2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00138

Variational Self-attention Network for Sequential Recommendation

Jing Zhao†, Pengpeng Zhao†*, Lei Zhao†, Yanchi Liu‡, Victor S. Sheng§, and Xiaofang Zhou††
†School of Computer Science and Technology, Soochow University, Suzhou, China
‡Rutgers University, New Jersey, USA
§Department of Computer Science, Texas Tech University, Lubbock, USA
††The Hong Kong University of Science and Technology, Hong Kong SAR, China
†jzhao1@stu.suda.edu.cn, {ppzhao,zhaol}@suda.edu.cn
‡yanchi.liu@rutgers.edu §victor.sheng@ttu.edu ††zxf@cse.ust.hk

***Abstract*—Sequential recommendation has become an attractive topic in recommender systems. Existing sequential recommendation methods, including the methods based on the state-of-the-art self-attention mechanism, usually employ deterministic neural networks to represent user preferences as fixed-points in the latent feature spaces. However, the fixed-point vector lacks the ability to capture the uncertainty and dynamics of user preferences that are prevalent in recommender systems. In this paper, we propose a new Variational Self-Attention Network (VSAN), which introduces a variational autoencoder (VAE) into the self-attention network to capture latent user preferences. Specifically, we represent the obtained self-attention vector as density via variational inference, whose variance well characterizes the uncertainty of user preferences. Furthermore, we employ self-attention networks to learn the inference process and generative process of VAE, which well captures long-range and local dependencies. Finally, we evaluate our proposed method VSAN with two public real-world datasets. Our experimental results show the effectiveness of our model compared to the state-of-the-art approaches.**

***Index Terms*—variational, attention, sequential recommendation.**

I. Introduction

In the age of information explosion, recommender systems are playing an increasingly significant role in our daily life. Assuming that user preferences are static, the approaches like matrix factorization [1] have achieved a great success in traditional recommendation. However, the key to an effective recommender system is accurately characterizing the users' tastes, which are full of uncertainty and evolving by nature. Sequential recommendation, which tries to capture users' dynamic preferences, has become an attractive topic in both academia and industry.

Sequential recommendation methods model user behaviors as a sequence of items instead of a set of items. In literature, various methods have been proposed to recommend the next item(s) that the user might like according to his/her historical interaction sequence. For example, Markov chain based methods [2]–[4] model the user sequential behaviors by learning the transition diagram of the items, which is used to predict the next item(s) that the user might be interested in [5]. Factorized personalized Markov chain (FPMC) is a classic method, which linearly combines both the Markov chain and the matrix factorization model to capture user preferences [2]. However, since the weights are fixed linearly, it is not sufficient to model high-level interactions. Moreover, motivated by the success of deep learning, recurrent neural networks (RNNs) have been intensively studied and achieved success in sequential recommendation [6]–[9]. For example, Hidasi et al. [6] proposed a modified gated recurrent unit (GRU) to model session data by utilizing session-parallel mini-batches based on the user's past interactions. However, these RNN-based models, even with the advanced memory cell structures such as long short-term memory (LSTM) and GRU, are notoriously challenged on maintaining long-range dependencies due to gradient disappearing. For example, Khandelwal et al. [10] demonstrated that language models using LSTM can apply about 200 context tokens on average. However, only 50 nearby tokens can be sharply distinguished, which reveals that even LSTM has difficulty in capturing long-range dependencies. Moreover, the sequential nature of RNN needs to learn to pass useful information forward step by step [11], which makes parallelization challenging [12].

Recently, self-attention networks (SANs) have been well employed in numerous natural language processing (NLP) tasks, such as machine translation [13], sentiment analysis [14], and question answering [15]. SANs also show better performances compared to classic RNN and convolutional neural network (CNN) in sequential recommendation [16], [17]. For example, Kang et al. [16] presented the Self-Attentive Sequential Recommendation (SASRec) model to capture both long-range and local dependencies of items in a sequence, which used to be respectively modeled by RNN and CNN. Compared to RNN-based models, SASRec has fewer parameters and higher training efficiency. Moreover, SASRec can well capture long-term dependencies, because it can access any part of the history regardless of distance. However, all the above methods model users' sequential behaviors with deterministic methods, which consider a user's preference as a fixed point vector. As shown in Figure 1, we argue that it can not characterize the uncertainty of user preferences without constraints of error terms [18], [19]. Suppose user u interacted with a sequence of items i_1, i_2, i_3 and i_4, and

*Corresponding author: Pengpeng Zhao

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00138

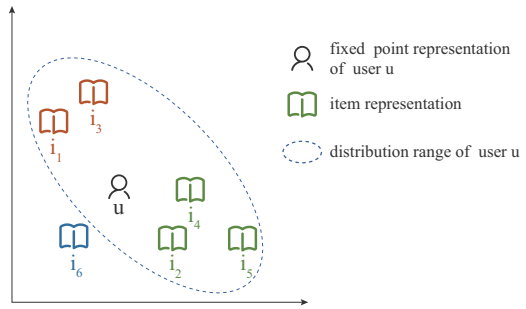

Fig. 1. An example to explain why deterministic methods can not handle uncertainty well. Different colors denote different categories.

the four items belong to two different categories. When we use a deterministic method to learn the user's preferences, u may locate in the middle of four items in the latent feature space (2D map). If we make recommendation based on the distance between u and candidate items, item i_6 (the category is different from any of the four items) may be recommended to user u rather than the ground-truth item i_5 (the same category as i_2 and i_4), because the distance between u and i_6 is smaller. Therefore, the fixed point representation cannot capture the uncertainty and incur inaccurate recommendation.

To address these problems, we put forward a novel **V**ariational **S**elf-**A**ttention **N**etwork (VSAN) for sequential recommendation. It introduces variational inference into self-attention networks to handle the uncertainty of user preferences by employing the variational inference paradigm [20], [21]. It is an approximate inference approach which involves: (1) parameterizing the inferred posterior distribution via a conventional neural network (inference network); and (2) transforming the inference problem into an optimization problem following variational fractional arithmetic. Specifically, we first feed the input embedding into the inference self-attention network. And we represent the obtained self-attention vector as density by imposing a Gaussian distribution, which handles the uncertainty of user preferences. Next, depending on the variational parameters output from the inference self-attention network, we can get a corresponding latent variable, which has the ability to characterize the uncertainty of user preferences. Then, in order to capture the user's long-range and local dependencies, we employ another self-attention network to model the generative process, which generates the final user representation based on the latent variable. Finally, we utilize the generative user representation to predict the probability of interacting on the next item(s). Recalling the example in Figure 1, in our work, the user's position in the latent space is a distribution range marked by the dashed ellipse. In this way, i_5 is within the distribution range of u, and i_6 is outside the distribution range of u, so the ground-truth item i_5 can be recommended to u. To summarize, our contributions are listed as follows:

- To the best of our knowledge, it is the first effort that introduces variational inference into self-attention networks for sequential recommendation. It presents the self-attention vector as a density instead of a fixed point vector by variational inference, whose variance well captures the uncertainty and dynamics of the user's preferences.
- We propose the VSAN for sequential recommendation. It employs self-attention networks to model the inference and generative process of VAE, which captures not only the uncertainty of user preferences but also the long-range and local dependencies of the user's behavior sequence.
- We evaluate our proposed VSAN on two public real-world datasets. Our experimental results reveal that VSAN outperforms various baselines by a significant margin, which confirms the effectiveness of VSAN handling the uncertainty of user preferences.

TABLE I
TABLE OF NOTATIONS

Notations	Descriptions
$\mathcal{U}$, $\mathcal{X}$	the set of users and items
M, N,	the number of users and items
$s_t^u \in \mathcal{X}$	the t-th item interacted by user u
$\lvert N^u \rvert$	the total number of interactions of user u
S^u	the user u's sequential interactions: $S^u = \{s_1^u, s_2^u, \ldots, s_{\lvert N^u \rvert}^u\}$
d	the embedding dimension
n	the maximum sequence length
$A \in \mathbb{R}^{n \times d}$	item embedding matrix
$P \in \mathbb{R}^{n \times d}$	position embedding matrix
$I \in \mathbb{R}^{n \times d}$	input embedding matrix
θ, λ	parameters in the variational inference
Q_1, K_1, V_1	the query, key, and value, respectively in the Inference Self-attention Layer
h_1	the self-attention blocks in the Inference Self-attention Layer
$G_i^{h_1}$	the final output of the Inference Self-attention Layer
μ_λ	the mean of posterior distribution
σ_λ	the variance of posterior distribution
ε	a standard Gaussian variable
z	the latent variable
Q_2, K_2, V_2	the query, key, and value, respectively: in the Generative Self-attention Layer
h_2	the self-attention blocks in the Generative Self-attention Layer
$G_g^{h_2}$	the final output of the Generative Self-attention Layer
k	the model can focus on predicting the next k items
β	the term to control the KL term
$L_\beta(\theta, \lambda; S^u)$	the objective function

The rest of this paper is organized as follows. We first describe our problem statement in Section II. Then we provide the preliminaries in Section III. Next, we introduce our proposed model VSAN in Section IV, and present the experimental results in Section V. Section VI reviews the related work. Finally, Section VII concludes our paper.

II. PROBLEM STATEMENT

In this section, before delving into the details of the proposed model, we will first introduce basic notations used in this paper and then present the sequential recommendation problem. For clarity, Table I summarizes all the notations and their meanings used in the paper.

We denote a user set as $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and an item set as $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, where M and N represent the amount of users and items, respectively. For each user $u \in \mathcal{U}$, by sorting his/her interaction records with a chronological order, the user u's sequential interactions are denoted as $S^u = \{s_1^u, s_2^u, \ldots, s_{|N^u|}^u\}$, where $s_t^u \in \mathcal{X}$, s_t^u is the t-th item interacted by user u at time t and $|N^u|$ represents the total number of interactions of user u. We denote the historical interaction sequence of all users as $S = \{S^{u_1}, S^{u_2}, \ldots, S^{u_M}\}$. Formally, we define the problem of sequential recommendation as follows. Given S by time t, our task is to recommend the next item(s) users probably interact with at time $t+1$ through modeling S. Specifically, the input of our model VSAN is the users' historical interaction sequence S by time t, and the output is the item(s) which a user $u \in \mathcal{U}$ may interact with at time $t+1$.

III. PRELIMINARIES

Recently, variational autoencoder has been applied to many issues [22], [23] (e.g., large-scale document analysis, computational neuroscience, and computer vision). In the recommendation system scenario, given the user's historical sequence: $p(s_t^u | s_1^u, \cdots, s_{t-1}^u)$, the goal is to maximize the probability of the next item(s). Extending this goal to the entire training set, the conditional probability of all interacting items in all sequences is as below:

$$\begin{aligned} p(S) &= \prod_{S^u \in S} p(S^u) \\ p(S^u) &= \prod_{s_t^u \in S^u} p(s_t^u | s_1^u, \cdots, s_{t-1}^u). \end{aligned} \tag{1}$$

Then, the focus of the model becomes how to model the joint probability $p(S^u)$.

Within the VAE framework, a continuous latent variable representation z is first assumed, which is sampled from a standard normal distribution, i.e., $z \sim \mathcal{N}(0; I)$. The central intuition here is that standard normal distribution variables can generate even complex dependencies. Then, the generative process is as follows:

(1) Sample a continuous latent vector representation z from the multivariate Gaussian prior $p_\theta(z)$;

(2) The sequence S^u can be modeled through a conditional distribution $p_\theta(S^u|z)$, which is parameterized by θ.

Importantly, VAE usually uses highly flexible function approximators (e.g., neural networks) to parameterize $p_\theta(S^u|z)$. Although the latent random variable models are not uncommon, assigning the conditional $p_\theta(S^u|z)$ to the potentially highly nonlinear mapping from z to S^u is a rather unique characteristic of VAE [24]. Hence the joint probability $p_\theta(S^u)$ can be specified by the marginal distribution as follows:

$$p_\theta(S^u) = \int p_\theta(S^u|z) p_\theta(z). \tag{2}$$

In order to optimize parameter θ, the best way is to maximize the above Equation 2. However, the true posterior distribution $p_\theta(z|S^u)$ is usually complicated and challenging to solve. Since the true posterior $p_\theta(z|S^u)$ involves the integral of z, the traditional inference algorithms (e.g., Expectation-Maximization algorithm) can't be relied on. So here we introduce a relatively simple distribution $q_\lambda(z|S^u)$ to approximate the posterior distribution resorting to the variational inference [23], where λ represents another set of parameters. Through derivation and reorganization, the relationship between the log likelihood of $p_\theta(S^u)$ and the introduced distribution $q_\lambda(z|S^u)$ is given by:

$$\begin{aligned} \log p_\theta(S^u) &= \int q_\lambda(z|S^u) \log p_\theta(S^u) \mathrm{d}z \\ &= \int q_\lambda(z|S^u) \log \frac{p_\theta(S^u|z) p_\theta(z)}{p_\theta(z|S^u)} \mathrm{d}z \\ &= \int q_\lambda(z|S^u) \log \frac{p_\theta(S^u|z) p_\theta(z)}{p_\theta(z|S^u)} \frac{q_\lambda(z|S^u)}{q_\lambda(z|S^u)} \mathrm{d}z \\ &= \int q_\lambda(z|S^u) \log p_\theta(S^u|z) \mathrm{d}z \\ &+ \int q_\lambda(z|S^u) \log \frac{q_\lambda(z|S^u)}{p_\theta(z|S^u)} \mathrm{d}z \\ &- \int q_\lambda(z|S^u) \log \frac{q_\lambda(z|S^u)}{p_\theta(z)} \mathrm{d}z \\ &= E_{z\sim q}[\log p_\theta(S^u|z)] + KL[q_\lambda(z|S^u)||p_\theta(z|S^u)] \\ &- KL[q_\lambda(z|S^u)||p_\theta(z)] \\ &\geq E_{z\sim q}[\log p_\theta(S^u|z)] - KL[q_\lambda(z|S^u)||p_\theta(z)], \end{aligned} \tag{3}$$

where $KL(q||p)$ denotes Kullback-Leibler Divergence between two distributions q and p. The ultimate goal is transformed into maximizing the two terms on the right side of inequality 3, which are called Evidence Lower Bound Objective (ELBO).

As a result, the modeling needs the help of two neural networks: the former infers the latent representation z based on S^u through $q_\lambda(z|S^u)$. The latter generates corresponding S^u from the latent representation z depended on $p_\theta(S^u|z)$. The learning process is controlled by the ELBO regarding their parameters mentioned above.

IV. THE ARCHITECTURE OF VARIATIONAL SELF-ATTENTION NETWORK (VSAN)

According to the framework we have discussed in Section III, we employ the self-attention networks to model the inference process and the generative process of the VAE, respectively. Figure 2 illustrates the overall architecture of the VSAN, which consists of five parts, i.e., Embedding Layer, Inference Self-attention Layer, Latent Variable Layer, Generative Self-attention Layer, and Prediction Layer.

In more detail, the left Embedding Layer converts item sparse vectors into low-dimensional dense vectors and obtains the item embedding and position embedding. And then, we feed the input embedding into the Inference Self-attention Layer. By imposing the Gaussian distribution, we model the obtained self-attention vector as a density instead of a conventional fixed-point representation by variational inference to characterize the uncertainty and dynamics of user preferences well. Next, we can obtain a corresponding latent variable with the help of the previous layer's variational parameters. Then, we can also capture long-term and local dependencies of the user's behavior sequence through another Generative Self-attention Layer. Finally, the Prediction Layer can predict the probability of the next item(s) that user may interest in depending on the generative user representation.

A. Embedding Layer

As shown in Figure 2, the input includes both item embedding and position embedding. First, we transform the user's historical interaction sequence $S^u = \{s_1^u, s_2^u, \ldots, s_{|N^u|}^u\}$ into a fixed-length sequence $S^u = \{s_1^u, s_2^u, \ldots, s_n^u\}$, where n denotes the maximum sequence length that our network models. We generate the sequence of n interaction records as [16]. For users whose sequence length is greater than n, we only select the nearest n items. For users whose sequence length is less than n, we repeatedly add the zero vector to the left side of the sequence as a padding item to convert its sequence length to n. Then, we construct one consecutive item embedding matrix $X \in \mathbb{R}^{N \times d}$, and obtain the input embedding matrix $A \in \mathbb{R}^{n \times d}$, where d represents the embedding dimension and $A_i = X_{s_i^u}$. We can note that unlike RNN or CNN, self-attention network ignores the positional information of historical interaction sequence [25]. Therefore, we add a learnable positional matrix $P \in \mathbb{R}^{n \times d}$ into the input matrix A as the final input embedding as follows:

$$I = \begin{bmatrix} A_1 + P_1 \\ A_2 + P_2 \\ \cdots \\ A_n + P_n \end{bmatrix} \tag{4}$$

In summary, the embedding layer first takes measures to obtain two embedding matrices, i.e., the item embedding matrix and the position embedding matrix, and then add them to get the input embedding matrix we finally need.

B. Inference Self-attention Layer

After obtaining the final input embedding, we feed it into the inference self-attention layer to output corresponding variational parameters of posterior distribution $q_\lambda(z|S^u)$. The top of Figure 2 demonstrates the specific structure of the inference self-attention layer.

1) ***Dot-product Attention****:* Formally, we first adopt the dot-product attention used in [13] as follows:

$$D_1 = softmax(\frac{Q_1 K_1^T}{\sqrt{d}})V_1, \tag{5}$$

where d denotes the final input embedding dimension and the scaling factor of $\frac{1}{\sqrt{d}}$ is to prevent excessive inner product values, especially when the size is high. Moreover, the Q_1, K_1 and V_1 represent the query, key, and value, respectively. In the Inference Self-attention Layer, Q_1, K_1 and V_1 are all generated by I as below:

$$Q_1 = IW^{Q_1}, K_1 = IW^{K_1}, V_1 = IW^{V_1}, \tag{6}$$

where $W^{Q_1}, W^{K_1}, W^{V_1} \in \mathbb{R}^{d \times d}$ represent the three projection matrices. After obtaining the three matrices through Equation 6, we input them into the above dot-product attention operation to obtain the result D_1. It is worth noting that sequential recommendation predicts the $(i+1)$-th item that a user is likely to prefer based on only the first i items of the user's interaction sequence. However, the i-th output of the dot-product attention (D_{1i}) contains the embedding of subsequent items, which can make the model unstable. Thus, similar to [16], when $j > i$, in order to ensure that D_{1i} is not affected by the embedding of subsequent items, we prohibit all links between Q_{1i} and K_{1j}.

2) ***Residual Connection and Layer Norm****:* To a certain extent, some research works (e.g., [26]) have proved that multi-layer neural networks can capture valuable features hierarchically. However, as the deeper network has the ability to begin to converge, a degradation problem has been revealed: the depth of the switching network increases, the accuracy tends to be saturated (which may not be surprising), and then quickly degrades. It is unanticipated that this degradation is not caused by overfitting. And adding more layers to a model of appropriate depth will lead to higher training errors, as reported in [27].

The proposal of residual network [28] solves this dilemma. The residual network's key idea is to spread the low-level information to higher layers through residual connections. In other words, if we think the low-level features are useful, we can easily propagate them to the final layer by this network. The existing sequential recommendation methods have shown that the last interacted item plays a crucial role in predicting the next item [2], [29], [30]. Therefore, we assume that the residual connections are also useful in our work. We utilize the residual connections to propagate low-level features to the high-level as follows.

$$E_1 = LayerNorm(D_1 + I), \tag{7}$$

where $LayerNorm$ denotes the Layer Normalization operation [31]. We apply the same method like [16]. Layer normalization is used to normalize the inputs, which makes the training of the neural network fast and stable. Moreover, different from batch normalization, the statistical information used in layer normalization is independent of other samples in the same batch.

3) ***Point-Wise Feed-Forward Network****:* Although the self-attention network can use adaptive weights to aggregate the embeddings of all previous items, it is still a linear model in the end. Therefore, a two-layer point-wise fully-connected

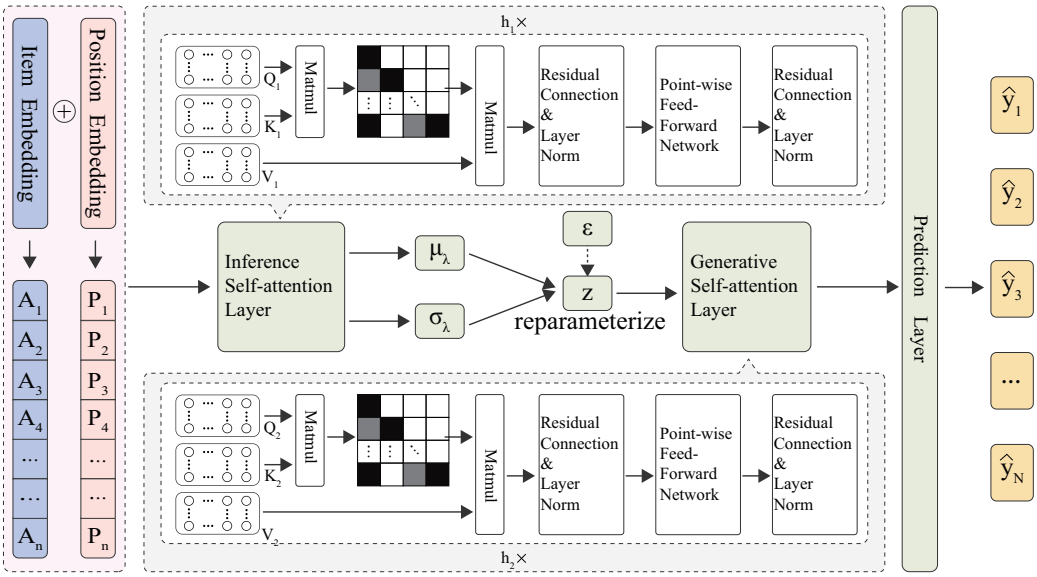

Fig. 2. The Architecture of the Variational Self-Aattention Network.

network with a ReLU activation function is used to consider the interaction among different potential dimensions and makes the network non-linear. Finally, the process is defined as follows:

$$F_1 = ReLU(E_1 W_1 + b_1)W_2 + b_2, \tag{8}$$

where W_1, W_2, b_1, b_2 are all network parameters. Please note that, in order to avoid information leakage (from back to front), there is still no interaction between E_{1i} and E_{1j} $(i \neq j)$. After that, we perform residual connection and layer normalization operations again on the obtained output F_1, as shown below:

$$G_i = LayerNorm(F_1 + E_1), \tag{9}$$

*4) **Applying Multiple Self-Attention Blocks**:* For convenience and simplicity, we define the entire self-attention network described above as:

$$G_i = SAN(I), \tag{10}$$

After the above process, G_i essentially integrates the embeddings of all previous items. To capture more complex item transitions, we can also stack h_1 self-attention blocks. Specifically, we apply multiple self-attention blocks (i.e., all the above inference process at the Inference Self-attention Layer) as follows:

$$\begin{aligned} G_i^1 &= SAN(G_i^0) \\ G_i^2 &= SAN(G_i^1) \\ &..... \\ G_i^{h_1} &= SAN(G_i^{h_1-1}), \end{aligned} \tag{11}$$

where $G_i^0 = I$. In the section of experiments, we will explore the effect of h_1 (i.e., the number of blocks at the Inference Self-attention Layer) on model performance.

*5) **Obtaining the Variational Parameters**:* According to the section preliminaries, the posterior $q_\lambda(z|S^u)$ is estimated based on the final self-attention vector, whose mean and variance are as follows:

$$\mu_\lambda = l_1(G_i^{h_1}), \quad \sigma_\lambda = l_2(G_i^{h_1}), \tag{12}$$

where $l(\cdot)$ represents linear transformations. Therefore, in this way, the deterministic self-attention vector $G_i^{h_1}$ is represented corresponding to a Gaussian distribution rather than traditional fixed-point representation, whose variance can well capture the uncertainty of user preferences.

C. Latent Variable Layer.

We can sample the latent variable z according to $q_\lambda(z|S^u)$. However, depending on the μ_λ and σ_λ(that in turn depend on λ), the sampling is an indeterminate function and is not differentiable. We can resort to the "reparameterization trick" [20]. Specifically, instead of sampling z, we can sample the auxiliary noise variable ε depending on a fixed distribution, and then obtain z through a differential transformation that depends on λ, ε and S^u. Then, we reparameterize z to a function of μ_λ and σ_λ as below:

$$\begin{aligned} \varepsilon &\sim \mathcal{N}(0; I) \\ z &= \mu_\lambda + \sigma_\lambda \cdot \varepsilon, \end{aligned} \tag{13}$$

where ε is a standard Gaussian variable that plays the role of introducing noises. In this way, the randomness in the sampling process can be isolated, and we can back-propagated the gradient of λ through the sampled z.

D. Generative Self-attention Layer.

Recently, as a particular case of attention mechanism, self-attention network has been successfully applied in many fields.

It not only has fewer parameters but also can guarantee the calculation efficiency. At the same time, it can flexibly capture the long-term and local dependence of users. Therefore, to define the generative process, we also employ the self-attention network, which is used to generate a corresponding S^u based on $p_\theta(S^u|z)$. Depending on the latent variable z, the $p_\theta(S^u|z)$ is formulated as:

$$p_\theta(S^u|z) = \prod_{s_t^u \in S^u} p_\theta(s_t^u|s_1^u, \cdots, s_{t-1}^u, z) = \prod_{s_t^u \in S^u} p_\theta(s_t^u|G_g^{h_2}), \tag{14}$$

where $G_g^{h_2}$ is the final output of the generative self-attention layer. The structure of the generative self-attention layer is shown on the bottom of Figure 2. Similar to the inference self-attention layer, we can get that:

$$\begin{aligned} D_2 &= softmax(\frac{Q_2 K_2^T}{\sqrt{d}})V_2, \\ Q_2 = zW^{Q_2}, K_2 &= zW^{K_2}, V_2 = zW^{V_2}, \end{aligned} \tag{15}$$

where $W^{Q_2}, W^{K_2}, W^{V_2}$ denote the projection matrices. Then, for the same reason, we apply residual connection and layer normalization in our generative self-attention layer. Moreover, we also adopt a two-layer fully-connected network with a ReLU activation function. Finally, the generative process is as follows:

$$\begin{aligned} E_2 &= LayerNorm(D_2 + z), \\ F_2 &= ReLU(E_2 W_{11} + b_{11})W_{22} + b_{22}, \\ G_g &= LayerNorm(F_2 + E_2), \end{aligned} \tag{16}$$

Specifically, we apply h_2 self-attention blocks (i.e., all the above generative process denoted by SAN_g) as follows:

$$\begin{aligned} G_g^1 &= SAN_g(G_g^0) \\ G_g^2 &= SAN_g(G_g^1) \\ &..... \\ G_g^{h_2} &= SAN_g(G_g^{h_2-1}), \end{aligned} \tag{17}$$

where $G_g^0 = z$. At this point, we obtain the final output by the generative self-attention layer.

Note that the above generative process only focuses on the next item in the user's historical sequence. Fortunately, it is flexible enough to focus on the next k items. The most straightforward way is to treat s_t^u as a time-ordered multi-set:

$$p_\theta(S^u|z) = \prod_{s_t^u \in S^u} p_\theta(s_t^u, \cdots, s_{t+k-1}^u|G_g^{h_2}), \tag{18}$$

E. Prediction Layer.

Based on the final sequence representation $G_g^{h_2}$ obtained by the generative self-attention layer, we can estimate the probabilities of being interacted next for all the candidate items as following:

$$\hat{y}^{(u,t)} = softmax(G_g^{(h_2,t)} W_g + b_g), \tag{19}$$

where $G_g^{(h_2,t)}$ represents the t-th row of $G_g^{h_2}$, $W_g \in \mathbb{R}^{N \times d}$, $b_g \in \mathbb{R}^N$, N denotes the number of candidate items. Similar to [32], in the evaluation phase, we simply consider the latent representation z for S^u as the mean of the variational distribution (i.e., μ_λ). Following the section of preliminaries, the loss function of our proposed VSAN is converted into the following equation:

$$\begin{aligned} L_\beta(\theta, \lambda; S^u) &= \beta \cdot KL[q_\lambda(z|S^u)||p_\theta(z)] - E_{z\sim q}[\log p_\theta(S^u|z)] \\ &= \beta \cdot \frac{1}{2} \sum_j (-\log \sigma_{\lambda j}^2 + \mu_{\lambda j}^2 + \sigma_{\lambda j}^2 - 1) \\ &\quad - \sum_t y^{(u,t)} log \hat{y}^{(u,t)}. \end{aligned} \tag{20}$$

where $y^{(u,t)}$ denotes the true item that the user likes at time t. Based on the Equation 14, $y^{(u,t)}$ denotes a one-hot vector, and $y_i^{(u,t)} = 1$ when item i is interacted at time t. Otherwise, $y_i^{(u,t)} = 0$. Based on the Equation 18, $y^{(u,t)}$ is a multi-hot vector, and $y_a^{(u,t)}, \cdots, y_b^{(u,t)} = 1$ when item $a, \cdots, b$ are interacted at time $t, \cdots, t+k-1$. Otherwise, $y_a^{(u,t)}, \cdots, y_b^{(u,t)} = 0$.

Inspired by KL annealing [22], we introduce a β to control the KL term on the above loss function. KL annealing is a common heuristic method used to train VAEs when the model is not fully utilized. We only need to multiply the KL term by a weight coefficient, which is β in our work. At the beginning of training, set the coefficient size to 0, give $q_\lambda(z|S^u)$ a little more time to learn to encode the information of S^u into z, and then gradually increase the coefficient as the training step increases.

Finally, by minimizing the loss function, we can jointly optimize the parameters θ and λ.

F. Complexity Analysis

1) Time Complexity: The time consumed by our model is mainly in the inference self-attention layer and generative self-attention layer, which is $O(n^2 d + nd^2)$. The former one comes from the self-attention network, and the latter one comes from the feed-forward network. Our main cost is $O(n^2 d)$ for the self-attention network. Comparing with SASRec [16], our model can handle the uncertainty of user preferences without increasing the time complexity, while the time complexity of RNN is $O(nd^2)$ and that of CNN is $O(wnd^2)$, where w is the window size of the convolutional filter. In most cases, our model is much faster than RNN and CNN. Moreover, the calculations in each self-attention layer are fully parallelizable, which are suitable for GPU acceleration and greatly reduces the training time. Therefore, our model is time-efficient and can have great practical value.

2) Space Complexity: The space consumed by our model is mainly in the item embedding and the parameters involved in the inference self-attention layer and generative self-attention layer (mainly including the self-attention calculation, feed-forward network, and layer normalization). Therefore, the total space complexity of the VSAN model is $O(Nd + nd + d^2))$.

V. EXPERIMENTS

In this section, we first briefly present the datasets, eight baseline methods, the evaluation metrics, and our implementation details in our experimental settings. Then, we compare our proposed model VSAN with state-of-the-art baseline methods, and show the experimental results of all models and analyze the reasons. Moreover, we study the influence of model components on the performance of our model VSAN. Finally, we discuss the impact of some key parameters on the model results. Specifically, to study the validity of our model VSAN, we conduct experiments to try to answer the following questions:

- **RQ1** Does our method VSAN outperform the state-of-the-art baselines?
- **RQ2** What is the influence of inference and generative self-attention blocks, i.e., h_1 and h_2 in the VSAN?
- **RQ3** What is the influence of latent variable z and feed-forward network on model performance?
- **RQ4** How do the parameters, such as embedding dimension d, the value of k, and the dropout rate, affect the effectiveness of VSAN?

A. Datasets

Amazon[1] and MovieLens[2] datasets are widely used in sequential recommendation [16], [33], [34]. Take the Beauty dataset as an example. A user purchased a shampoo at Amazon, and then purchased a conditioner followed by a hair mask. This user is likely to purchase hair oil next instead of other skin care products. Because the normal hair care process requires the use of these four products in sequence. In such a case, each of the user's next actions depends on the previous ones. All the four consumption actions are sequentially dependent. Moreover, the two real-world datasets belong to different domains and have different sparsity. The experimental results obtained from them are more universal and convincing. Therefore, we perform our experiments on the two real-world datasets. For the Amazon dataset, as an E-commerce platform dataset, we apply the "Beauty" category based on a 5-core version and filter out users who have interacted with less than five items. We binarize explicit data by discarding ratings of less than four. For the MovieLens, we adopt the version MovieLens-1M (ML-1M) and perform the same operations as the Amazon dataset.

Similar to [33], we split datasets according to strong generalization [35]: we divide the dataset into training, validation, and test set based on users. The full click histories of the training users are employed to train the network. We set the same number of users for the validation and the test set, which are called held-out users. In the last row of Table II, we present the number of held-out users. When evaluation, for each held-out user, we use the first 80% of time-ordered click histories to learn the necessary representations and then apply the rest of 20% to evaluate the performance of the network.

[1]http://jmcauley.ucsd.edu/data/amazon/links.html

[2]http://files.grouplens.org/datasets/movielens/

TABLE II
DATASETS STATISTICS.

Datasets	Beauty	ML-1M
#user	14,993	6,031
#item	12,069	3,516
#interactions	130,455	571,519
sparsity	99.93%	97.30%
# of held-out users	1,200	750

Although this operation is more complicated, we believe that this way is more robust and realistic than weak generalization, in which the same user can exist during both training and evaluation. Table II summarizes the information of the datasets after preprocessing.

B. Baselines

We compare our presented method VSAN with the following various models. Note that, for the baselines that can only provide meaningful predictions for users who are already utilized during the training phase, we adopt the same operation as [33]. Now, in order to better understand these models, we briefly introduce our competitors.

POP: It is a simple baseline method, that ranks items by their popularity, and the users are recommended the most popular items.

BPR [36]: To model the relative preferences of users, BPR designs a pair-wise optimization method combining with the matrix factorization model. This is the classic method of constructing recommendation from implicit feedback data.

FPMC [2]: A hybrid model integrates underlying Markov chain and the normal matrix factorization model for recommendation. The method uses a paired interaction model to decompose the transition cube, which is a special case of Tucker Decomposition.

TransRec [30]: This model considers third-order relationships between the user, the user's previously interacted item, and the next item by modeling users as translation vectors acting on item sequences. Embed the item as a point in the (latent) "transition space"; and then each user is represented as a "translation vector" in the same space.

GRU4Rec [6]: A Gated Recurrent Unit(GRU)-based model for session-based recommendation. It contains GRUs to model user interaction sequences and utilizes session-parallel mini-batches as well as a pair-wise loss function for training.

Caser [34]: This model utilizes CNN to capture sequential patterns. The core opinion is to embed a sequence of recent items into the "image" among time and latent spaces, and apply convolution filters to study the sequential patterns as local features of the image.

SVAE [33]: This model combines recurrent neural networks and variational autoencoders to model the temporal dynamics of user preferences. The key idea is the variational autoencoders have the ability to represent latent spaces, and the recurrent neural networks have powerful sequence modeling capabilities.

SASRec [16]: A state-of-the-art model, which leverages self-attention networks for sequential recommendation. It can not only model long-term preferences but also capture local dependencies.

C. Evaluation Metrics

To compare the performance of each model, we take three metrics widely used in previous works, including Precision@N, Recall@N, and Normalized Discounted Cumulative Gain (NDCG@N). The larger the values of three metrics, the better the performance is. Here, we set $N = \{10, 20\}$. The specific information is as follows:

- Precision@N is used to describe the percentage of the final recommendation list is the user-item rating record, i.e., what the user actually likes. We denote a recommendation list of top N predicted items for a user as R_N, and using T to represent the corresponding test set. Therefore, the calculation process of the Precision@N is as follows:

$$Precision@N = \frac{|T \cap R_N|}{N}. \tag{21}$$

- Recall@N describes the percentage of users—item rating records are included in the final recommendation list. We adopt the same symbolic representation as Precision@N, and then the calculation process is as below:

$$Recall@N = \frac{|T \cap R_N|}{|T|}. \tag{22}$$

- NDCG@N is an evaluation index to measure the quality of ranking, which puts more emphasis on the relevance of items at the top of the recommender list. It is defined as [33].

D. Implementation Details

We implement our proposed VSAN with Tensorflow. We adopt the Adam optimizer as the optimizer. Our learning rate is set as 0.001 on two datasets. The batch size on two datasets are set as 128. The embedding dimension on the Beauty and ML-1M datasets are set as 200. On the Beauty dataset, we all stack 1 self-attention blocks on the inference and generative self-attention layer. As for the ML-1M dataset, we stack 3 and 1 self-attention blocks respectively on the inference and generative self-attention layer. We set the dropout rate of turning off neurons as 0.2 on the ML-1M dataset and 0.5 on the Beauty dataset. Moreover, we set the maximum sequence length n as 200 on the ML-1M dataset and 50 on the Beauty dataset respectively. We conducted multiple experiments to ensure that the error of every experimental result is negligible. Moreover, we report the average performance under five times. We will discuss some other hyper-parameters below.

E. Comparison of Performance (RQ1)

To prove the effectiveness of our proposed method VSAN, we compare the performance of VSAN with eight baselines on the two datasets. Table III demonstrates the overall performance of all models. From this table, we can get the following interesting observations.

First, it is no doubt that all other baselines outperform BPR and POP on both datasets since BPR and POP are non-sequential recommendation methods. This demonstrates that sequential information is helpful for improving the recommendation performance, where the two ways ignore it. Surprisingly, POP performs a little better than BPR on the ML-1M dataset. We suspect that this may be because the number of items on the ML-1M dataset is relatively small. Then POP is easier to recommend the items the user may like from popular items.

Second, the performance of the neural network models, such as Caser and GRU4Rec, is better than traditional baselines (e.g., FPMC and TransRec) under most cases, which validates the effectiveness of deep learning. Moreover, SASRec outperforms all other baselines with significant progress on Beauty, and also gets relatively good results on ML-1M. This confirms the effectiveness of the self-attention network in capturing users' long-range dependencies and sequential patterns, even though it does not use any RNN or CNN structures. Besides, interestingly, we find that the results of Caser and SVAE are volatile. This may be because the beauty dataset is very sparse, Caser and SVAE can not obtain good results on sparse datasets using CNN and RNN structures.

Finally, no matter the dataset is sparse or dense, it is clear to observe that our proposed method VSAN consistently outperforms all baselines with a large margin. VSAN gains 20.60% NDCG@10, 14.17% Recall@10 and 15.48% Precision@10 improvements (on average) against the strongest baseline. Compared with SASRec, our proposed VSAN can characterize the uncertainty of user preferences through the structure of VAE and gains 22.57% NDCG@10, 15.38% Recall@10 and 18.53% Precision@10 improvements (on average) against the SASRec. Compared with SVAE that uses RNN, the results reveal that VSAN can better capture the long-term dependencies via the self-attention network. Moreover, compared with the second-best method, VSAN improves a lots (e.g., it grows 32.74% at NDCG@10) as expected on the Beauty dataset, though the Beauty dataset is very sparse. It is easy to understand that our network can well capture the uncertainty of user preferences, while the uncertainty is pervasive in sparse datasets.

F. Influence of Components (RQ2-RQ3)

1) Influence of the number of self-attention blocks (RQ2): As we discussed above, we can stack h_1 and h_2 self-attention blocks on the inference and generative self-attention layer, respectively. Table IV only summaries the results at Recall@20 on two datasets due to space constraints. When $h_1 = 0$, it means that no self-attention network is used in the inference self-attention layer, and only the input embedding with position information is included. When $h_2 = 0$, it means that the latent variable z is directly employed to predict the probability of the candidate items. There is no doubt that when h_1 and h_2 are equal to 0, the result is poor. And when we stack too many blocks, e.g., $h_1 = h_2 = 3$, it also gets poor performance. This may be because stacking more

TABLE III
THE OVERALL PERFORMANCE OF ALL MODELS ON BEAUTY AND ML-1M DATASETS (IN PERCENTAGE).

Datasets	Beauty						ML-1M					
Metrics	NDCG		Recall		Precision		NDCG		Recall		Precision	
	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20
POP	0.607	0.809	1.401	2.201	0.140	0.110	0.966	1.431	2.212	4.075	0.221	0.204
BPR	1.196	1.749	2.360	4.600	0.236	0.230	0.939	1.386	2.027	3.807	0.203	0.190
FPMC	2.343	3.296	4.478	7.707	0.941	0.777	4.241	5.727	4.094	7.879	3.507	3.533
TransRec	2.302	2.980	4.163	6.404	0.765	0.626	6.105	7.060	4.327	7.804	4.920	4.473
GRU4Rec	3.551	4.347	4.568	7.389	0.800	0.663	13.754	15.687	4.523	8.565	4.214	4.086
Caser	3.072	3.783	4.801	7.115	1.168	0.912	19.778	21.794	14.445	22.993	15.960	13.520
SVAE	2.697	3.589	4.052	6.816	0.933	0.821	19.413	21.672	14.108	22.942	16.093	13.700
SASRec	5.105	6.180	7.796	11.027	1.825	1.333	19.084	20.724	14.130	21.897	15.213	12.480
VSAN	**6.776**	**7.772**	**9.338**	**12.472**	**2.292**	**1.542**	**21.450**	**22.973**	**15.681**	**23.709**	**16.960**	**13.780**
Improv.	32.74	25.76	19.78	13.10	25.57	15.66	8.45	5.41	8.56	3.12	5.39	0.58

TABLE IV
THE INFLUENCE OF THE NUMBER OF SELF-ATTENTION BLOCKS (RQ2) (IN PERCENTAGE).

Datasets	Recall@20	h_1			
		0	1	2	3
Beauty	$h_2 = 0$	10.681	12.225	11.723	10.656
	$h_2 = 1$	12.062	**12.496**	11.626	10.397
	$h_2 = 2$	12.225	11.372	10.810	9.873
	$h_2 = 3$	11.352	10.598	10.117	7.750
ML-1M	$h_2 = 0$	17.141	23.266	23.391	23.405
	$h_2 = 1$	22.607	23.657	23.546	**23.709**
	$h_2 = 2$	23.385	23.673	23.188	23.634
	$h_2 = 3$	23.693	23.646	23.314	23.106

TABLE V
THE INFLUENCE OF THE LATENT VARIABLE z (RQ3)(IN PERCENTAGE).

Datasets	Methods	@10		@20	
		NDCG	Recall	NDCG	Recall
Beauty	VSAN-z	6.385	8.957	7.378	12.015
	VSAN	6.776	9.338	7.772	12.472
	Improv.	6.12	4.25	5.34	3.80
ML-1M	VSAN-z	20.810	15.405	22.616	23.568
	VSAN	21.450	15.681	22.973	23.709
	Improv.	3.08	1.79	1.58	0.60

blocks would cause the VSAN to lose lower-level information. Moreover, Table IV demonstrates that the VSAN acquires the best result with $h_1 = 1, h_2 = 1$ on Beauty, while it obtains the best performance with $h_1 = 3, h_2 = 1$ on ML-1M. For h_1, we think that the ML-1M dataset is dense, and the layered self-attention structure helps to capture more complex item transitions. For h_2, we think that stacking more self-attention blocks during the generative process may weaken the expression of latent variable z.

2) Influence of the latent variable z (RQ3) : As we have discussed in the previous paper, relying on the parameters obtained by the inference self-attention layer, we can sample the latent variable z. We claim that uncertainty modeling with VAE leads to a better performance. Moreover, we believe that the existence of latent variable z can greatly improve the performance of the model. Table V demonstrates the specific result, where VSAN-z means that the process of obtaining the latent variable z is removed from our original model, and the result of the inference self-attention layer is directly fed into the generative self-attention layer. From the experimental results, we can see that no matter which dataset it is on and no matter which evaluation metrics it is based on, the result of our model is always better than the result of removing z (e.g., it grows 6.12% at NDCG@10 on Beauty dataset). Therefore, this proves that our method can capture the uncertainty of user preferences by modeling the latent variable z with VAE.

TABLE VI
THE INFLUENCE OF THE POINT-WISE FEED-FORWARD NETWORK(RQ3) (IN PERCENTAGE).

Datasets	Methods	@10		@20	
		NDCG	Recall	NDCG	Recall
Beauty	VSAN-all-feed	6.418	8.696	7.489	12.138
	VSAN-infer-feed	6.546	8.776	7.499	12.170
	VSAN-gene-feed	6.587	8.801	7.604	12.298
	VSAN	6.776	9.338	7.772	12.472
ML-1M	VSAN-all-feed	20.129	14.706	21.610	22.510
	VSAN-infer-feed	20.665	15.272	22.378	23.267
	VSAN-gene-feed	20.758	15.367	22.614	23.585
	VSAN	21.450	15.681	22.973	23.709

3) Influence of the point-wise feed-forward network (RQ3): The point-wise feed-forward network is used to consider the interaction among different potential dimensions and give the model non-linear expression ability. In order to show its capabilities, we design the related experiments, and the experimental results are shown in the table VI. In the table, VSAN-all-feed means removing all point-wise feed-forward networks in the inference self-attention layer and generative self-attention layer. VSAN-infer-feed represents only removing the point-wise feed-forward network in the inference self-attention layer,

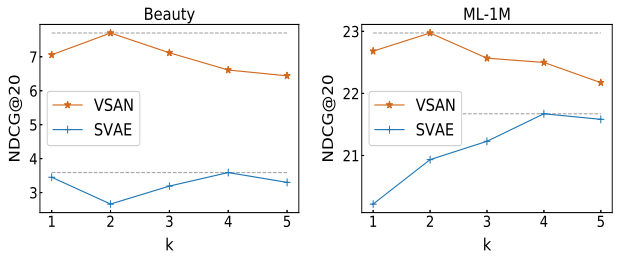

Fig. 3. The performance under different k.

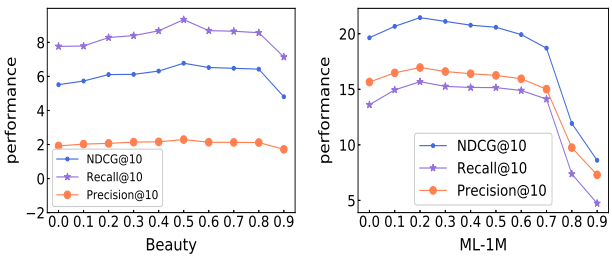

Fig. 5. The performance under different dropout rates.

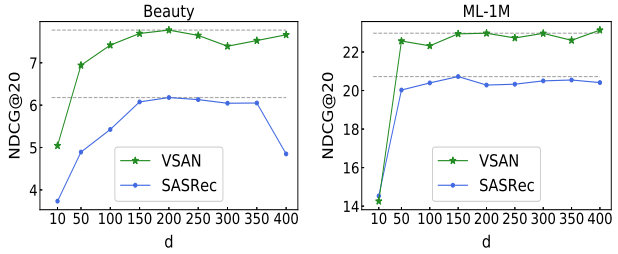

Fig. 4. The performance under different d.

while VSAN-gene-feed represents only removing the point-wise feed-forward network in the generative self-attention layer. It is obvious that VSAN-all-feed performs the worst on two datasets among several variant models. Because the point-wise feed-forward network is removed, the model lacks non-linear expression ability. Moreover, the result of VSAN-gene-feed is better than the result of VSAN-infer-feed, which shows that the point-wise feed-forward network is more important in the inference layer than in the generative layer, which makes the result worse when the inference layer removes it. This is because when the inference layer removes the feed-forward network, the model loses its non-linear expression ability and cannot generate a latent variable z closer to the user's preferences, which leads to a decrease in the performance of the model. Finally, our model VSAN obtains the best results, which further validates the importance of the point-wise feed-forward network in our work.

G. Influence of Hyper-parameters (RQ4)

1) Influence of the value of k: As we discussed earlier, our model can be easily extended to focus on predicting the next k items. Figure 3 displays the performance of VSAN with different k on two datasets compared to the SVAE, which can also be developed to focus on predicting the next k items. We can clearly observe that no matter the value of k, our method VSAN always performs better than SVAE. Moreover, we can observe that the increasing k can improve the effectiveness, but when k becomes larger, the model performance will decline. In our work, we choose $k = 2$ for our VSAN and $k = 4$ for SVAE on two datasets since they obtain the best performance.

2) Influence of the embedding dimension d: As we all know, the embedding dimension has a great influence on the model performance. Therefore, we investigate the influence of the embedding dimension d ranging from 10 to 400 on two datasets. Because SASRec performs well and stable among all baselines, we use it as a comparison. Figure 4 displays the experimental results. We can observe that VSAN consistently outperforms SASRec by a large margin regardless of the value of d. Obviously, the high dimension can improve the performance of the network. This phenomenon is similar to the traditional latent factor model. But when the dimension exceeds a certain value, the results will no longer increase, and even begin to decline. This shows that it may cause overfitting when the embedding dimension of the latent factor is too large. Finally, we choose 200 for our proposed VSAN on two datasets. And we set the dimension as 200 for SASRec on Beauty while setting 150 on ML-1M.

3) Influence of the dropout rate: Dropout means that during the training process of the deep learning network, the neural network unit is temporarily dropped from the network according to a certain probability. It has been proven to be an effective way to prevent overfitting of various neural networks [37], [38]. Therefore, we investigate the influence of the dropout rate ranging from 0 to 0.9 on two datasets. Figure 5 displays the experimental results under different dropout rate settings on both datasets. It can be seen from the figure that when the dropout rate is set to 0 (that is, no neurons are dropped), the result is relatively poor. This proves that dropout is indeed effective in preventing overfitting. Moreover, for the Beauty dataset, when the dropout rate is set to 0.5, the best result can be obtained. Therefore, in our work, we set it to 0.5. As for the ML-1M dataset, the model can obtain the best result when the dropout rate is set to 0.2, so we adopt the 0.2 in our work. Finally, we can observe that the two figures show the same trend: as the dropout rate increases, the performance of the model first improves and then declines or even declines sharply. This demonstrates that proper dropout rate helps improve the expressive ability and generalization ability of the model, but when the dropout rate is too large, too many neurons are lost, which will limit the expression of the model.

4) Influence of the β to control the KL term: In our work, we introduce the KL annealing to control the KL term. In order to prove the effectiveness of KL annealing, we set β to a fixed value ranging from 0 to 0.9 for related experiments. Figure 6 displays the experimental results under different β on both datasets. The results our model VSAN using KL annealing

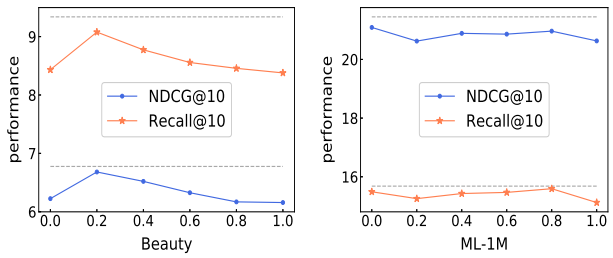

Fig. 6. The performance under different β.

are shown in the dotted line in the figure, and we can clearly see that our method achieves the best results on both datasets. This also further proves the effectiveness of KL annealing.

VI. RELATED WORK

In this section, we outline recent research work from three aspects, including Sequential Recommendation, Attention Mechanism, and Deep Generative Model.

A. Sequential Recommendation

Due to the considerable appeal and value of sequential recommendation, researchers have put a lot of effort into it. For instance, [2] presented a model that generated a personalized individual transition matrix for each user, and it combined matrix factorization (MF) with Markov chain (MC) for recommendation. Wang et al. [29] integrated the similarity of the factored items with a Markov chain to capture the user's long- and short-term preferences. However, all these Markov chain based approaches model only local sequential patterns between every two adjacent items but have difficulties to model long-term preferences and capture high-order relationships. However, as we know, the user's long-term preferences and high-level relationships are very important in sequential recommendation. To model the transition between adjacent items, [30] proposed a model TransRec, representing each user as a translation vector. Caser [34] employed convolutional filters to capture sequential patterns as local features of the image by embedding recent sequential items into an "image".

On the other hand, RNNs have been widely applied in sequential recommendation. For example, [6] employed GRU to predict the next item in a user session. [7] proposed the Spatio-Temporal Gated Network (STGN) to model personalized sequential patterns for users' long and short-term preferences in the next POI recommendation. Zhang et al. [9] presented a framework that directly models the dependency on users' sequential behaviors into the click prediction process through the recurrent structure in RNN. However, these models all employ RNNs, which struggle to preserve long-term dependencies. Furthermore, they are challenging to parallelize due to the sequential nature of RNN.

B. Attention Mechanism

Recently, self-attention networks have attracted a lot of attention. [13] proposed a model Transformer, which achieved significant progress in the machine translation task. Inspired by Transformer, SASRec [16] was proposed to use self-attention networks to model users' sequential behaviors for sequential recommendation. The model is very effective and flexible for both long-range and short-range dependencies, which were modeled by RNN and CNN respectively in the past. [12] captured heterogeneous user behaviors by introducing a self-attention network. Zhou et al. [39] proposed ATRank, which is an attention-based user behavior modeling framework. [40] proposed the FDSA network to capture both item and feature transition patterns by two separate self-attention blocks based on item-level and feature-level sequences. However, all the above models have the same disadvantage that they all employ deterministic self-attention networks to consider the users' preferences as a fixed point representation. We argue that these approaches are not sufficient to model the uncertainty and dynamics of user preferences.

C. Deep Generative Model

Recently, various deep generative models (e.g., variational autoencoders (VAE)) have a demonstrated potential and enabled significant performance in many applications, such as text generation [22], [41] and recommendation [32], [42]. With their successes, hybrid approaches integrated deep generative models (e.g., VAE) and deep learning (e.g., RNNs) have obtained attention. For example, [43] proposed a hierarchical latent variable model, which is a Bayesian version of GRU, and the inferred posterior distribution is parameterized through a GRU network. [33] introduced a recurrent version of the VAE to handle temporal information among the users' historical sequence and captured the temporal dynamics of user preferences upon different perspectives combining recurrent neural networks and variational autoencoders. [44] proposed a variational recurrent model for session-based recommendation, which models the sessions by a stochastic seq2seq pattern. [45] presented a hierarchical neural variational model, which captures users' distinct preferences, i.e., the users' general, long- and short-term preferences, through hierarchical Gaussian latent representations. However, most of these hybrid models use RNNs, and we have discussed many shortcomings of RNNS above.

Motivated by the success of the deep generative model, in our work, we follow the direction of the state-of-the-art self-attention networks and tightly combine VAE with self-attention networks. Therefore, our proposed network can make up for the shortcomings of the above methods. It can not only capture the uncertainty of user preferences, but also model the long-term and local dependencies of users' sequential behaviors.

VII. CONCLUSION

In this paper, a new **V**ariational **S**elf-**A**ttention **N**etwork (VSAN) was proposed for sequential recommendation. VSAN is the first model to introduce VAE into the self-attention network to capture latent user preferences. The self-attention networks were employed to model the inference and generative

process of VAE. As a result, our method VSAN could not only capture the uncertainty of user preferences resorting to VAE, but also model the long-term and local dependencies of users' sequential behaviors via self-attention networks. Extensive experimental results verified the effectiveness of our proposed method VSAN, compared to various baselines.

ACKNOWLEDGMENTS

This research was partially supported by NSFC (No. 61876117, 61876217, 61872258, 61728205), ESP of the State Key Laboratory of Software Development Environment, and A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

REFERENCES

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

[2] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*. ACM, 2010, pp. 811–820.

[3] J. Ye, Z. Zhu, and H. Cheng, "What's your next move: User activity prediction in location-based social networks," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 171–179.

[4] Q. He, D. Jiang, Z. Liao, S. C. Hoi, K. Chang, E.-P. Lim, and H. Li, "Web query recommendation via sequential query prediction," in *2009 IEEE 25th international conference on data engineering*. IEEE, 2009, pp. 1443–1454.

[5] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "Air: Attentional intention-aware recommender systems," in *ICDE*. IEEE, 2019, pp. 304–315.

[6] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *ICLR*, 2016.

[7] P. Zhao, H. Zhu, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, and X. Zhou, "Where to go next: A spatio-temporal gated network for next poi recommendation," in *AAAI*, vol. 33, 2019, pp. 5877–5884.

[8] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm." in *IJCAI*, vol. 17, 2017, pp. 3602–3608.

[9] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," *arXiv preprint arXiv:1404.5772*, 2014.

[10] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," *arXiv preprint arXiv:1805.04623*, 2018.

[11] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *AAAI*, vol. 33, 2019, pp. 3159–3166.

[12] X. Huang, S. Qian, Q. Fang, J. Sang, and C. Xu, "Csan: Contextual self-attention network for user sequential recommendation," in *Multimedia*. ACM, 2018, pp. 447–455.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[14] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[15] X. Li, J. Song, L. Gao, X. Liu, W. Huang, X. He, and C. Gan, "Beyond rnns: Positional self-attention with co-attention for video question answering," in *AAAI*, vol. 33, 2019, pp. 8658–8665.

[16] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*. IEEE, 2018, pp. 197–206.

[17] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *IJCAI*, 2019, pp. 3940–3946.

[18] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in wasserstein space," in *KDD*. ACM, 2018, pp. 2827–2836.

[19] J. Jiang, D. Yang, Y. Xiao, and C. Shen, "Convolutional gaussian embeddings for personalized recommendation with uncertainty," in *IJCAI*, 08 2019, pp. 2642–2648.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, 2014, pp. 3581–3589.

[22] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[23] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *JASA*, vol. 112, no. 518, pp. 859–877, 2017.

[24] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in neural information processing systems*, 2015, pp. 2980–2988.

[25] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[27] K. He and J. Sun, "Convolutional neural networks at constrained time cost." in *CVPR*. IEEE Computer Society, 2015, pp. 5353–5360.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE*, 2016, pp. 770–778.

[29] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*. IEEE, 2016, pp. 191–200.

[30] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*. ACM, 2017, pp. 161–169.

[31] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[32] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *WWW*. International World Wide Web Conferences Steering Committee, 2018, pp. 689–698.

[33] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," in *WSDM*. ACM, 2019, pp. 600–608.

[34] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*. ACM, 2018, pp. 565–573.

[35] B. Marlin, *Collaborative filtering: A machine learning perspective*. University of Toronto Toronto, 2004.

[36] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.

[37] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, 2012.

[38] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, pp. 1929–1958, 2014.

[39] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *AAAI,*, 2018, pp. 4564–4571.

[40] T. Zhang, P. Zhao, Y. Liu, V. Sheng, J. Xu, D. Wang, G. Liu, and X. Zhou, "Feature-level deeper self-attention network for sequential recommendation," in *IJCAI*. AAAI Press, 2019, pp. 4320–4326.

[41] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *ICML*. JMLR. org, 2017, pp. 1587–1596.

[42] F. Zhou, Z. Wen, K. Zhang, G. Trajcevski, and T. Zhong, "Variational session-based recommendation using normalizing flows," in *WWW*. ACM, 2019, pp. 3476–3475.

[43] S. P. Chatzis, P. Christodoulou, and A. S. Andreou, "Recurrent latent variable networks for session-based recommendation," in *DLRS*. ACM, 2017, pp. 38–45.

[44] Z. Wang, C. Chen, K. Zhang, Y. Lei, and W. Li, "Variational recurrent model for session-based recommendation," in *CIKM*. ACM, 2018, pp. 1839–1842.

[45] T. Xiao, S. Liang, and Z. Meng, "Hierarchical neural variational model for personalized sequential recommendation," in *WWW*. ACM, 2019, pp. 3377–3383.