Field-aware Variational Autoencoders for Billion-scale User Representation Learning

Ge Fan
Tencent Inc.
Shenzhen, China
gefan@tencent.com

Chaoyun Zhang
Tencent Inc.
Shenzhen, China
vyokkyzhang@tencent.com

Junyang Chen*
Shenzhen University
Shenzhen, China
junyangchen@szu.edu.cn

Baopu Li
Baidu USA
Sunnyvale, USA
baopuli@baidu.com

Zenglin Xu
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
xuzenglin@hit.edu.cn

Yingjie Li
Tencent Inc.
Shenzhen, China
wallaceyjli@tencent.com

Luyu Peng†
Tencent Inc.
Shenzhen, China
louispeng@tencent.com

Zhiguo Gong
University of Macau
Macau, China
fstzgg@um.edu.mo

***Abstract*—User representation learning plays an essential role in Internet applications, such as recommender systems. Though developing a universal embedding for users is demanding, only few previous works are conducted in an unsupervised learning manner. The unsupervised method is however important as most of the user data is collected without specific labels. In this paper, we harness the unsupervised advantages of Variational Autoencoders (VAEs), to learn user representation from large-scale, high-dimensional, and multi-field data. We extend the traditional VAE by developing Field-aware VAE (FVAE) to model each feature field with an independent multinomial distribution. To reduce the complexity in training, we employ dynamic hash tables, a batched softmax function, and a feature sampling strategy to improve the efficiency of our method. We conduct experiments on multiple datasets, showing that the proposed FVAE significantly outperforms baselines on several tasks of data reconstruction and tag prediction. Moreover, we deploy the proposed method in real-world applications and conduct online A/B tests in a look-alike system. Results demonstrate that our method can effectively improve the quality of recommendation. To the best of our knowledge, it is the first time that the VAE-based user representation learning model is applied to real-world recommender systems.**

***Index Terms*—User Representation Learning, Recommender Systems, Lookalike Systems, Variational Autoencoder**

I. Introduction

Understanding users' interests is the precondition to improve their quality of experience in a variety of Internet applications. This requires to extract knowledge from users' behavior data, so as to discover their latent interested topics and recommend appropriate contents to the targeted audiences. Since the common user features are usually sparse and high-dimensional, heavy feature engineering is necessary for preprocessing in modeling users' behaviors. Otherwise, those high-dimensional features will lead to the curse of dimensionality [1], which will significantly increase the complexity of the model.

To mitigate this issue, user representation learning [2] is proposed to reduce the reliance on feature engineering. It allows to learn a low-dimensional latent representation for individual users from the original high-dimensional counterparts by designing mapping functions. Recently, deep learning based techniques have become popular due to the non-linear representation ability, and have been proven effective in many real-world applications such as online games [3], recommender systems [4], and e-commerce ecosystems [5]. Nevertheless, most of existing deep learning approaches learn user representations with an end-to-end supervised manner, and only use single-source data collected at the same platform. As users' preference diversifies from different scenes and platforms, adopting single-source data may not comprehensively profile the users. Therefore, it is demanding to develop a universal embedding learner to capture users' profile from different perspectives to support various applications.

To fulfill the requirement, learning user representation in an unsupervised manner becomes a natural solution, as it can learn user representation from original data without specific labels. This is important since most of the data is unlabeled in real-world applications. For example, the performance of an online lookalike system relies on the quality of user representations. Moreover, a universal embedding of a user is useful for the entire system [6], [7]. Inspired by the success in the unsupervised image modeling and generation, Variational Autoencoders (VAEs) are playing an important role in learning representation from sparse data [8], [9], as they demonstrate outstanding modeling capacity with its non-linear probabilistic latent-variable. However, acquiring universal representation by VAEs from massive user profile data is not straightforward. We recognize there exist two challenges of learning representation from large-scale user data from multiple perspectives.

* Corresponding author.
† Luyu Peng leads the project in Tencent.

This work was supported in part by National Natural Science Foundation of China under Grant 62102265, in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), under Grant GML-KF-22-29, and in part by the Science and Technology Development Fund, Macau, SAR, under Grant FDCT/0068/2020/AGJ.

Zenglin Xu has been supported by a key program of fundamental research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826).

Specifically:

1) Multi-field fusion: Early research has shown that exploiting features in different fields is beneficial for click-through rate prediction. This has become a default process in recommender systems that embrace multi-field features [10]–[12]. Nevertheless, only few of them employ multi-field adaption in an unsupervised manner. Regarding VAEs, although the encoder can learn high-level representations from different fields by non-linear deep neural networks (DNNs), most of them neglect the importance of different fields in the decoder. However, the well-suited modeling between different fields in the decoder can benefit the user representations learning of VAEs. In addition, the importance of different fields varies in a system, which makes it difficult for fusing [13], [14].

2) Computational cost: Many Internet applications require to process billions of users, where billion-scale features exist and need to be processed in a system. For example, the number of Tencent's [1] monthly active users of smart device is over 1 billion, which generate tremendous amount of data with massive features. Thus, it is extremely computationally expensive to handle for current methods. To mitigate this issue, recent research adopts *feature hashing* to handle large-scale features [15], while this leads to the collision problem and is not efficient for billion-scale samples. Processing such magnitude of data within acceptable time and computing resources remains a challenge in industry.

To harness the advantages of both unsupervised learning and VAE, in this paper, we propose Field-aware Variational Autoencoders (FVAE), a novel VAE structure for user representation learning. The FVAE extends the traditional VAE by modeling each field of feature with independent multinomial distribution, to capture the diversity of information among different fields. This enables to jointly fuse knowledge in the encoder as well as decoder. In order to reduce the computation complexity, the FVAE first employs dynamic hash tables to efficiently transform the sparse inputs into dense tensors, then employs the batched softmax function to accelerate estimating the distribution of output. For those fields that are very sparse, the FVAE employs a feature sampling strategy, which randomly samples features again with a uniform distribution. This strategy is well-suited for user representation learning from features that follow the power-law distribution, as it improves the performance of other sampling strategies, such as Frequency and Zipfian [16]. In summary, the contributions are as follows:

- **Effectiveness.** To the best of our knowledge, we are the first to theoretically extend VAEs for multi-field data with the independent multinomial distribution, which captures the diversity of information among different fields, and jointly fuse knowledge in the encoder as well as the decoder. The results in section V-B and section V-C verify the efficiency of the proposed method.
- **Efficiency.** We propose to utilize several strategies (i.e., dynamic hashing, batched softmax, feature sampling) to improve the effectiveness of the proposed methods. Results in Table V show that the proposed method acquires thousands of speedup improvements in the training step compared with the original Mult-VAE, These proposed contributions make it possible to learn representations with billion-scale users within hours.
- **Practicality.** We design a framework to apply the proposed model in real-world recommendation systems. The learned user representations of the FVAE can be used in various downstream tasks, such as the matching step and the look-alike systems. Online A/B test shows that user embedding based on FVAE can improve the recommendation quality of our look-alike systems. *To the best of our knowledge, it is the first time that the VAE has been deployed in a billion-scale industrial look-alike systems.*

[1] https://www.tencent.com/en-us/

II. RELATED WORKS

A. VAEs on Sparse Data.

Inspired by the success in unsupervised learning, VAEs and their variants are applied in different types of data, including text [17], [18], graph [19]–[21], and recommender systems [8], [22], [23]. Vanilla VAEs [24], [25] are latent variable models that can approximate complex distributions. It can be employed as a representation learning . Beta-VAE [26] learns disentangled representations by a regularization coefficient β to control the Kullback-Leiber (KL) divergence term.

Recently, Liang et al. [8] extended VAEs to collaborative filtering by modeling implicit feedback data with multinomial likelihoods. They also utilize the annealing technique to avoid underfitting when learning with large, sparse, and high-dimensional data. Sachdeva et al. [22] exploited rich information of user profiles, and model temporal data with RNN to improve the accuracy of VAEs. Sankar et al. [21] proposed a generalizable VAEs framework, which utilized graph neural network architectures to selectively exploit information from social networks. Shenbin et al. [23] propose a Recommender VAE (RecVAE) to improve the Mult-VAE based model. Time-Aware Sequential Autoencoder (TASA) [27] is proposed to learn user representations by modeling the importance of temporal information within user sequences. It utilizes a long short-term memory (LSTM) network to embed the user sequence of activities. This method largely focuses on the user activities, while ignoring other important user profiles. The RecVAE designs a novel encoder architecture with a composite prior distribution for the latent distributions. It also develops a user-specific β to control the strength of KL term. Zheng et al. [28] employ a VAE in social recommendation by adopting an attention structure to feed the embedding information into an inherited VAE structure. Yu et al. [29] employ a Adversarial Variational Bayes (AVB) framework based on adversarial training. They show that Generative Adversarial Network (GAN) based reconstruction loss can further improve the performance of the VAE. Xie et al. [30] present the Adversarial and Contrastive Variational Autoencoder (ACVAE) for the sequential recommendation. The ACVAE uses the adversarial training under the AVB

framework for sequence generation to improve the quality of latent variables. However, those methods do not adapt VAEs to the multi-field profiles.

B. User Representation Learning.

Users' representations can be extracted from different sources, such as static features [31], [32] and dynamic behaviors [33].The matrix factorization (MF) based collaborative filtering methods are able to learn users' representations efficiently [34]. For instance, the probabilistic MF (PMF) is used as a probabilistic interpretation of the traditional MF method. It scales linearly with the number of ratings [35]. Furthere, a Bayesian treatment is applied in the PMF to exploit the Markov Chain Monte Carlo (MCMC), to perform approximation inference [36].

Deep learning techniques are widely employed for recommender systems [37]–[39]. As skip-gram models (e.g. word2vec) [40] had made great achievements in various linguistics tasks, they have been further employed to learn user representations. Barkan et al. [41] present a item-based collaborative filtering model called Item2Vec to produce embedding for items in a latent space. In the Item2Vec model, a user representation can be aggregated by its context historical items. An advanced version of Item2Vec [42] employs a attention mechanism to produce neural attentive users' representations. Enhanced Graph Embedding with Side information (EGES) [43] is a graph-based method, which constructs an item graph from users' behavior sequences, and learns the item representations by a weighted Skip-Gram model. The EGES method learns representations of items only, whereas user information is largely overlooked. Therefore, employing EGES for modelling the correlations between items and users becomes difficult.

Another effective way for improving the performance of recommendation is to develop the multi-field information of users/items profiles [10], [13], [44] . Covington et al. [44] applied DNNs to build user embeddings from rich-field features. The Deep Structured Semantic Models (DSSM) [45], [46] also utilizes DNNs to concurrently extract users' and items' high-level representations. Zhou et al. [12] introduced the attention mechanism to learn the interest representation of underlying user behaviors. Li et al. [13] represented users with multiple representation vectors by the Capsule Network [47]. Chen et al. [48] captured the sequential signals from users' behavior sequences by using the transformer model. Nevertheless, most of those works only focused on learning user embedding with end-to-end models from single source data. This leads to modest generalization of the representation, which will affect the performance of downstream tasks [49]. Conure [50] aims to address the catastrophic forgetting problem in the representation learning by transfer learning. It focuses on the finetuning stage in the learning process with a pretrained temporal convolutional network (TCN) model. Similar to the TASA, the Conure method trains with the user activities only, while overlooking other important information.

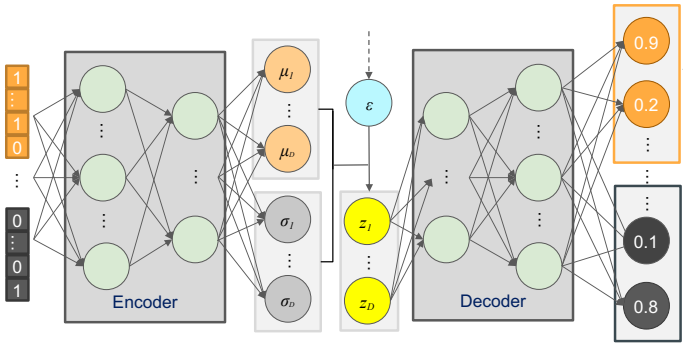

Fig. 1. The overall architecture of the proposed FVAE. Input features of different fields are fed to the encoder to learn the latent representation. The decoder accepts the sample of the latent representation to reconstruct original inputs. Each field is modeled by an independent distribution. μ and σ respectively denote the *mean* and the *standard deviation* of the latent learned representation. $\varepsilon \sim \mathcal{N}(0,1)$ is the sample noise of the *reparametrization trick*.

III. Problem Formulation

User representation learning is a sub-task of user modeling that aims at learning a latent low-dimensional embedding for each user from its features to support subsequent applications, e.g., recommender systems. We define the user feature matrix as $\mathbb{U} = [u_1, u_2, ..., u_N]^T$, where N is the number of users. Features are grouped in different fields. For each user i, we denote all features from K fields as $u_i = [(F_i^1)^T, (F_i^2)^T, ..., (F_i^K)^T]^T$, where F_i^k is a multi-hot vector embracing the features in the k-th field. For each $F_i^k = [F_{i,1}^k, F_{i,2}^k, \cdots, F_{i,J_k}^k]^T$, $F_{i,j}^k = 1$ means that the i-th user has the j-th feature of k-th field, and 0 means the user do not have the feature. We also define J_k as the number of the features of field k, and $J = \sum^k J_k$ to represent the total number of all features.

In this paper, we consider the user representation learning task as a probability-based unsupervised learning problem. Given features of users $\mathbb{U} \in \mathbb{N}^{N \times J}$. We aim at learning a mapping $f : u_i \rightarrow z_i$ for each user i, where z_i denotes the representation for the variables u_i, and $z_i = \mathcal{N}(\mu_i, \Sigma_i)$ is a Gaussian distribution. The mean value μ_i represents the position of user i in the embedding space, and Σ_i is the uncertainty vector of user i. The objective of the mapping f is to extract meaningful information from the original feature space for improving the performance of subsequent tasks.

IV. Proposed Method

In this section, we introduce the designed structure, inference process, learning algorithm of the proposed FAVE in large-scale data, as well as the deployed solution of the industrial system.

A. Field-aware Variational Autoencoder

We show the overall architecture of the Field-aware Variational Autoencoder (FVAE) in Fig. 1. The FVAE is an

improved variant of VAEs, which consists of an encoder and a decoder. The encoder learns a non-linear mapping from input features u_i to a latent distribution z_i, whereas the decoder reconstructs the output u_i' from the latent distribution z_i. In this paper, the representation of user i is modelled by the latent distribution z_i. The dimension of the input and output layer are set to the same. The objective of the FVAE is to minimize the reconstruction error between input u_i and output u_i'. We assume each field of observed feature vectors F_i^k follows an independent multinomial distribution. For each user i, the D-dimensional latent representations z_i is transformed via a sampling from a Gaussian prior distribution, defined as:

$$\begin{aligned} \pi^k(z_i) &\propto \exp\{f_{\theta^k}(z_i)\}, \\ F_i^k &\sim Mult(N_i^k, \pi^k(z_i)), \end{aligned} \tag{1}$$

where $N_i^k = \sum_j F_{i,j}^k$ denotes the total number of observed features in field k by user i, and π^k is the corresponding probability for each feature of field k . The softmax function is employed to normalize the probability vector $\pi^k(z_i)$ over the features in field k. In our works, $f_{\theta^k}(\cdot)$ is instantiated as the non-linear multilayer perceptron (MLP) with parameters θ^k:

$$\hat{F}_i^k = f_{out}^k(f_{L_d}(...f_2(f_1(z_i)))), \tag{2}$$

where f_{out}^k denotes the mapping function for the output layer of k-th field, f_j denotes the j-th hidden layer of the decoder. Note that the parameters of the MLP in the decoder are shared across all fields, excluding the output layer. By assuming the probability of all feature fields is independent, the log-likelihood of data of user i is equivalent to averaging the log-likelihood of all fields, formulated as:

$$\begin{aligned} \log p_\theta(u_i|z_i) &= \log p_\theta([F_i^1, F_i^2, ...F_i^K]|z_i) \\ &= \frac{1}{K}\sum_k \log p_{\theta^k}(F_i^k|z_i). \end{aligned} \tag{3}$$

The log-likelihood for field k of user i is therefore:

$$\log p_\theta(F_i^k|z_i) = \sum_j F_{i,j}^k \log \pi_j^k(z_i). \tag{4}$$

B. Variational inference

We employ the variational inference to estimate parameters of FVAEs. The first step of the inference process is to compute the lower-bound of the log marginal likelihood from the input data [51]. Following the work in [8], the lower-bound of the log marginal likelihood can be represented as:

$$\begin{aligned} \log p(u_i;\theta) &\geq \mathbb{E}_{q_\phi(z_i|u_i)}[\log p_\theta(u_i|z_i)] - KL(q_\phi(z_i|u_i)\|p(z_i)) \\ &\equiv \mathcal{L}_o(u_i;\theta,\phi), \end{aligned} \tag{5}$$

where q_ϕ is the variational distribution. The q_ϕ is set to be a fully factorized diagonal Gaussian distribution. In the FVAE, the mean and standard deviation of $q(z_i)$ are approximated by a MLP model $g_\phi(\cdot)$ parametrized by ϕ:

$$\begin{aligned} g_\phi(u_i) &= [\mu_\phi(u_i), \sigma_\phi(u_i)] \in \mathbb{R}^{2D}, \\ &= [g_\mu(g_{L_e}(...g_2(g_1(u_i)))), g_\sigma(g_{L_e}(...g_2(g_1(u_i))))], \end{aligned} \tag{6}$$

where g_μ, g_σ and g_l denote the mapping functions for the mean, standard deviation and the *l-th* hidden neural layer.

The evidence lower bound (ELBO) of Eq. (5) is composed of two terms: the first term is the reconstruction error and the second term is the regularization. We introduce a hyper-parameter β to control the strength of regularization, and apply annealing to tune the parameter. In addition, since the full features are grouped by multiple fields, we introduce another hyper-parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_K\}$ to balance the reconstruction error of different fields. Combining Eq. (5), we can represent the ELBO as:

$$\begin{aligned} \mathcal{L}(u_i;\theta,\phi) \equiv &\frac{1}{|\alpha|}\sum_k \alpha_k \cdot \mathbb{E}_{q_\phi(z_i|u_i)}[\log p_\theta(F_i^k|z_i)] \\ &- \beta \cdot KL(q_\phi(z_i|u_i)\|p(z_i)), \end{aligned} \tag{7}$$

where α_k is a hyper-parameter controlling the weight of reconstruction loss for field k. $|\alpha| = \sum_k |\alpha_k|$ is the normalized weight to ensure that the sum of weights is equal to 1. β controls the weight of regularization. FVAE can be learned by optimizing Eq. (7) with the Stochastic Gradient Descent (SGD) once the *reparametrization trick* [24], [52] is used.

C. Learning with big data

The complexity of the optimizing Eq. (7) is $O(J \times (D_{L_e} + D_{L_d}))$, where D_{L_e} and D_{L_d} denote the number of neurons for the first encoder hidden layer and the last decoder hidden layer. As the input (output) layer in the encoder (decoder) has the same dimension as the feature space, scales of D_{L_e} and D_{L_d} are non-trivial. This means that it is infeasible to deploy in the industrial scenario, as training the model with billion-scale users and features will require unacceptable time. Simply using the SGD for training will hinder the fast product iterations in the Internet industry. In addition, the users' features are constantly changing and increasing with new data sources available, the traditional VAEs cannot satisfy such requirements. To mitigate these issues, we propose to reduce the training complexity from both input and output perspective, by using dynamic hash tables, the batched softmax function, and the feature sampling, as detailed below.

1) Dynamic Hash Table: The computational complexity of the encoder highly depends on the first layer of DNNs. In order to reduce the complexity of the process, we create a dynamic hash table to achieve the fast look-up [53]. Specifically, we create a hash table for each feature field to map the feature IDs into different weight matrices. Given the ID of a feature field, the encoder looks up the corresponding weights for each feature. The output of the first layer is then obtained by summing all embedding outputs, which is equivalent to the original output of the first layer.

When dealing with the unknown feature ID, the embedding weights of this ID are randomly initialized and pushed into the hash table. The key will be dynamically incremented when a new key is encountered. In the training process, we initialize the hash tables with empty keys, and they will grow dynamically with the training. This addresses the feature growth problem as well as the collision compared to feature hashing, which allows the encoder to handle inputs with different size.

Generally, for user i, $N_i = \sum_k N_i^k \ll J$ holds, as the input data is sparse and the complexity of looking up a weight is O(1). By using this method, the complexity of Eq. 7 reduces to $O(\overline{N} \times D_{L_e} + J \times D_{L_d})$, where $\overline{N}$ denotes the average number of observed features.

2) Batched Softmax: We also reduce the complexity of the output layer by using the batched softmax function. The legacy softmax function requires to compute the values over all neurons of each feature field. This is quite expensive when the output space is large. To mitigate this issue, we substitute the softmax function with the new batched softmax. This can be viewed as a special case in sampled softmax [54]. The batched softmax function only samples features that include at least one user to compute the softmax logits in a training batch. Specifically, in a training batch, we first select a subset of features which is included by at least one user of the batch. Subsequently, we calculate the output of softmax logits, and approximate probability by a normalized function. We denote $\overline{N_b}$ as the mean of the number of observed features in a batch. The batched softmax enables to reduce the complexity of Eq. 7 to $O(\overline{N} \times D_{L_e} + \overline{N_b} \times D_{L_d})$. This complexity reduction is significant, as the feature space is usually sparse and follows a power-law distribution, i.e., $\overline{N} < \overline{N_b} \ll J$. This means that only a minority of features appear frequently in the full feature set.

3) Feature Sampling: Although the batched softmax can reduce the complexity, training the proposed model on large-scale data remains time-consuming. We, therefore, propose feature sampling to further reduce the training complexity. As certain feature fields are super sparse (e.g. topic tag), performing computation over these fields becomes unnecessary. We propose an inter-batched sampling strategy for the sparse fields to address this problem. Specifically, given a field of feature set, we first employ the batched softmax function to filter those features with no users included. Subsequently, we sample the remaining feature with a probability r. By doing this, the feature size of the current batch is further reduced. Although some long-tail features may be missing in the training process, a suitable sampling rate r can improve the generalization of the proposed method. Note that this feature sampling is only used in the training process. The overall training process is shown in Algorithm 1.

D. Real-world Deployment

The overall framework of the FVAE based embedding system for real applications is shown in Figure 2. The framework includes three modules: a data construction module, an offline module, and an online module.

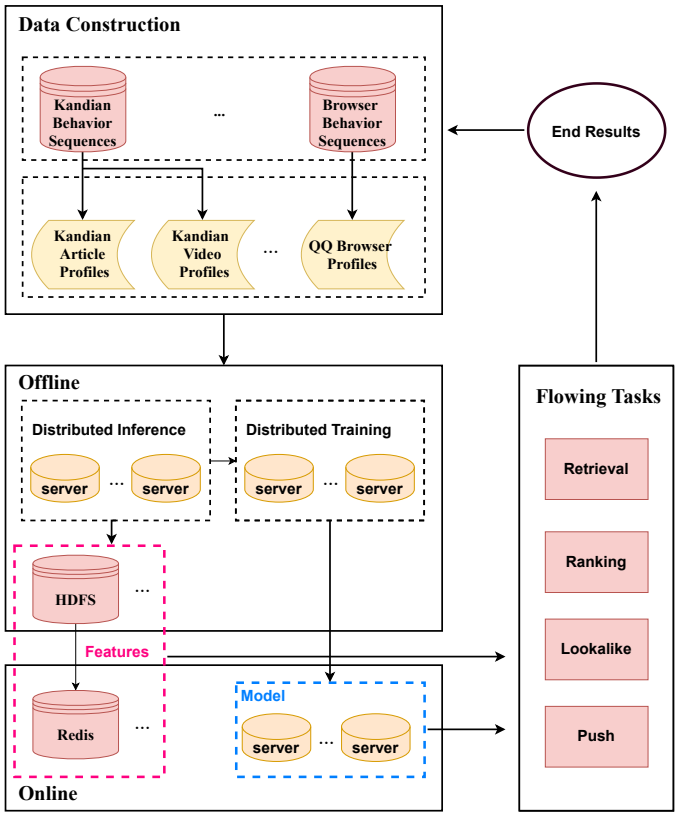

Fig. 2. The framework of real applications.

- **Data construction module.** This module constructs user profiles from original server logs, such as Kandian [2] and QQ Browser[3]. In the system, logs collected from different sources are projected to different fields of features, and subsequently fed into the offline module.
- **Offline module.** This module includes offline training, inference and data storage structures. We first train our FVAE with multi-field user profiles passed from data construction module, then infer the embeddings of each user. The embeddings are saved in the storage system, e.g., HDFS, and will be used for downstream tasks.
- **Online module.** It contains model serving Proxy and high performance cache, e.g., Redis, so as to adapt to different applications. The model inference is deployed on the model serving proxy. High performance cache enables faster user embedding look-up for different applications.

V. Experiment

In this section, we show experiments conducted to answer the following research questions:

[2]https://sdi.3g.qq.com/v/2019111020060911550
[3]https://browser.qq.com

ALGORITHM 1: Training a FVAE.

Input:
$\mathbb{U}$: Feature matrix.
B: Batches for training.
r : Sampling ratio.
randomly initialize θ, ϕ;
while *not converge* **do**
  for b $\in\{1, ..., B\}$ **do**
    Sample a batch of users $\mathcal{U}_b$
    for k $\in \{1, ..., K\}$ **do**
      get unique feature set $\mathcal{F}^k$ for field k
      if *field k should sampling* **then**
        random sampling sampled feature set $\mathcal{F}^k_{sam}$ form $\mathcal{F}^k$ with sampling rate r
      else
        $\mathcal{F}^k_{sam} = \mathcal{F}^k$
      end
    end
    for i $\in \mathcal{U}_b$ **do**
      Sample ϵ and compute z_i via reparametrization trick
      for $k \in \{1, ..., K\}$ **do**
        Compute the log-likelihood for field k with z_i and $\mathcal{F}^k_{sam}$
      end
      Combine the log-likelihood from all fields
      Compute noisy gradient ∇_θ and ∇_ϕ with z_i
    end
    Average noisy gradients from the batch
    Use back propagation to optimize model parameters θ, ϕ.
  end
end

- **RQ1:** Can our proposed FVAE outperform the state-of-the-art methods for downstream tasks including million and billion datasets?
- **RQ2:** How do the key hyper-parameter settings affect on the performance of our proposed FVAE? How does the proposed sampling strategy help improve the performance?
- **RQ3:** Can the proposed FVAE learn user representation efficiently in large scale data?
- **RQ4:** How to deploy the FVAE in real-world applications?

A. Experiment Settings

1) Baseline Methods: We compare our proposed method with several existing methods, including Principal Component Analysis (PCA), Latent Dirichlet Allocation (LDA) and different VAEs. Specifically,

- **PCA [55] .** It is one of the most wildly used method for dimension reduction. This method projects data to a lower dimensional space by Singular Value Decomposition. It is applied in feature matrix $\mathbb{U}$ to embed each user as a latent k-dimensional vector.
- **LDA [56] .** It is a topic model used for discovering latent topics from a collection of vectors. We implement in a batch update form. By applying LDA, the feature matrix $\mathbb{U}$ is decomposed into a user-topic matrix and a topic-feature matrix. We consider the $i-th$ vector of user-topic matrix as the representation of user i.
- **Denoising Autoencoders with a multinomial likelihood (Mult-DAE) [8].** The Denoising Autoencoder (DAE) reduces overfitting by training AEs with corrupted data. Mult-DAE is a variant of DAE by using a multinomial likelihood for the data distribution. We follow a common setting to apply dropout in the input layer.
- **VAE with a multinomial likelihood (Mult-VAE) [8] .** Mult-VAE models the distribution with a multinomial likelihood, which is similar as the Mult-DAE. For the Mult-DAE and Mult-VAE, we consider the output values of the encoder structure as the representation of each user.
- **Item2Vec [42] .** Item2Vec is a state-of-the-art embedding model for learning item representations. Its variants are wildly used in real-world applications. In this work, each feature is regarded as an item fed into a skip-gram model to learn a latent vector. The representation vector of a user is then aggregated by the average of the features. We apply the negative sampling in the training process.
- **RecVAE [23].** The RecVAE introduces a new composite prior distribution for the latent distribution z. It also use a user-specific β in the annealing to improve the performance of VAE-based models.
- **Job2Vec [57].** The Job2Vec is a novel multi-view representation learning model for representing Job Title Benchmarking. It has used for reference with our proposed multi-field user profiles.

2) Datasets: In order to verify the effectiveness and efficiency of the proposed method, we conduct the experiments on several industrial large-scale datasets collected by the applications of Tencent. We show details of all datasets in the Table I.

- **Kandian (KD).** We construct this profile dataset based on *Kandian* users' behaviors. Users are classified into 4 fields, namely first channel IDs (ch_1), second channel IDs (ch_2), third channel IDs (ch_3) and tags. For each user, we construct each field using his top 512 weights with the highest values.
- **QQ Browser (QB).** This dataset has similar characteristics with KD dataset and is collected by *QQ Browser* users. It also contains the same 3 channel-level IDs and tags.
- **Short Content (SC).** We collect this million-scale dataset from *Kandian Short Content* users to compare the baseline methods with the proposed FVAE. This dataset contains the top 128 highest weighted tags for each user.

3) Parameter Settings: We employ two wildly used metrics, *Area Under the ROC Curve* (AUC) and *mean Average Preci-*

TABLE I
STATISTICS OF THE BENCHMARK DATASETS.

Dataset	#Users	#Fields	$\overline{N}$	J
KD	0.65 billion	4	193.68	1.32 billion
QB	0.33 billion	4	123.69	0.52 billion
SC	1 million	4	211.16	130159

sion (mAP) [12], [37], [58], [59] to evaluate the performance of the FVAE. We partition the dataset into two parts, where 80% is used for training and the rest 20% for testing. We apply the feature sampling strategy in tag field, and test the sampling ratio $r \in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$. For PCA and LDA, we set the latent dimension D to 128. For Mult-DAE, Mult-VAE, RecVAE and FVAE, we uniformly set the encoder structure to $256 \rightarrow 128$, and set the decoder structure to $128 \rightarrow 256$.

4) Summary of this section: In order to answer the proposed research questions, we conduct several experiments in this section.

Detailedly, for answer **RQ1**, we choose two popular tasks, namely Reconstruction and Tag Prediction to compare the FVAE with baselines. Our experiments perform at different scales real-word datasets. Results are reported in Section V-B and Section V-C. We exclude the Mult-DAE, the Mult-VAE, the RecVAE and the Job2Vec in the experiments on billion-scale datasets for their scalability issues. We also conduct a case study to visualize the learned user representations in a billion-scale dataset.

To answer **RQ2**, we conduct experiments to test the key hyper-parameters, α, β and the sampling strategy of the FVAE in Section V-D.

To answer **RQ3**, we evaluate the efficiency of FVAE in Section V-E. We first compare the training speed between FVAE and Mult-VAE on three real-word datasets. Then we test the scalability of FVAE on several synthetic datasets. We finally examine the performance in distributed computing of the FVAE.

To answer **RQ4**, we conduct an online A/B testing in a lookalike system detailed in Section V-F.

B. Results on Million-scale Dataset

1) Reconstruction: Data reconstruction is a fundamental task for representation learning [60], [61]. It aims at reconstructing the data with a learned representation and minimizing the reconstruction error. We evaluate the reconstruction performance by AUC and mAP. We show the results in Table II. We can observe that

- Our proposed FVAE outperforms all baseline methods on both metrics from the perspective of field performance. This demonstrates that the FVAE is able to reconstruct feature ranking orders for users with high performance in ch1, ch2, ch3 and tag.
- FVAE obtains slightly lower AUC compared to the Mult-VAE and RecVAE. This is because the FVAE models each field with an independent multinomial distribution. Therefore, the outputs of different fields are measured by different standards thus cannot be compared directly. However, field-based metrics are more important than the overall metric, as field-based application is used more frequently in real-world systems.

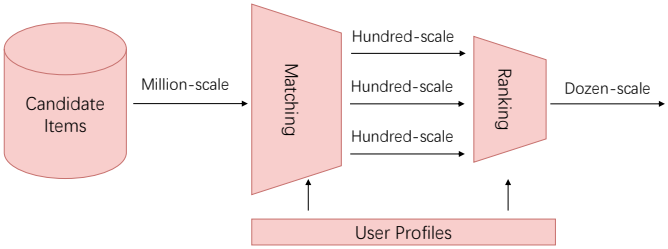

Fig. 3. A typical framework of industrial recommendation systems. The matching stage aims at finding satisfied items from millions of candidates, and feeding them to the ranking stage.

2) Tag Prediction: Tag prediction is a crucial task in real recommender systems and it plays an essential role in the matching stage [14], [62]. In an industrial setting, the whole pipeline is usually consisted of several stages (shown in fig. 3), including the matching stage and the ranking stage [13], [63]. The matching stage consists of several different models or strategies, where Tag-based matching is one of the most popular one. It recalls candidates by matching the same or similar tag observed in the item and user profiles. We employ FVAE for tag prediction as a downstreaming recommender system task to evaluate its performance.

Specifically, we use the model to predict the tag for each user. To this end, we first train FVAE in the training set. Subsequently, for each held-out user of the test set, we choose features of Ch1, Ch2 and Ch3 as the fold-in set to learn the important user-level representation and predict each tag. We pick the observed tags as the positives and randomly select unobserved tags as the negatives with the same number of the observed tags. We use AUC and mAP to evaluate the performance of the prediction, and report the results in Table III. We can observe that:

- Our proposed methods outperform all baseline methods in both metrics. Specifically, FVAE obtains $3.61\% \sim 26.83\%$ (average 10.68%) improvement over baseline methods in AUC and $4.02\% \sim 21.63\%$ (average 12.40%) in mAP. This demonstrates that our model can predict the probability of tags with better accuracy.
- Compared to FVAE and Mult-VAE, the FVAE significantly outperforms Mult-VAE in the task. This is because that FVAE uses an independent multinomial likelihood for the data distribution of each field. Therefore, FVAE can reduce the feature ordering bias between different fields.

C. Results on Billion-scale Dataset.

1) Tag Prediction: We conduct experiments on the KD and QB dataset to evaluate the performance of methods with

TABLE II
THE AUC AND MAP OF RECONSTRUCTION TASK ON SHORT CONTENT DATASET. BOLD VALUES INDICATE THE BEST RESULTS.

	AUC					mAP				
	Overall	Ch1	Ch2	Ch3	Tag	Overall	Ch1	Ch2	Ch3	Tag
PCA	0.8650	0.8761	0.9677	0.8711	0.8437	0.5329	0.9548	0.8225	0.7216	0.3498
LDA	0.8860	0.8249	0.9532	0.8686	0.8762	0.4243	0.8658	0.6801	0.6063	0.2568
Item2Vec	0.8917	0.8825	0.9534	0.8766	0.8554	0.5343	0.8916	0.8033	0.7151	0.3699
Mult-DAE	0.8811	0.8755	0.8434	0.8629	0.7789	0.4689	0.8522	0.7792	0.6628	0.3119
Mult-VAE	0.9592	0.9418	0.9573	0.9713	0.9447	0.5940	0.9582	0.7897	0.7296	0.4035
RecVAE	**0.9593**	0.9315	0.9559	0.9629	0.9473	0.5846	0.9487	0.8297	0.7351	0.3955
Job2Vec	0.9184	0.8858	0.8592	0.9023	0.8588	0.3784	0.8640	0.6063	0.6308	0.3168
FVAE	0.9583	**0.9660**	**0.9769**	**0.9832**	**0.9782**	**0.6171**	**0.9736**	**0.8460**	**0.7798**	**0.4439**

TABLE III
THE AUC AND MAP OF TAG PREDICTION FOR ALL METHODS.

	AUC	mAP
PCA	0.7602	0.8731
LDA	0.8541	0.8662
Item2Vec	0.8793	0.8825
Mult-DAE	0.8775	0.8124
Mult-VAE	0.9305	0.9367
RecVAE	0.9210	0.9129
Job2Vec	0.8987	0.8011
FVAE	**0.9641**	**0.9744**

billion-scale data. Experimental setting in this subsection is similar to the Section V-B2. We exclude the results of Mult-DAE and Mult-VAE in this section as they require considerable amount of training time which cannot be tolerated in industry. We report the results in Table IV. Observe that:

- For each sampling ratio r, FAVE outperforms all baselines in terms of AUC and mAP by a large margin. This proves that FVAE can keep the effectiveness of performance on billion-scale datasets.
- We note that although the optimal value of r is different for various datasets, the gap of performance is subtle. As our method can be trained much faster, our method is still the best solution among all approaches.

TABLE IV
THE AUC AND MAP OF TAG PREDICTION ON THE BILLION-SCALE DATASETS.

	KD		QB	
	AUC	mAP	AUC	mAP
PCA	0.7321	0.8508	0.7017	0.7439
LDA	0.8424	0.8718	0.7901	0.8406
Item2Vec	0.8537	0.8699	0.8569	0.8872
FVAE(r=0.01)	0.9525	0.9699	0.9435	0.9356
FVAE(r=0.05)	0.9630	0.9703	**0.9441**	**0.9453**
FVAE(r=0.1)	**0.9632**	**0.9712**	0.9432	0.9452

2) Visualization: We also conduct a case study to verify the learned embedding of FVAE on KD dataset. We randomly select 1000 users from 3 topics. Then, we follow a common experimental setting [58] mapping those vectors into the 2-D space with t-SNE.

Figure 4 shows the visualization of FVAE embeddings. Points with the same shape and color belong to the same topic. We can see that almost all topics can be intuitively distinguished and similar users are closer to each other than dissimilar ones in the low-dimensional space. This shows that FVAE can cluster data with different topics with clear boundaries.

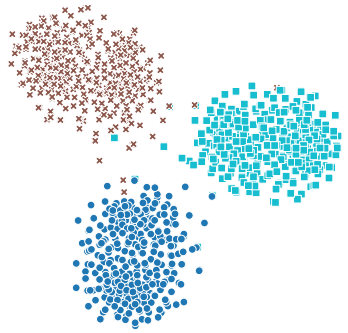

Fig. 4. 2-D visualization of embeddings of users in FVAE. Each point indicates one user. Different shapes and colors points represent different topics. FVAE forms most of the points belonging to different topics to different clusters.

D. Parameter Sensitivity

1) Sampling Strategy: Sampling unique features in epochs is an efficient way to speed up the training process. We apply different sampling strategies in our model to evaluate the performance. We compare the difference between three sampling distributions [16]:

- **Uniform**. This strategy ignores the frequency information in each batch and select features by a Uniform distribution.
- **Frequency**. This strategy samples features by their frequency rate of the current batch.
- **Zipfian**. In a training batch, this strategy first ranks features by their decreasing frequency of this batch, then samples features by an approximately Zipfian distribution.

We report the results in Figure 5. For all compared sampling strategy, we test sampling ratio r in $\{0.2, 0.4, 0.6, 0.8\}$. Note that the Uniform sampling outperforms other sampling strategies in all sampling ratios, which validates the effectiveness of the proposed sampling strategy. We also notice that the performance of the FVAE does not monotonically increase $w.r.t$ the sampling ratio r. This is because the sampling

strategy with lower r has greater possibilities of filtering long-tail features in the training process. Similar results can also be observed in [64].

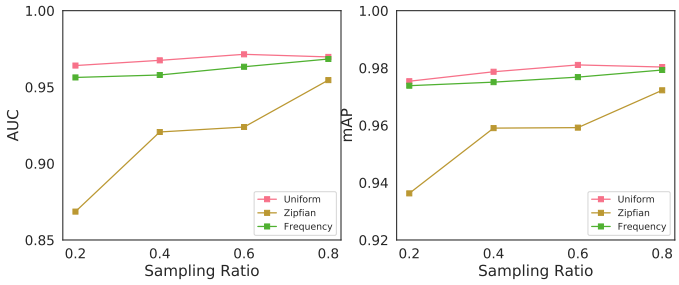

Fig. 5. The effect with different sampling strategies. The uniform strategy achieve best performance in both AUC and MAP *w.r.t* different sampling rate.

In addition, we plot the AUC score in the validation set $w.r.t.$ the training time and different sampling ratios in Figure 6. We can see that $r = 0.1$ achieves the best performance and speed up the training nearly $4\times$ compared to the $r = 0.2$. Using $r = 0.01$ achieves similar AUC score with $r = 0.1$, but the AUC score improves slowly. This means that a proper r can not only improve the performance but also reduce the training time.

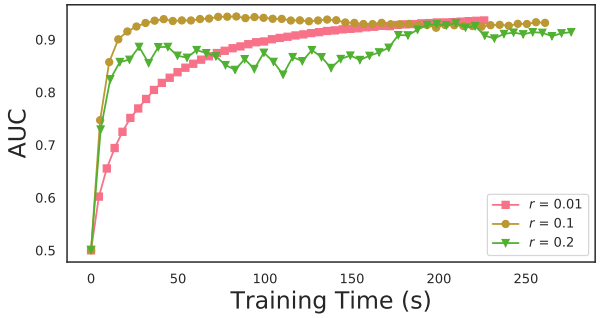

Fig. 6. AUC $w.r.t$ training time with different sampling rate r in tag prediction task. $r = 0.1$ achieves the best performance and consumes shorter training time.

2) The Effect of α : Recall that α_i is a hyper-parameter that controls the weight of reconstruction loss in fields, and i is the field index. The size of α equals the number of feature fields. We study the sensitivity of α_i by fixing the values of others to 1. The result is shown in Figure 7. It is observed that different fields exhibit different patterns in terms of tag prediction performance. Specifically, for α_{Ch1} and α_{Ch2}, the optimal values change from 0.1 to 1. When the weight become too large, the information expressed by other fields may be insufficient. α_{Ch3} and α_{Tag} are more insensitive, this is because those fields only contain the partial information of other fields. Moreover, the results show that the αs maintain high performance in an extensive range (from 0.001 to 10), outperforming the traditional Mult-VAE even in the worse setting. Therefore, a common-used hyper-parameters search algorithm, i.e., Random search [65], or a simple experienced setting, i.e., setting all $\alpha = 1$, could make the FVAE achieve remarkable performance.

3) The Effect of β : β is another hyper-parameter that controls the peaking weight of KL annealing. We test different β values: $\{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$.It could be early tuned by the *early stopping*. Results in Figure 8 show that β can improve the performance of FVAE if a proper value is chosen. In the SC dataset, the β makes FVAE more robust since the annealing is applied [8]. Overall, we argue that both α and β are valuable and easy to tune.

E. Efficiency and Scalability

1) Training Speed: To compare the efficiency of different methods, we report the training speed of Mult-VAE[4] and FVAE on all datasets in Table V. All experiments are conducted in a single machine with an Intel 6133 CPU and 256G memory. The batch size is set as 512 for these two methods, and the sampling r is set as 0.1. Results in Table V show that FVAE can substantially improve the throughput especially in the large-scale datasets.

Specifically, FVAE acquires $56\times$ speedup in million-scale datasets, $3085\times$ speedup in the KD dataset, and $4020\times$ speedup in QB dataset. This is because that the number of max feature size are over million, such that the batched softmax function and the batched feature sampling strategy in FVAE can significantly reduce the computation cost of the output layer.

TABLE V
THROUGHPUT (SAMPLES/SEC) OF MULT-VAE AND FVAE ON THREE DATASETS.

Dataset	Mult-VAE	FVAE	Improvement
SC	39.88	2257.92	$56\times$
KD	0.72	2739.85	$3085\times$
QB	0.71	2872.13	$4020\times$

2) Scalability: To analyze the scalability of FVAE, we conduct experiments on synthetic data. We generate random samples with large sparse features by Barabási–Albert preferential attachment model [66]. We fix the average feature size (to 200) or max feature size (to 10^5) in the experiments. We report the running time in Figure 9. Results show that the running time of our method grows linearly with respect to the average feature size. This demonstrates the scalability of FVAE. In addition, the running time changes very slightly with respect to the max feature size, which means that this will not affect the efficiency of FVAE when new features are continually added.

3) Speedup via Distributed Computing: In order to evaluate the performance of the proposed method in distributed computing, we report the speedup ratio on the KD dataset $w.r.t$ training servers. All experiments run on the Tencent Cloud servers. We vary the number of servers from 3 to 12 for distributed computing machines. Figure 10 shows that

[4]For KD and QB dataset, all features are mapped to a 20bit space by *feature hashing* since original billion-scale size is too large for Mult-VAE

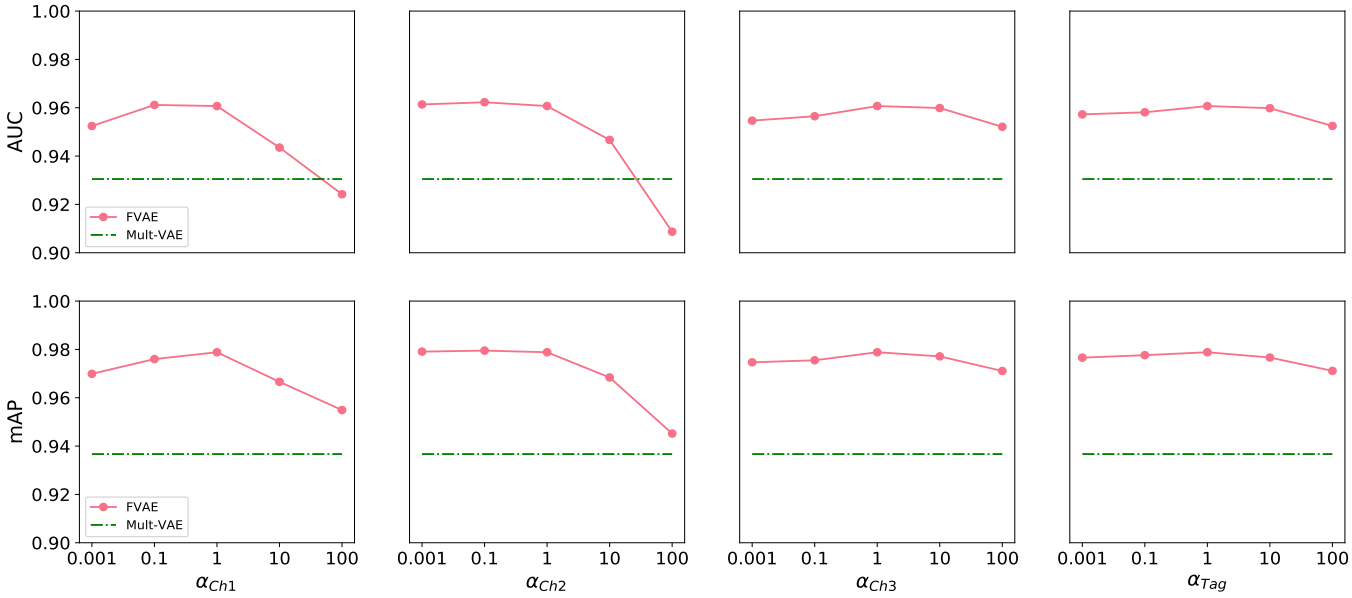

Fig. 7. AUC and mAP $w.r.t$ α in tag prediction on SC. While α shows divergent sensitivities for different fields, it keeps high performance in an extensive range.

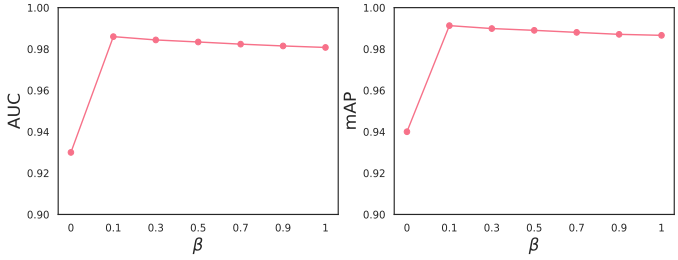

Fig. 8. AUC and mAP $w.r.t$ β in tag prediction task on SC. A simple positive β could improve the performance of the proposed FVAE.

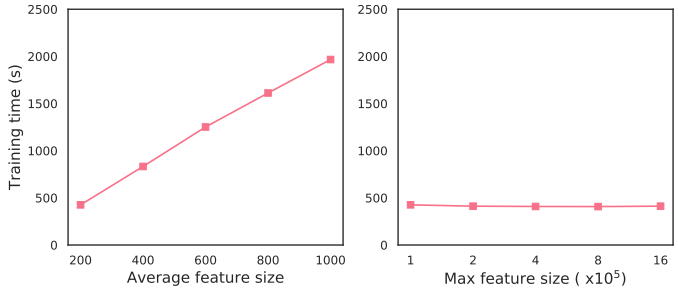

Fig. 9. Scalability comparison of the FVAE. FVAE scales linearly with the number of average feature size, and keeps stable *w.r.t.* the max size of features.

the speedup ratios increase almost linearly with the number of servers, demonstrating that the FVAE can achieve high efficiency in the distributed environment.

F. Online Application

The look-alike model has been already deployed in recommendation systems to extend audiences with high quality long-tail contents. Meanwhile, user representation learning plays a key role in a look-alike system [7]. In order to evaluate the performance of FVAE in the real application, we conduct an online A/B testing in *QQ Browser Uploader Recommendation* based on a look-alike system. This offers lists of uploaders to users and attracts those users to follow uploaders that they are interested in. We first employ a FVAE to learn user representation, then generate account emebeddings by using average pooling to merge all followed users. Finally, we recall similar accounts by the L2 similarity for candidates. Note that here we employ the skip-gram model [41] as the baseline to learn user representations. We evaluate the performance of the models with the number of following clicks and the following online metrics:

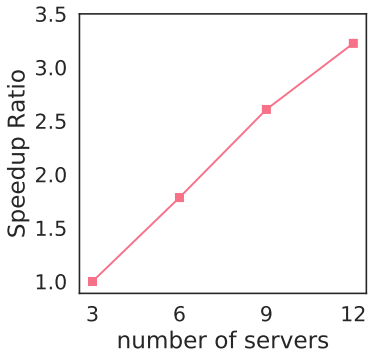

Fig. 10. The speedup of FVAE via distributed computing. FVAE scales linearly with the number of computing resources.

- **Like**. It is considered as a more important positive feedback than click [67]. We chose two metrics, $\#Like$ and Avg. Like $= \frac{\#Like}{\#users_{liked}}$, to evaluate different models

more comprehensively. The $\#Like$ denotes the total number of Like and $\#users_{liked}$ means the number of users who like contents at least one time.

- **Share**. This is another strong feedback to measure the goodness of exhibited contents. We compared different models by $\#Share$ and Avg. Share= $\frac{\#Share}{\#users_{shared}}$, where $\#Share$ denotes the total number of Shares and $\#users_{shared}$ represents the number of users who share contents at least one time.

TABLE VI
RELATIVE CHANGES IN METRICS IN ONLINE A/B TEST.

Metric	Change
#Following Click	$+7.92\%$
$\#Like$	$+1.31\%$
Avg. Like	$+1.16\%$
$\#Share$	$+1.90\%$
Avg. Share	$+2.12\%$

We show the results in Table VI. We can see that the Following Click gains a great improvement, which means that the proposed FVAE generates a better user representation. (Avg.) Like and (Avg.) Share also acquire a good increase. This shows that extended users are truly interested in those recalled accounts as well.

VI. CONCLUSION

In this paper, we propose a novel model FVAE to learn users' representations from large-scale, high-dimensional, and multi-field user features. We model input data with several independent multinomial distributions, bijectively mapping to different fields. This effectively extends Mult-VAE to multi-field features. We utilize dynamic hash tables, the batched softmax, and a feature sampling strategy to improve the efficiency of FVAE during training. We further evaluate the performance of our method on several tasks, including reconstruction and tag prediction and recommender systems. Offline experiments on 3 large-scale datasets show that our method can learn user embedding efficiently and effectively. Our FVAE achieves 12.40% relative improvement on AUC score than state-of-the-art baselines in tag prediction task on a billion-scale dataset. We also conduct experiments on online lookalike systems. Results show that the FVAE outperforms baseline methods in all metrics, demonstrating that the embedding learned by our method can effectively improve the performance of online applications.

REFERENCES

[1] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2566–2581, 2016.

[2] S. Li and H. Zhao, "A survey on representation learning for user modeling," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020, pp. 4997–5003.

[3] L. Gong, X. Feng, D. Ye, H. Li, R. Wu, J. Tao, C. Fan, and P. Cui, "OptMatch: Optimized matchmaking via modeling the high-order interactions on the arena," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2300–2310.

[4] D. Kang, A. Balakrishnan, P. Shah, P. Crook, Y.-L. Boureau, and J. Weston, "Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1951–1961.

[5] H. Gong, Q. Zhao, T. Li, D. Cho, and D. Nguyen, "Learning to profile: User meta-profile network for few-shot learning," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2469–2476.

[6] S. deWet and J. Ou, "Finding users who act alike: transfer learning for expanding advertiser audiences," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2251–2259.

[7] Y. Liu, K. Ge, X. Zhang, and L. Lin, "Real-time attention based look-alike model for recommender system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2765–2773.

[8] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 689–698.

[9] K. Luo, H. Yang, G. Wu, and S. Sanner, "Deep critiquing for vae-based recommender systems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1269–1278.

[10] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 43–50.

[11] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: learning the weight of feature interactions via attention networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3119–3125.

[12] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[13] C. Li, Z. Liu, M. Wu, Y. Xu, H. Zhao, P. Huang, G. Kang, Q. Chen, W. Li, and D. L. Lee, "Multi-interest network with dynamic routing for recommendation at tmall," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2615–2623.

[14] E. Quintanilla, Y. Rawat, A. Sakryukin, M. Shah, and M. Kankanhalli, "Adversarial learning for personalized tag recommendation," *IEEE Transactions on Multimedia*, vol. 23, pp. 1083–1094, 2021.

[15] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1113–1120.

[16] W. Li, "Random texts exhibit zipf's-law-like word frequency distribution," *IEEE Transactions on information theory*, vol. 38, no. 6, pp. 1842–1845, 1992.

[17] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *International conference on machine learning*. PMLR, 2016, pp. 1727–1736.

[18] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *International conference on machine learning*. PMLR, 2017, pp. 3881–3890.

[19] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NIPS Workshop Bayesian Deep Learning*, 2016, pp. 1–3.

[20] D. Tang, D. Liang, T. Jebara, and N. Ruozzi, "Correlated variational auto-encoders," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6135–6144.

[21] A. Sankar, X. Zhang, A. Krishnan, and J. Han, "Inf-VAE: A variational autoencoder framework to integrate homophily and influence in diffusion prediction," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 510–518.

[22] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 600–608.

[23] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko, "Recvae: A new variational autoencoder for top-n recommendations with implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 528–536.

[24] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations(ICLR)*, 2014, pp. 1–14.

[26] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations(ICLR)*, 2017, pp. 1–22.

[27] M. Pavlovski, J. Gligorijevic, I. Stojkovic, S. Agrawal, S. Komirishetty, D. Gligorijevic, N. Bhamidipati, and Z. Obradovic, "Time-aware user embeddings as a service," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3194–3202.

[28] Q. Zheng, G. Liu, A. Liu, Z. Li, K. Zheng, L. Zhao, and X. Zhou, "Implicit relation-aware social recommendation with variational autoencoder," *World Wide Web*, pp. 1–16, 2021.

[29] X. Yu, X. Zhang, Y. Cao, and M. Xia, "VAEGAN: A collaborative filtering framework based on adversarial variational autoencoders." in *IJCAI*, 2019, pp. 4206–4212.

[30] Z. Xie, C. Liu, Y. Zhang, H. Lu, D. Wang, and Y. Ding, "Adversarial and contrastive variational autoencoder for sequential recommendation," in *Proceedings of the Web Conference 2021*, 2021, pp. 449–459.

[31] I. Porteous, A. Asuncion, and M. Welling, "Bayesian matrix factorization with side information and dirichlet process mixtures," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010, pp. 563–568.

[32] S. Jing and S. Li, "Contextual collaborative filtering for student response prediction in mixed-format tests," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 8095–8096.

[33] S. Li and Y. Fu, "Robust representations for response prediction," in *Robust Representation for Data Analytics*. Springer, 2017, pp. 147–174.

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

[35] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[36] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 880–887.

[37] R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. S. Bhowmick, "Scaling attributed network embedding to massive graphs," *Proceedings of the VLDB Endowment*, vol. 14, no. 1, p. 37–49, Sep. 2020.

[38] H. Liu, J. Han, Y. Fu, J. Zhou, X. Lu, and H. Xiong, "Multi-modal transportation recommendation with unified route representation learning," *Proceedings of the VLDB Endowment*, vol. 14, no. 3, pp. 342–350, 2020.

[39] W. Zeng, G. Fan, S. Sun, B. Geng, W. Wang, J. Li, and W. Liu, "Collaborative filtering via heterogeneous neural networks," *Applied Soft Computing*, p. 107516, 2021.

[40] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[41] O. Barkan and N. Koenigstein, "ITEM2VEC: neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[42] O. Barkan, A. Caciularu, O. Katz, and N. Koenigstein, "Attentive item2vec: Neural attentive user representations," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3377–3381.

[43] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 839–848.

[44] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[45] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2333–2338.

[46] X. Yi, J. Yang, L. Hong, D. Z. Cheng, L. Heldt, A. Kumthekar, Z. Zhao, L. Wei, and E. Chi, "Sampling-bias-corrected neural modeling for large corpus item recommendations," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 269–277.

[47] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International conference on artificial neural networks*. Springer, 2011, pp. 44–51.

[48] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1–4.

[49] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, "Parameter-efficient transfer from sequential behaviors for user modeling and recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1469–1478.

[50] F. Yuan, G. Zhang, A. Karatzoglou, J. Jose, B. Kong, and Y. Li, "One person, one model, one world: Learning continual user representation without forgetting," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 696–705.

[51] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[52] Q.-T. Truong, A. Salah, and H. W. Lauw, "Bilateral variational autoencoder for collaborative filtering," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 292–300.

[53] H. Rong, Y. Wang, F. Zhou, J. Zhai, H. Wu, R. Lan, F. Li, H. Zhang, Y. Yang, Z. Guo, and D. Wang, "Distributed equivalent substitution training for large-scale recommender systems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 911–920.

[54] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015, pp. 1–10.

[55] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[56] M. Hoffman, F. R. Bach, and D. M. Blei, "Online learning for latent dirichlet allocation," in *advances in neural information processing systems*. Citeseer, 2010, pp. 856–864.

[57] W. Wang, F. Xia, J. Wu, Z. Gong, H. Tong, and B. D. Davison, "Scholar2vec: vector representation of scholars for lifetime collaborator prediction," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 3, pp. 1–19, 2021.

[58] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations(ICLR)*, 2018, pp. 1–13.

[59] G. Fan, B. Geng, J. Tao, K. Wang, C. Fan, and W. Zeng, "PPPNE: personalized proximity preserved network embedding," *Neurocomputing*, 2021.

[60] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[61] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[62] L. Wu, R. Jin, and A. K. Jain, "Tag completion for image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 3, pp. 716–727, 2012.

[63] J. Chen, Z. Gong, Y. Li, H. Zhang, H. Yu, J. Zhu, G. Fan, X.-M. Wu, and K. Wu, "Meta-path based neighbors for behavioral target generalization in sequential recommendation," *IEEE Transactions on Network Science and Engineering*, 2022.

[64] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th international conference on world wide web*, ser. WWW '15, Republic and Canton of Geneva, CHE, 2015, p. 278–288.

[65] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[66] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[67] L. Tang, B. Long, B.-C. Chen, and D. Agarwal, "An empirical study on recommendation with multiple types of feedback," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 283–292.