± 0.027
Aggregate Sequence	10.29 ± 0.20	0.33 ± 0.025
Average	10.86 ± 0.37	0.41 ± 0.032
Popularity	8.17 ± 0.42	0.56 ± 0.046
LatentCross	4.53 ± 0.18	0.27 ± 0.023
JODIE	3.56 ± 0.13	0.27 ± 0.016
VAE-CF	4.22 ± 0.15	0.28 ± 0.018
FS-VAE (Our Model)	**3.18 ± 0.19**	**0.26 ± 0.012**

To study this, we plot the average changes in user embeddings by finding L2 distance as well as cosine distance after one iteration and seven iterations of updates from the starting embedding. Figure 6 shows the distribution of distance values in 80-dimensional user embedding space after one time step. The top plots correspond to the results for the Average baseline, in which user embeddings are calculated as an average of the track they listened to. The bottom plots show the results of our model. On average, our model shows more movement in the user embedding space which could be contributing to adjusting to users' interests sequentially.

Figure 7 takes the time steps forward and shows the movement in users' embeddings after seven time steps. Similar to the previous figure, top and bottom plots correspond to Average baseline and our model, respectively. There seem to be more movements in the embeddings space on average for our model compared to the baseline. This suggests that our model is more dynamic in that it captures users' instantaneous tastes signaled by fast features as well as general their general interests.

4.5 Ablation Study

4.5.1 Fast features. To investigate the fast features, we experiment with different variations of the model. First, we remove these features (No Fast Features row) which results in a significant drop in prediction results. Second, we kept those features, but randomly shuffled each sequence's items in training (Shuffle Sequence row). This could be an indicator of whether there is any valuable information in the order with which tracks appear in sequences consumed by users. As noted in the table, we can observe that the prediction errors increase certifying that there are crucial patterns in each sequence order that the model is learning from.

4.5.2 Slow features. To evaluate the impact of slow features, we run an experiment in which we removed all slow features and allowed the model to only train with the fast features (No Slow Features row). As can be seen, the performance worsens. One analogy that we could make is to a session-based model. When we remove the slow feature model, our approach could resemble a session-based recommender. The drop in performance suggests that slow-moving features can help the model understand the user perhaps by signaling to slow-moving user tastes. Moreover, slow

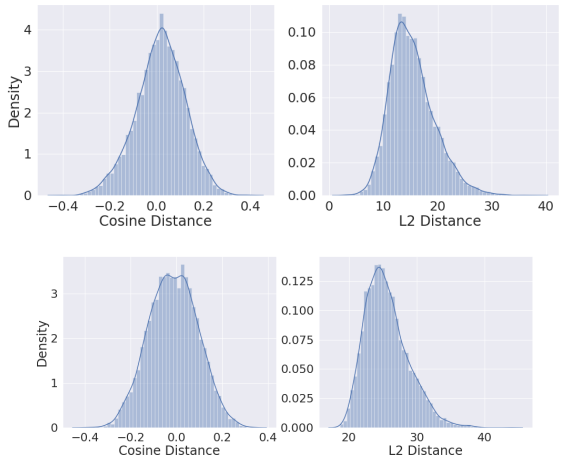

Figure 6: Average changes in user representations after one time step. The top plots correspond to the average baseline model while the bottom plots show the result for our proposed model. The left and right hand side plots show the Cosine distance and L2 distance, respectively.

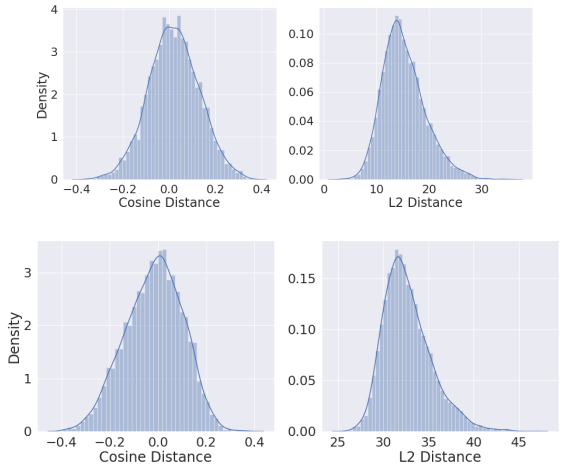

Figure 7: Average changes in user representations in the embedding space after seven time steps. The top plots correspond to the average baseline model while the bottom plots show the result for our proposed model. The left and right hand side plots show the Cosine distance and L2 distance, respectively.

features are still dynamic but move at a much slower pace than fast features. This enables the model to capture any new user interest and modify the representation accordingly. We view our approach as a dynamic user representation with a global-local strategy, global being the slow-moving users' tastes and local being their instantaneous preferences. If any current fast tastes continue to appear, it will be captured by the slow part as well eventually. It should be noted that the drop in prediction errors is less than removing

Table 2: Prediction results on a held-out test dataset for an ablation study of the model. Prediction errors as L2 Distance and Cosine distance show the average L2 distance and cosine distance between the predicted track and the truth, respectively.

Model	L2 Distance	Cosine Distance
No Fast Features	12.13 ± 0.19	0.73 ± 0.044
Shuffle Sequence	7.66 ± 0.17	0.31 ± 0.042
No Slow Features	6.43 ± 0.37	0.32 ± 0.035
Non Probabilistic	5.44 ± 0.38	0.34 ± 0.065

the fast sequential feature, suggesting that it has a lower impact on learning users' preferences than fast features. This could be due to our target task being next item prediction. Our goal is to learn embeddings that represent users within the music space. Next track prediction is our proxy for approximating users' interests within that space. However, exploring other training tasks and investigating their impact on the embeddings would be an interesting future direction.

4.5.3 Probabilistic model. Finally, to evaluate the role of the probabilistic model and the Bayesian inference, we run experiments without the probabilistic network. The training loss is defined only as the reconstruction error, and variables z's are removed. The results indicate that the generative process and the variational inference are influential in capturing users' preferences and producing better recommendations.

5 CONCLUSION

We present a novel approach for learning user representations in a recommender system by learning from their past interaction with the items. Our variational autoencoder-based model distinctively process slow-moving and fast-moving input features in a non-sequential and sequential process, respectively. In particular, in music streaming applications, users' preferences at any given moment are influenced by their general interests as well as their instantaneous tastes. Our goal is to learn these factors and capture them in users' embeddings from the slow-moving and fast-moving features, respectively. We develop a variational autoencoder model with a generative and inference network. We train an end-to-end network to learn user embeddings by optimizing for the next item as a proxy prediction target. Our experimental results on a real-world music streaming dataset show clear improvements over the current baselines.

In this paper, we mainly focus on music streaming applications; however, we could potentially extend this work to other domains. We incorporated the fact that users' taste in music is influenced by their general slow-moving interests as well their instantaneous preferences. Other domains such as news, online shopping, and movie recommendation are similar in nature. Our approach could be extended to those domains with some adaptations. Future works include an extension of this work to other domains such as text, video, or e-commerce. A/B testing is another important future work to confirm the results in an online setting.

REFERENCES

[1] Charu C Aggarwal. 2016. Content-based recommender systems. In *Recommender systems*. Springer, 139–166.

[2] Xavier Amatriain and Justin Basilico. 2016. Past, present, and future of recommender systems: An industry perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 211–214.

[3] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2019. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review* 52, 1 (2019), 1–37.

[4] Robert M Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *Acm Sigkdd Explorations Newsletter* 9, 2 (2007), 75–79.

[5] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[6] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association* 112, 518 (2017), 859–877.

[7] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)* 47, 2 (2014), 1–35.

[8] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The music streaming sessions dataset. In *The World Wide Web Conference*. 2594–2600.

[9] Erion Çano and Maurizio Morisio. 2017. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis* 21, 6 (2017), 1487–1524.

[10] Toni Cebrián, Marc Planagumà, Paulo Villegas, and Xavier Amatriain. 2010. Music recommendations with temporal context awareness. In *Proceedings of the fourth ACM conference on Recommender systems*. 349–352.

[11] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[12] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[14] Prem Gopalan, Jake M Hofman, and David M Blei. 2015. Scalable Recommendation with Hierarchical Poisson Factorization.. In *UAI*. 326–335.

[15] Prem Gopalan, Francisco J Ruiz, Rajesh Ranganath, and David Blei. 2014. Bayesian nonparametric poisson factorization for recommendation systems. In *Artificial Intelligence and Statistics*. PMLR, 275–283.

[16] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and sequential user embeddings for large-scale music recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 53–62.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. (2016).

[18] Matthew D Hoffman and Matthew J Johnson. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, Vol. 1.

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

[20] Folasade Olubusola Isinkaye, YO Folajimi, and Bolande Adefowoke Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal* 16, 3 (2015), 261–273.

[21] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[22] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 447–456.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[24] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 99–106.

[25] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[26] Huoran Li, Xuan Lu, Xuanzhe Liu, Tao Xie, Kaigui Bian, Felix Xiaozhu Lin, Qiaozhu Mei, and Feng Feng. 2015. Characterizing smartphone usage patterns from millions of android users. In *Proceedings of the 2015 Internet Measurement Conference*. 459–472.

[27] Sheng Li and Handong Zhao. 2020. A Survey on Representation Learning for User Modeling.. In *IJCAI*. 4997–5003.

[28] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[29] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th acm international conference on information and knowledge management*. 2243–2251.

[30] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[31] Aanchal Mongia, Neha Jhamb, Emilie Chouzenoux, and Angshul Majumdar. 2020. Deep latent factor model for collaborative filtering. *Signal Processing* 169 (2020), 107366.

[32] James Neve and Ivan Palomares. 2019. Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 219–227.

[33] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[34] Shaina Raza and Chen Ding. 2019. Progress in context-aware recommender systems—An overview. *Computer Science Review* 31 (2019), 84–97.

[35] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten De Rijke. 2019. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4806–4813.

[36] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*. PMLR, 1278–1286.

[37] Christian Robert and George Casella. 2013. *Monte Carlo statistical methods*. Springer Science & Business Media.

[38] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 600–608.

[39] Markus Schedl. 2019. Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics* 5 (2019), 44.

[40] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval* 7, 2 (2018), 95–116.

[41] Andreu Vall, Massimo Quadrana, Markus Schedl, Gerhard Widmer, and Paolo Cremonesi. 2017. The Importance of Song Context in Music Playlists.. In *RecSys Posters*.

[42] Steven Van Canneyt, Marc Bron, Andy Haines, and Mounia Lalmas. 2017. Describing patterns and disruptions in large scale mobile app usage data. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 1579–1584.

[43] Dongjing Wang, Shuiguang Deng, and Guandong Xu. 2018. Sequence-based context-aware music recommendation. *Information Retrieval Journal* 21, 2 (2018), 230–252.

[44] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[45] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830* (2019).

[46] 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*. 495–503.

[47] Di Wu, Xin Luo, Mingsheng Shang, Yi He, Guoyin Wang, and MengChu Zhou. 2019. A deep latent factor model for high-dimensional and sparse matrices in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2019).

[48] Teng Xiao and Hong Shen. 2019. Neural variational matrix factorization for collaborative filtering in recommendation systems. *Applied Intelligence* 49, 10 (2019), 3558–3569.

[49] Yingyuan Xiao, Pengqiang Ai, Ching-Hsien Hsu, Hongya Wang, and Xu Jiao. 2015. Time-ordered collaborative filtering for news recommendation. *China Communications* 12, 12 (2015), 53–62.

[50] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[51] Fan Zhou, Zijing Wen, Kunpeng Zhang, Goce Trajcevski, and Ting Zhong. 2019. Variational session-based recommendation using normalizing flows. In *The World Wide Web Conference*. 3476–3475.