# Conditioning of variational autoencoder by user traits for item recommendation

Yuuki Tachioka
tachioka.yuki@core.d-itlab.co.jp
Denso IT Laboratory
Shibuya, Tokyo, Japan

## ABSTRACT

Deep learning-based recommendation algorithms have recently attracted attention due to their effectiveness at processing big data. Methods based on the variational autoencoder (VAE) are particularly promising thanks to their advantage with the data sparsity problem in recommendation tasks. However, because user traits affect the preference of recommended items, to improve the performance of VAE-based recommendation methods, it is necessary to carefully consider user traits. In this paper, we propose a method that conditions the VAE with user trait labels for switching the distributions of a generative model of latent variables. Experiments on a music recommendation task demonstrate that utilizing user trait labels estimated from tweet history leads to an improved performance and that the distribution can be changed depending on the individual traits of users.

## CCS CONCEPTS

• **Information systems** → **Personalization**.

## KEYWORDS

recommendation model, user traits, OCEAN (BigFive) traits, conditional variational autoencoder

**ACM Reference Format:**
Yuuki Tachioka. 2023. Conditioning of variational autoencoder by user traits for item recommendation. In *Proceedings of MuRS: Music Recommender Systems Workshop (MuRS '23).* ACM, New York, NY, USA, 5 pages.

## 1 INTRODUCTION

When using item recommendation systems, it is important to estimate user preferences accurately by considering the preferences of similar users through processes such as collaborative filtering [1, 10]. In addition, personalized recommendation is typically needed in order to fully satisfy all users [5, 6, 11, 21]. For that purpose, it is necessary to consider user traits, because user traits affect the preference of items [2]. Prior studies on music recommendation tasks have shown that the most desirable recommended item will differ depending on the user traits [3, 7, 18] and that user traits significantly affect the performance of the recommendation system [16]. In the field of personality research, various types of user traits have been proposed [4, 17, 22], and the most widely utilized of which is the set of OCEAN traits known as "BigFive" (listed in Table 1) [4, 17]. Traits can be inferred from social media [8, 9] and then analyzed for making better recommendations [15]. For example, for users with low *ope*, it is preferable to recommend items similar to their previously selected items, but for users with high *ope*, it makes more sense to recommend unknown items from various genres. Users with high *agr* tend to select popular items [3].

**Table 1: "BigFive" traits represented as OCEAN.**

| Trait | Abbreviation |
|---|---|
| Openness to experience | ope |
| Conscientiousness | con |
| Extroversion | ext |
| Agreeableness | agr |
| Neuroticism | neu |

Recommendation algorithms based on deep learning have recently been attracting interest [10, 14, 19, 23], including methods based on the use of a denoising autoencoder (DAE) [23]. However, due to the data sparsity problem in recommendation tasks, the DAE tends to be overfitted. To avoid overfitting, multDAE utilizes a multinomial distribution for sampling latent variables [14]. In addition, the variational autoencoder (VAE), which assumes a generative model of latent variables [13] has been more effective for recommendation tasks [14, 19], and MultVAE, which assumes a multinomial distribution for a generative model, has outperformed the conventional DAE [14].

However, because user traits are different from user to user, the distributions of a VAE's generative model will be also different from user to user and heavily dependent on the individual user traits. We therefore propose a method called conditional MultVAE (MultCVAE) in which we condition MultVAE on the basis of user trait labels in order to properly consider the traits of users for personalized recommendation. This represents the first study to utilize user traits for conditioning of VAE for recommendation tasks, although conditional VAE [20] has previously been implemented in recommendation tasks while taking users' activity history into account [12]. In this paper, we investigate the effectiveness of our proposed method on a music recommendation task with user trait labels (BigFive traits) [16] estimated from tweet history [9].

## 2 DAE, MULTDAE, AND MULTVAE

In this section, we provide a brief overview of MultDAE and MultVAE [14], which we utilize as baselines. Figure 1 shows the structures of DAE and VAE. DAE converts the input $\boldsymbol{x}_u \in \mathbb{N}^I$ into $M$-dimensional latent variables $\boldsymbol{z}_u \in \mathbb{R}^M$ by encoder $\phi$, where $\boldsymbol{x}_u$ is

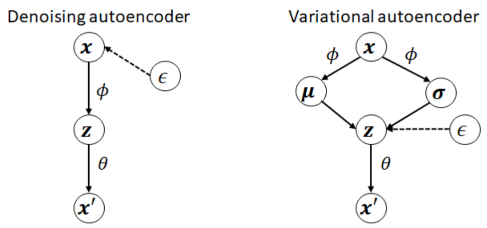

**Figure 1: Denoising autoencoder (MultDAE) and variational autoencoder (MultVAE) for recommendation task.**

the click history or listening history of a user $u$ and is a sparse vector whose dimensions are equal to the number of items $I$. For the items users clicked on or listened to, the corresponding part of $\boldsymbol{x}_u$ is one. From now, $u$ is omitted for simplicity. Decoder $\theta$ obtains the output $\boldsymbol{x}'$ that reproduces its input $\boldsymbol{x}$. To improve the robustness of conversion, noise $\epsilon$ is added to DAE. DAE is optimized by maximizing the objective function $\mathcal{L}$, which is a negative reconstruction error between the inputs and outputs, as $\mathcal{L}(\boldsymbol{x};\theta,\phi) = -||\boldsymbol{x}-\boldsymbol{x}'||^2$. For training, all of $\boldsymbol{x}$ is used. For testing, after a part of $\boldsymbol{x}$ is masked and then input to the autoencoder and to generate recommendation lists $\boldsymbol{x}'$, the masked parts of the recommendation results are used for evaluation. MultDAE is a variant of DAE that samples $M$-dimensional latent variables from a standard Gaussian prior to produce a probability distribution $p$ over $I$ items [14]. MultDAE is optmized by maximizing its likelihood $\mathcal{L}$ as

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = \log p_\theta(\boldsymbol{x}|g_\phi(\boldsymbol{x})), \tag{1}$$

where $g$ is a non-linear function of the model.

On the other hand, VAE does not directly produce latent variables $\boldsymbol{z}$ but estimates the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ of a generative model before latent variables are drawn from an assumed generative model by using a reparametrization trick [13]. MultVAE [14] assumes its generative model as a multinomial distribution. When VAE is trained, the objectives are also a maximization of data likelihood $p(\boldsymbol{x})$ but the processing is quite difficult. As an alternate, evidence lower bound (ELBO) is maximized by using approximate posterior distributions $q_\phi$ [13]. The objective function $\mathcal{L}$ to be maximized is the weighted sum of a negative reconstruction error term and a regularization term by the Kullback Leibler (KL) divergence with a weight parameter $\beta$.

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = \mathbb{E}_{q_\phi(z|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|z)\right] - \beta KL\left(q_\phi(z|\boldsymbol{x})||p(z)\right). \tag{2}$$

## 3 PROPOSED METHOD (MULTCDAE AND MULTCVAE)

In conventional methods, generated recommendation lists $\boldsymbol{x}'$ reflect all user preferences in training data, because all training data are equally used for model training. This degrades the recommendation performance and leads to the need for personalization, as the training data include preferences of users who have different traits from a target user.

Figure 2 shows the structures of the proposed method, which is an extension of the MultDAE and MultVAE in Fig. 1. In the proposed method, we condition encoder $\phi$ and decoder $\theta$ on the basis of user trait labels $\boldsymbol{y}$. This emphasizes the preferences of users whose traits are similar to those of a target user for inferring a recommendation list $\boldsymbol{x}'$. For the encoders, labels $\boldsymbol{y}$ are concatenated

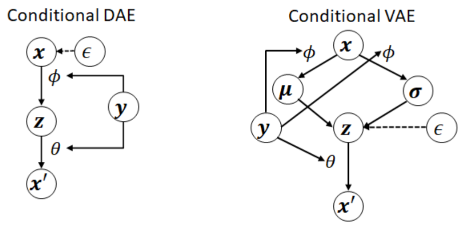

**Figure 2: Proposed conditional autoencoders (MultCDAE and MultCVAE).**

with low-dimensional hidden variables that are converted from the input $\boldsymbol{x}$ by fully-connected layers. For the decoders, labels $\boldsymbol{y}$ are concatenated with latent variables $\boldsymbol{z}$ and then input. This conditioning can be introduced to both VAE and DAE. For MultDAE, the objective function $\mathcal{L}$ is modified as

$$\mathcal{L}(\boldsymbol{x};\theta,\phi,\boldsymbol{y}) = \log p_\theta(\boldsymbol{x}|g_\phi(\boldsymbol{x}|\boldsymbol{y}),\boldsymbol{y}). \tag{3}$$

For MultVAE, the objective function $\mathcal{L}$ is modified as

$$\begin{aligned}\mathcal{L}(\boldsymbol{x};\theta,\phi,\boldsymbol{y}) = {} & \mathbb{E}_{q_\phi(z|\boldsymbol{x},\boldsymbol{y})}\left[\log p_\theta(\boldsymbol{x}|z,\boldsymbol{y})\right] \\ & - \beta KL\left(q_\phi(z|\boldsymbol{x},\boldsymbol{y})||p(z|\boldsymbol{y})\right).\end{aligned} \tag{4}$$

We propose three types of labels $\boldsymbol{y}$ for the conditioning: vec, pos-neg, and one-hot. The BigFive traits are analyzed to obtain the corresponding five-dimensional real-valued scores for each user $u$ as $\boldsymbol{t}_u = [\boldsymbol{t}_u[1],\boldsymbol{t}_u[2],...,\boldsymbol{t}_u[5]] \in \mathbb{R}^5$.

*Real-valued vector of BigFive score (vec).* The first one is the most straightforward feature, which directly uses the BigFive scores as input.

$$\boldsymbol{y}_u = \boldsymbol{t}_u. \tag{5}$$

*Binary label of positive and negative (pos-neg).* The second one is a binarized label (positive and negative) for each trait. After calculating the median of the BigFive scores for each trait $j$, users whose scores are higher than the median are set to positive and otherwise to negative, as

$$\boldsymbol{y}_u[j] = \begin{cases} 1 \text{ if } \boldsymbol{t}_u[j] \geq \text{median}([\boldsymbol{t}_1[j],...,\boldsymbol{t}_U[j]]) \\ 0 \text{ if } \boldsymbol{t}_u[j] < \text{median}([\boldsymbol{t}_1[j],...,\boldsymbol{t}_U[j]]) \end{cases}, \tag{6}$$

where the number of users is $U$.

*One-hot vector of index that takes the maximum of BigFive scores (one-hot).* The third label is a one-hot version of the second one. The most dominant trait is represented by a one-hot vector as

$$\boldsymbol{y}_u[j] = \begin{cases} 1 \text{ if } \arg\max_{j'}(\boldsymbol{t}_u[j']) = j \\ 0 \text{ if } \arg\max_{j'}(\boldsymbol{t}_u[j']) \neq j \end{cases}. \tag{7}$$

## 4 EXPERIMENTS

### 4.1 Experimental conditions

We evaluated the performance of our method on a music recommendation dataset[1] that contains user trait labels [16]. The dataset consists of 15,753 tracks including 395,056 listening events. A listening track list by user $u$, $\boldsymbol{x}_u$, is a binarized vector whose dimension $I$ is 15,753. For all 18,310 users, the dataset provides five-dimensional real-valued scores $\boldsymbol{t}_u$ that represent the strength of

[1]https://github.com/CPJKU/pers_bias

each BigFive trait. The number of users whose dominant traits are *ope*, *con*, *ext*, *agr*, and *neu* are 9100, 686, 1259, 1600, and 5665, respectively. Users' listening lists were obtained from tweets with the "#nowplaying" tag [24], and user traits were estimated from their 1,000 most recent tweets (excluding retweets) using the IBM personality Insight API. Data were split into training (80%) and testing (20%) by using the provided code[1] with seed 2034976. For each trait, positive and negative users were evenly split, i.e., training and testing data included an almost equal number of positive and negative users. Three types of test data were prepared in accordance with the procedure of the paper [16]: all users, positive users, and negative users.

For MultDAE and MultVAE, both encoders $\phi$ had two fully-connected layers. After the input vector $\boldsymbol{x}$ was reduced to the 600-dimensional hidden variables, 200(= $M$)-dimensional latent variables $\boldsymbol{z}$ were obtained. Encoders $\phi$ and decoders $\theta$ had symmetric structures (15,753→600→200→600→15,753). The proposed MultDAE and MultCVAE conditioned encoders $\phi$ by concatenating above 600-dimensional variables with label $\boldsymbol{y}$ and conditioned decoders $\theta$ by concatenating latent variables $\boldsymbol{z}$ with label $\boldsymbol{y}$, after the embedding layer converted the label $\boldsymbol{y}$ to a 20-dimensional, 2-dimensional, or 5-dimensional real-valued embedded vector for vec, pos-neg, or one-hot, respectively. We implemented CVAE with reference to [20] and publicly available codes[2]. Hyperparameters were set according to [14]. After training for 50 epochs, the best model was selected for testing on the basis of $NDCG@50$ of the validation data. We compared another VAE-based method, RecVAE [19], for reference, which was implemented on the basis of publicly available codes[3] [4].

Measurement indices were $Recall@K$ and $NDCG@K$, as shown in Eq. (8) and (10), respectively. Here, $\mathrm{rel}(i)$ is an indicator function, which takes one when the user selects an item $i$ and otherwise zero. Recall is the ratio of cases where rel is one among $K$ recommended items.

$$Recall@K = \frac{1}{K}\sum_{i=1}^{K}\mathrm{rel}(i) \tag{8}$$

DCG discounted items recommended to users according to the rank of a recommendation list, where the rank is determined by the number of plays.

$$DCG@K = \sum_{i=1}^{K}\frac{\mathrm{rel}(i)}{\log_2(i+1)} \tag{9}$$

$NDCG@K$ measures the order correspondences between users' listening lists and the lists output by the recommendation system up to $K$ ranks. NDCG is a division of DCG by ideal DCG (IDCG).

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{10}$$

## 4.2 Results and discussion

*4.2.1 Evaluation on all users.* Table 2 shows the $NDCG@\{50, 100\}$ and $Recall@20$ results evaluated on all users. As we can see, MultDAE and MultVAE outperformed RecVAE, which demonstrates that the assumption of a multinomial distribution is effective for this task. MultCDAE did not improve the performance of MultDAE but MultCVAE improved MultVAE[5]. This indicates that conditioning is effective for VAE because it is necessary to switch the distribution for each trait rather than simply concatenating the variables with labels. In terms of NDCG, on average, MultCVAE with the pos-neg label and one-hot outperformed MultVAE, which is significant and demonstrates the effectiveness of our proposed method. For all cases except two, one of the proposed methods performed the best. In terms of recall, the average overall performance was improved by using pos-neg labels. The performance of recall was not necessarily improved, because the optimal model was selected on the basis of $NDCG@50$.

**Table 2: NDCG and recall evaluated on all users. A paired t-test was performed and $^{\dagger}$ indicates statistical significance at $\alpha = 0.1$ level, $^{*}$ at $\alpha = 0.05$ level, and $^{**}$ at $\alpha = 0.01$ level.**

| | RecVAE | MultDAE | MultCDAE | MultVAE | MultCVAE | | |
|---|---|---|---|---|---|---|---|
| | | | vec | | vec | pos-neg | one-hot |
| | | | | NDCG@50 | | | |
| ope | 0.0554 | 0.0611 | 0.0541 | 0.0610 | 0.0577 | **0.0629** | 0.0622 |
| con | 0.0566 | 0.0634 | 0.0589 | 0.0648 | 0.0679$^{\dagger}$ | **0.0682**$^{\dagger}$ | 0.0673 |
| ext | 0.0610 | 0.0651 | 0.0611 | 0.0630 | 0.0616 | 0.0640 | **0.0660**$^{\dagger}$ |
| agr | 0.0543 | 0.0599 | 0.0614 | 0.0599 | 0.0611 | **0.0638**$^{*}$ | 0.0626$^{\dagger}$ |
| neu | 0.0563 | **0.0624** | 0.0589 | 0.0601 | 0.0591 | 0.0575 | 0.0599 |
| avg | 0.0567 | 0.0624 | 0.0589 | 0.0618 | 0.0615 | 0.0633$^{\dagger}$ | **0.0636**$^{*}$ |
| | | | | NDCG@100 | | | |
| ope | 0.0673 | 0.0742 | 0.0668 | 0.0729 | 0.0686 | **0.0749** | 0.0740 |
| con | 0.0671 | 0.0768 | 0.0706 | 0.0767 | **0.0822**$^{**}$ | 0.0803$^{\dagger}$ | 0.0796$^{\dagger}$ |
| ext | 0.0720 | 0.0776 | 0.0732 | 0.0750 | 0.0736 | 0.0752 | **0.0782**$^{\dagger}$ |
| agr | 0.0646 | 0.0734 | 0.0730 | 0.0722 | 0.0727 | **0.0768**$^{*}$ | 0.0747 |
| neu | 0.0663 | **0.0751** | 0.0709 | 0.0727 | 0.0704 | 0.0697 | 0.0729 |
| avg | 0.0675 | 0.0754 | 0.0709 | 0.0739 | 0.0735 | 0.0754$^{\dagger}$ | **0.0759**$^{*}$ |
| | | | | Recall@20 | | | |
| ope | 0.0696 | 0.0797 | 0.0690 | **0.0826** | 0.0777 | 0.0824 | 0.0770 |
| con | 0.0695 | 0.0772 | 0.0723 | 0.0802 | 0.0848 | **0.0880**$^{*}$ | 0.0834 |
| ext | 0.0760 | **0.0852** | 0.0742 | 0.0810 | 0.0801 | 0.0836 | 0.0816 |
| agr | 0.0707 | 0.0794 | **0.0826** | 0.0772 | 0.0810 | 0.0815 | 0.0787 |
| neu | 0.0702 | **0.0802** | 0.0728 | 0.0773 | 0.0758 | 0.0761 | 0.0745 |
| avg | 0.0712 | 0.0803 | 0.0742 | 0.0797 | 0.0799 | **0.0823**$^{\dagger}$ | 0.0790 |

*4.2.2 Evaluation on users who have positive and negative traits.* Next, we evaluated the same models on only the positive users whose BigFive scores were more than the median (i.e., almost half of the users) for each trait. Table 3 lists the results for this case. In general, compared with the case of all users in Table 2, for *ope* the performance was worse, for *con* and *ext*, the performance was similar, and for *agr* and *neu* and in average, the performance was better. In this case, the proposed method improved the performance. Except for one case, the performance of the proposed method was best, which is a similar trend to the results in 4.2.1. In this case, all types of labels were effective.

Table 4 shows the results for the negative users. In this case, MultDAE was better than MultVAE because the negative users include users with mixed traits who have different distributions, and it is not suitable for MultVAE without conditioning. The proposed

[2]https://github.com/chendaichao/VAE-pytorch
[3]https://github.com/ilya-shenbin/RecVAE
[4]https://github.com/dawenl/vae_cf

[5]Results of MultCDAE with pos-neg or one-hot were omitted for space.

**Table 3: NDCG and recall evaluated on users whose traits are positive.**

| | RecVAE | MultDAE | MultCDAE | MultVAE | MultCVAE | | |
|---|---|---|---|---|---|---|---|
| | | | vec | | vec | pos-neg | one-hot |
| | | | | NDCG@50 | | | |
| ope | 0.0490 | 0.0538 | 0.0485 | 0.0540 | 0.0510 | 0.0564 | **0.0574** |
| con | 0.0538 | 0.0602 | 0.0564 | 0.0652 | **0.0702**$^{\dagger}$ | 0.0675 | 0.0647 |
| ext | 0.0607 | 0.0630 | 0.0625 | 0.0639 | **0.0656** | 0.0639 | 0.0641 |
| agr | 0.0570 | 0.0641 | 0.0604 | 0.0643 | 0.0656 | **0.0681**$^{\dagger}$ | 0.0669 |
| neu | 0.0673 | **0.0706** | 0.0686 | 0.0697 | 0.0695 | 0.0654 | **0.0706** |
| avg | 0.0576 | 0.0623 | 0.0593 | 0.0634 | 0.0644 | 0.0643 | **0.0647** |
| | | | | NDCG@100 | | | |
| ope | 0.0602 | 0.0657 | 0.0596 | 0.0650 | 0.0609 | 0.0681 | **0.0690**$^{\dagger}$ |
| con | 0.0644 | 0.0734 | 0.0674 | 0.0764 | **0.0840**$^{*}$ | 0.0796 | 0.0771 |
| ext | 0.0712 | 0.0744 | 0.0737 | 0.0743 | **0.0765** | 0.0753 | 0.0763 |
| agr | 0.0673 | 0.0769 | 0.0726 | 0.0763 | 0.0783 | **0.0820**$^{*}$ | 0.0796 |
| neu | 0.0768 | **0.0843** | 0.0810 | 0.0820 | 0.0814 | 0.0782 | 0.0838 |
| avg | 0.0680 | 0.0749 | 0.0709 | 0.0748 | 0.0762 | 0.0766$^{\dagger}$ | **0.0772**$^{*}$ |
| | | | | Recall@20 | | | |
| ope | 0.0588 | 0.0705 | 0.0591 | 0.0713 | 0.0714 | **0.0729** | 0.0703 |
| con | 0.0659 | 0.0718 | 0.0719 | 0.0776 | **0.0887**$^{*}$ | 0.0830 | 0.0835 |
| ext | 0.0755 | 0.0860 | 0.0786 | 0.0773 | **0.0868**$^{*}$ | 0.0827 | 0.0826 |
| agr | 0.0770 | 0.0870 | 0.0772 | 0.0832 | 0.0857 | **0.0879** | 0.0861 |
| neu | 0.0844 | **0.0898** | 0.0841 | 0.0892 | 0.0896 | 0.0846 | 0.0866 |
| avg | 0.0723 | 0.0810 | 0.0742 | 0.0797 | **0.0844**$^{*}$ | 0.0822 | 0.0818 |

**Table 4: NDCG and recall evaluated on users whose traits are negative.**

| | RecVAE | MultDAE | MultCDAE | MultVAE | MultCVAE | | |
|---|---|---|---|---|---|---|---|
| | | | vec | | vec | pos-neg | one-hot |
| | | | | NDCG@50 | | | |
| ope | 0.0619 | 0.0684 | 0.0597 | 0.0679 | 0.0645 | **0.0694** | 0.0669 |
| con | 0.0593 | 0.0666 | 0.0615 | 0.0644 | 0.0656 | 0.0689 | **0.0700**$^{*}$ |
| ext | 0.0612 | 0.0672 | 0.0598 | 0.0621 | 0.0577 | 0.0641 | **0.0679**$^{*}$ |
| agr | 0.0516 | 0.0557 | **0.0625**$^{*}$ | 0.0554 | 0.0565 | 0.0596$^{\dagger}$ | 0.0583 |
| neu | 0.0453 | **0.0542** | 0.0492 | 0.0505 | 0.0487 | 0.0496 | 0.0492 |
| avg | 0.0559 | 0.0624 | 0.0585 | 0.0601 | 0.0586 | 0.0623$^{\dagger}$ | **0.0625**$^{*}$ |
| | | | | NDCG@100 | | | |
| ope | 0.0744 | **0.0827** | 0.0741 | 0.0808 | 0.0762 | 0.0818 | 0.0790 |
| con | 0.0698 | 0.0801 | 0.0737 | 0.0771 | 0.0803 | 0.0809 | **0.0821**$^{*}$ |
| ext | 0.0727 | **0.0808** | 0.0728 | 0.0757 | 0.0707 | 0.0751 | 0.0801$^{\dagger}$ |
| agr | 0.0619 | 0.0699 | **0.0734** | 0.0681 | 0.0671 | 0.0717 | 0.0698 |
| neu | 0.0559 | **0.0658** | 0.0609 | 0.0633 | 0.0594 | 0.0613 | 0.0619 |
| avg | 0.0669 | **0.0759** | 0.0710 | 0.0730 | 0.0707 | 0.0742 | 0.0746 |
| | | | | Recall@20 | | | |
| ope | 0.0804 | 0.0890 | 0.0790 | **0.0939** | 0.0839 | 0.0918 | 0.0837 |
| con | 0.0730 | 0.0827 | 0.0726 | 0.0829 | 0.0808 | **0.0930**$^{\dagger}$ | 0.0834 |
| ext | 0.0765 | 0.0844 | 0.0697 | **0.0848** | 0.0734 | 0.0845 | 0.0805 |
| agr | 0.0644 | 0.0719 | **0.0879**$^{*}$ | 0.0712 | 0.0763 | 0.0750 | 0.0714 |
| neu | 0.0561 | **0.0706** | 0.0615 | 0.0654 | 0.0621 | 0.0676 | 0.0625 |
| avg | 0.0701 | 0.0797 | 0.0741 | 0.0796 | 0.0753 | **0.0824** | 0.0763 |

method using pos-neg and one-hot labels improved the performance of MultVAE because it can reduce this influence and gave a consistent improvement for MultVAE.

*4.2.3 Distribution of latent variables of MultCVAE.* Figure 3 shows the probability density function of $\boldsymbol{\mu}$ at node 100 of the hidden layer of MultCVAE with one-hot labels. It was difficult to see the difference between traits from the raw histograms, so we fit Gaussian

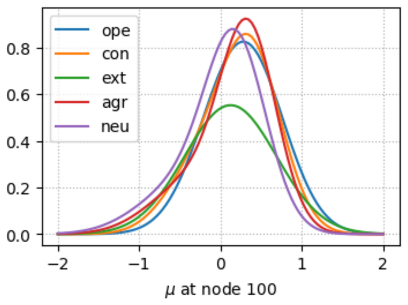

**Figure 3: Probability density function of $\boldsymbol{\mu}$ at nodes 100 dependent on user traits.**

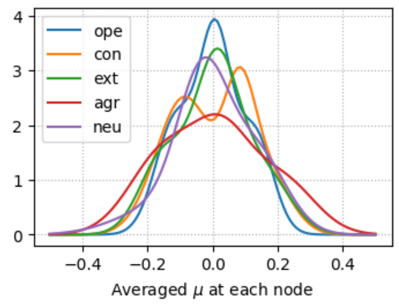

**Figure 4: Probability density function of $\boldsymbol{\mu}$ averaged over samples dependent on user traits.**

mixture models and drew their probability density functions. All the peaks of distributions took positive (non-zero) values and the variances were different from trait to trait, which demonstrates that the distributions were dependent on the traits.

Figure 4 shows the probability density function of $\boldsymbol{\mu}$ of each node averaged over all test samples. Gaussian mixture models were also fitted. As we can see, the variance of *agr* was larger than the others, the peaks were dependent on each trait, and there were two peaks for *con*. The shapes of the distributions are clearly different from trait to trait, which demonstrates the importance of switching distributions for user traits by conditioning.

## 5 CONCLUSION

To consider user traits for item recommendation task, we proposed a conditional MultVAE that conditions the MultVAE on the basis of user trait labels. We adopted five traits of the BigFive (OCEAN) for the user traits and introduced three types of labels: real-valued BigFive scores, binary labels of positive and negative, and one-hot index of the maximum BigFive scores. Experiments on a music recommendation task showed that both binary labels and the one-hot index were effective, which demonstrates that the proposed method that considers user traits is better than the conventional method that does not. We also found that the distribution of the generative model was dependent on user traits and clarified the importance of switching distributions by conditioning. Since the proposed framework is general and can be applied to other tasks, our future work will investigate the effectiveness of the proposed method on other tasks.

## REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17 (July 2005), 734–749. https://doi.org/10.1109/TKDE.2005.99

[2] Laura Burbach, Johannes Nakayama, Nils Plettenberg, Martina Ziefle, and André Calero Valdez. 2018. User Preferences in Recommendation Algorithms: The Influence of User Diversity, Trust, and Product Category on Privacy Perceptions in Recommender Algorithms. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 306–310. https://doi.org/10.1145/3240323.3240393

[3] T Chamorro-Premuzic and A Furnham. 2007. Personality and Music: Can Traits Explain How People Use Music in Everyday Life? *British Journal of Psychology* 98, 2 (May 2007), 175–185. https://doi.org/10.1348/000712606X111177

[4] Paul T Costa and Robert Barlow McCrea. 1992. *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO-FFI)*. Psychological Assessment Resources.

[5] Sahraoui Dhelim, Nyothiri Aung, Mohammed Amine Bouras, Huansheng Ning, and Erik Cambria. 2022. A Survey on Personality-aware Recommendation Systems. *Artificial Intelligence Review* 55 (2022), 2409–2454. https://doi.org/10.1007/s10462-021-10063-7

[6] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. 2016. Alleviating the New User Problem in Collaborative Filtering by Exploiting Personality Information. *User Modeling and User-Adapted Interaction* 26, 2–3 (Jun 2016), 221–255. https://doi.org/10.1007/s11257-016-9172-z

[7] Bruce Ferwerda, Marko Tkalcic, and Markus Schedl. 2017. Personality Traits and Music Genres: What Do People Prefer to Listen To?. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization* (Bratislava, Slovakia) *(UMAP '17)*. Association for Computing Machinery, New York, NY, USA, 285–288. https://doi.org/10.1145/3079628.3079693

[8] Rui Gao, Bibo Hao, Shuotian Bai, Lin Li, Ang Li, and Tingshao Zhu. 2013. Improving User Profile with Personality Traits Predicted from Social Media Content. In *Proceedings of the 7th ACM Conference on Recommender Systems* (Hong Kong, China) *(RecSys '13)*. Association for Computing Machinery, New York, NY, USA, 355–358. https://doi.org/10.1145/2507157.2507219

[9] J Golbeck, C Robles, M Edmondson, and K Turner. 2011. Predicting Personality from Twitter. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) / 2011 IEEE Third Intional Conference on Social Computing (SocialCom)*. IEEE Computer Society, Los Alamitos, CA, USA, 149–156. https://doi.org/10.1109/PASSAT/SocialCom.2011.33

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[11] Yoshinori Hijikata, Yuki Kai, and Shogo Nishida. 2014. A Study of User Intervention and User Satisfaction in Recommender Systems. *Journal of Information Processing* 22, 4 (2014), 669–678. https://doi.org/10.2197/ipsjjip.22.669

[12] Jun Hozumi, Yusuke Iwasawa, and Yutaka Matsuo. 2021. Time-sequential Variational Conditional Auto-encoders for Recommendation. *Transactions of the Japanese Society for Artificial Intelligence* 36, 3C (2021), 1–10.

[13] DP Kingma and M Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of International Conference on Learning Representations*. 1–14.

[14] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[15] Xinyuan Lu and Min-Yen Kan. 2023. Improving Recommendation Systems with User Personality Inferred from Product Reviews. In *Proceedings of the WSDM 2023 Workshop on Interactive Recommender Systems*. 1–9.

[16] Alessandro B Melchiorre, Eva Zangerle, and Markus Schedl. 2020. Personality Bias of Music Recommendation Algorithms. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 533–538. https://doi.org/10.1145/3383313.3412223

[17] S Rothmann and EP Coetzer. 2003. The Big Five Personality Dimensions and Job Performance. *SA Journal of Industrial Psychology* 29, 1 (2003). https://doi.org/10.4102/sajip.v29i1.88

[18] Markus Schedl, David Hauger, Katayoun Farrahi, and Marko Tkalcic. 2015. On the Influence of User Characteristics on Music Recommendation. In *European Conference on Information Retrieval*, A Hanbury, G Kazai, A Rauber, and N Fuhr (Eds.). Springer, 339–345. https://eprints.soton.ac.uk/420576/

[19] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I Nikolenko. 2020. 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* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 528–536. https://doi.org/10.1145/3336191.3371831

[20] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. 2015. Learning Structured Output Representation Using Deep Conditional Generative Models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS '15)*. MIT Press, Cambridge, MA, USA, 3483–3491.

[21] Marko Tkalcic and Li Chen. 2015. *Personality and Recommender Systems*. Springer, 715–739.

[22] EC Tupes and RE Christal. 1992. Recurrent Personality Factors Based on Trait Ratings. *Journal of Personality* 60, 2 (Jun 1992), 225–251. https://doi.org/10.1111/j.1467-6494.1992.tb00973.x

[23] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (San Francisco, California, USA) *(WSDM '16)*. Association for Computing Machinery, New York, NY, USA, 153–162. https://doi.org/10.1145/2835776.2835837

[24] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. 2014. #nowplaying Music Dataset: Extracting Listening Behavior from Twitter. In *Proceedings of the First International Workshop on Internet-Scale Multimedia Management* (Orlando, Florida, USA) *(WISMM '14)*. Association for Computing Machinery, New York, NY, USA, 21–26. https://doi.org/10.1145/2661714.2661719