Persona-Aware Tips Generation*

Piji Li
[1]Tencent AI Lab
Shenzhen, China
[2]The Chinese University of Hong Kong
Hong Kong, China
pijili@tencent.com

Zihao Wang
The Chinese University of Hong Kong
Hong Kong, China
zhwang@se.cuhk.edu.hk

Lidong Bing
R&D Center Singapore
Alibaba DAMO Academy
l.bing@alibaba-inc.com

Wai Lam
The Chinese University of Hong Kong
Hong Kong, China
wlam@se.cuhk.edu.hk

ABSTRACT

Tips, as a compacted and concise form of reviews, were paid less attention by researchers. In this paper, we investigate the task of tips generation by considering the "persona" information which captures the intrinsic language style of the users or the different characteristics of the product items. In order to exploit the persona information, we propose a framework based on adversarial variational auto-encoders (aVAE) for persona modeling from the historical tips and reviews of users and items. The latent variables from aVAE are regarded as persona embeddings. Besides representing persona using the latent embeddings, we design a persona memory for storing the persona related words for users and items. Pointer Network is used to retrieve persona wordings from the memory when generating tips. Moreover, the persona embeddings are used as latent factors by a rating prediction component to predict the sentiment of a user over an item. Finally, the persona embeddings and the sentiment information are incorporated into a recurrent neural networks based tips generation component. Extensive experimental results are reported and discussed to elaborate the peculiarities of our framework.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; • **Information systems** → **Recommender systems**; **Personalization**.

KEYWORDS

Abstractive Tips Generation; Rating Prediction; Persona Modeling; Adversarial Variational Auto-Encoders.

*The work described in this paper was partially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055093).

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313496

ACM Reference Format:
Piji Li, Zihao Wang, Lidong Bing, and Wai Lam. 2019. Persona-Aware Tips Generation. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313496

1 INTRODUCTION

Tips, specifically defined by Yelp[1], as a compacted and concise form of reviews, have unique advantages for helping users get a quick insight over an item. Conventional reviews are extensively studied for rating prediction [26, 37] and review generation [6, 29, 35, 40], while tips are paid relatively less attention. In [21], the tip information was explored for tips generation and rating prediction for the first time. The rationality for performing the joint task can be attributed to "writing some tips and giving a numerical rating are two facets of a user's product assessment action, expressing the user experience and feelings". Moreover, compared with reviews, tips are likely more consistent with the rating score with respect to sentiment tendency, because of its intrinsic form, i.e. compacted and concise.

In this paper, we investigate another dimension, namely user persona, which is plausibly helpful for the task of tips generation and has not be considered in the previous work [21]. Here the term "**persona**" denotes the characteristics of the written text by users such as wording and style. Figure 1a shows some tips for a shower radio from different users. [2] These tips clearly show different styles, although all of them have the same ratings. Some users (e.g. 1, 4 and 5) prefer short sentences and direct wordings such as "great", "easy", and "excellent" to describe the product quality and their experience directly. On the other hand, some users (e.g. 2, 3, and 6) share their experience indirectly by talking about some facts with longer sentences. Therefore, different users indeed have different "persona" style when writing tips. Figure 1b shows a few tips with different ratings from the same user for different items, we can observe that the user prefers short sentences, and moreover he has his own style (i.e. preferred vocabulary) for writing tips of different sentiments/ratings (e.g. "perfect" and "excellent" for high rating tips, and "piece of crap" for low rating tips).

[1]https://www.yelp-support.com/article/What-are-tips
[2]https://www.amazon.com/sony-icf-s79v-weather-shower-radio/dp/b00000dm9w

Tips	Rating
(1) Great fit and finish for shower.	5
(2) I selected this radio for myself several years ago and i have found that all claims for it are true.	5
(3) If your looking for a radio for your shower then look no further.	5
(4) Easy to set up stations.	5
(5) Excellent design and quality construction.	5
(6) First one lasted years just bought another one.	5

(a) Tips for the item "Sony Weather Band Shower Radio".

Tips	Rating
(1) Works perfectly in my msi wind.	5
(2) Perfect size for a home office.	5
(3) Excellent player for price.	5
(4) Wonderful docking speaker with full sound.	4
(5) I like it when it not dropping the signal.	4
(6) Works fine in a pinch.	3
(7) Piece of crap do bother.	1
(8) Revised star piece of crap.	1

(b) Tips for different items written by a particular user.

Figure 1: Example of tips.

Intuitively, the quality of abstractive tips generation can be improved if the model considers the user persona information when conducting the text generation. To do so, in this paper, we investigate an approach called **P**ersona-**A**ware **T**ips **G**eneration (**PATG**). There are two main challenges for the design of PATG: (1) How to capture and represent the persona information; (2) How to integrate the sentiment signal with the persona information to jointly control the style and the sentiment of the generated tips.

We distill persona information from all the historical tips and reviews of a user into the form of **Persona Embeddings**. Then the persona embeddings can be directly incorporated into the tips generation component as context information. Specifically, we employ variational auto-encoders (VAEs) [12] (which show strong capability in modeling latent random variables [20, 22]) to conduct persona modeling, and we regard the latent variables of VAEs as the persona embeddings. In the context of user behaviour analysis of online retailing, another indispensable party is the product item. Items also have their own intrinsic characteristics, such as product category in a coarser granularity, or specific features in a finer granularity. In this work, we personify the items and enable modeling them with "item persona", similar to modeling users. Besides distilling persona information using VAEs, we also design an external **Persona Memory** for the framework to store the persona related words for the current user and item. Pointer Networks [36] is used to retrieve appropriate words from the persona memory for generating tips.

Another obvious signal from the examples in Figure 1 is that the sentiment related wording is also bound to the sentiment ratings. For example, positive expressions are only used in tips with high ratings, no matter what persona the users have. To explore this signal for generating more accurate tips, our framework includes an auxiliary component: rating prediction with the information of users and items used for tips generation. The intuition is that if we can predict the rating of a user on an item accurately, the same input information should provide rich, if not complete, information for generating a tip satisfying that rating. Thus, in order to control the sentiment of the generated tips, we design a rating prediction component. The distilled persona embeddings are regarded as latent factors for users/items, and fed into the rating prediction component for detecting sentiment. A vectorization process is conducted on the predicted rating values and then the rating vectors are incorporated into the tips generation component as context information to control the sentiment of the generated tips.

The main contributions of our framework are summarized below:

- We develop a framework that tackles the task of persona-aware tips generation, where persona information, such as writing style and vocabulary preference, is considered for the first time to conduct the tips generation.
- In order to exploit the persona information, we design an adversarial variational auto-encoders (aVAE) based approach for persona modeling for users and items, i.e. generating persona embeddings. We employ an external memory based Pointer Networks to conduct the memory reading to retrieve more accurate persona information.
- In order to control the sentiment of the generated tips, we tightly couple an auxiliary component of rating prediction with the tips generation component. The distilled persona embeddings are used as latent factors of users and items for the sentiment rating prediction.
- Experimental results show that our framework achieves better performance than the state-of-the-art models on tips generation. Moreover, an additional observation is that the persona information can improve the performance of the auxiliary task, i.e. rating prediction.

2 FRAMEWORK DESCRIPTION

2.1 Overview

The data consists of users, items, ratings, review content, and tips. We denote the whole training corpus by $\mathcal{X} = \{\mathcal{U}, \mathcal{I}, \mathcal{R}, C, \mathcal{S}\}$, where $\mathcal{U}$ and $\mathcal{I}$ are the sets of users and items respectively, $\mathcal{R}$ is the set of ratings, C is the set of review documents, and $\mathcal{S}$ is the set of tips texts. We use C_u and $\mathcal{S}_u$ to denote all the historical reviews and tips respectively of the user u. For a quick reference, Table 1 lists all notations used in our paper.

As shown in Figure 2, our framework contains two major modules: persona modeling on the left and abstractive tips generation on the right. For modeling persona, our framework leverages the tips and reviews from each individual user or written by multiple users for the same item. Take the historical tips $\mathcal{S}_u$ of the user u as an example, we represent them using bag-of-words (BoWs) vectors $\mathbf{x}_u^s$. Then we feed $\mathbf{x}_u^s$ into the adversarial variational auto-encoders (aVAE^s) and obtain the persona embedding $\mathbf{z}_u^s$ for the user u. For the item i, we can also conduct similar persona modeling based on the historical tips $\mathcal{S}_i$ written by different users, and the obtained persona embedding is denoted as $\mathbf{z}_i^s$. The purpose of persona modeling for the item i is that when conducting tips generation for the user u, the model will also consider the tips from other users having similar interests with u, since they will disclose more characteristics of the item. We call this phenomenon *personalized collaborative*

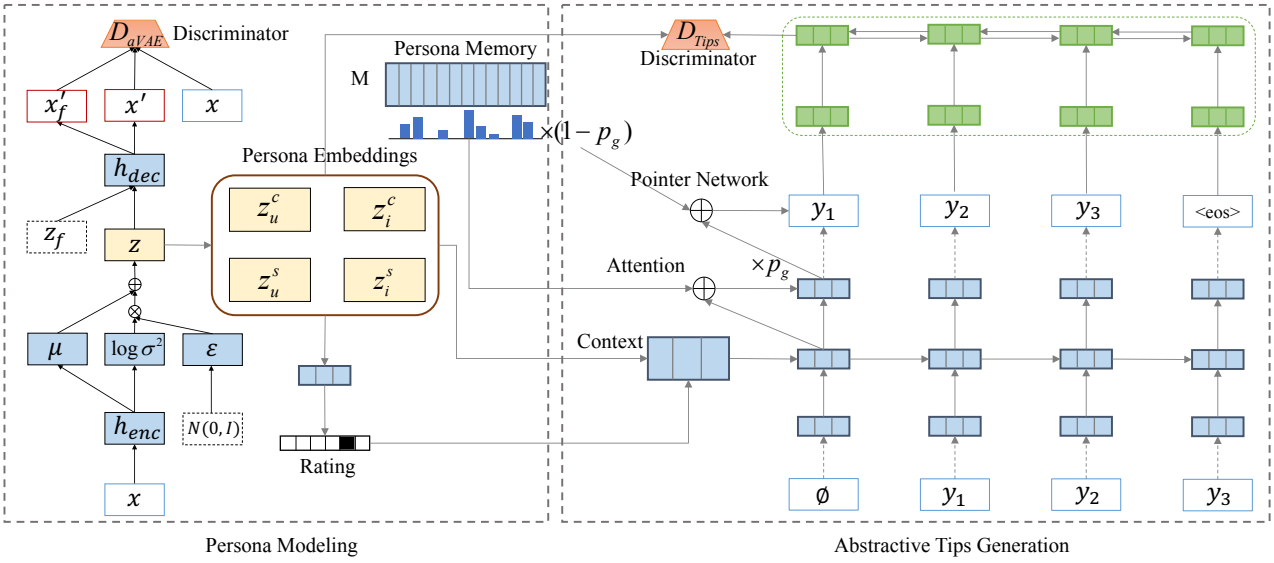

Figure 2: Our proposed framework for persona-aware abstractive tips generation.

Table 1: Glossary.

Symbol	Description
$\mathcal{X}$	training set
$\mathcal{V}$	vocabulary
$\mathcal{U}$	set of users
$\mathcal{I}$	set of items
$\mathcal{R}$	set of ratings
$\mathcal{C}$	set of reviews
$\mathcal{S}$	set of tips
$\mathcal{C}_u$	historical reviews for user u
$\mathcal{S}_u$	historical tips for user u
$\mathbf{M}$	external memory
$\mathbf{Z}$	persona embeddings
$\mathbf{E}$	word embeddings
$\mathbf{H}$	neural hidden states
$\mathbf{W}$	mapping matrix
$\mathbf{b}$	bias item
Θ	set of neural parameters
$r_{u,i}$	rating of user u to item j
σ	sigmoid function
ς	softmax function
$relu$	rectified linear unit
$tanh$	hyperbolic tangent function

influence. We also distill persona information from reviews with another aVAE model (aVAE^c) to map the historical reviews C_u and C_i to persona embeddings $\mathbf{z}_u^c$ and $\mathbf{z}_i^c$ for the user u and the item i respectively.

We design an external persona memory $\mathbf{M}$ for storing the persona related words for the current user and item which will be utilized in abstractive tips generation. In order to control the sentiment of the generated tips, the distilled persona embeddings are used as latent factors for users and items and are fed into a multilayer perceptron (MLP) based neural network component to predict the rating r. Then we transform r to a one-hot vector $\mathbf{r}$ which will be used as the sentiment controller when conducting the tips generation. For the step of tips generation, we design a sequence decoding model based on a neural network of Gated Recurrent Units (GRUs) [5]. Importantly, the persona embeddings and the rating vector are combined to construct a context vector which plays a significant role in the abstractive tips generation. In addition, Pointer Networks is used to retrieve relevant words from the persona memory $\mathbf{M}$, with a gate p_g to control the source of the next output word.

2.2 Persona Modeling

2.2.1 ***Persona Embedding Learning.*** The target of persona modeling is to distill the persona information from the users' historical tips and reviews. Some previous works in recommendation systems [26, 30, 37] employ topic modeling methods such as Latent Dirichlet Allocation (LDA) [2] or its variants to analyze the text corpus and use the latent topic distribution to represent each document. Considering the fact that our tips generation component is based on neural networks, existing topic modeling paradigms cannot be incorporated into our framework in an elegant manner. Instead, we employ the variational auto-encoders (VAEs) [12] for detecting the latent topics with neural modeling paradigm [4]. VAEs consists of two parts: inference (variational-encoder) and generation (variational-decoder). Recall that the dictionary is V. For historical tips based persona modeling, the input are the BoWs vectors $\mathbf{x}_u^s \in \mathbb{R}^{|V|}$ and $\mathbf{x}_i^s \in \mathbb{R}^{|V|}$ for the user u and the item i respectively. For convenience, we will use $\mathbf{x}$ to represent them in this section. As shown in the left part of Figure 2, for each input BoWs vector $\mathbf{x}$, the variational-encoder can map it to a latent variable $\mathbf{z} \in \mathbb{R}^K$, which can be used to generate a new variable $\mathbf{x}'$ via the variational-decoder component to reconstruct the original term vector. The target is to maximize the probability of each $\mathbf{x}$ in the dataset based

on the generation process according to:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{1}$$

For the purpose of solving the intractable integral of the marginal likelihood, a model $q(\mathbf{z}|\mathbf{x})$ is introduced as the approximation to the intractable of the true posterior $p(\mathbf{z}|\mathbf{x})$. The aim of optimization is to reduce the Kulllback-Leibler divergence (KL) between $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ by maximizing the variational lower bound $\mathcal{L}_{VAE}$:

$$\mathcal{L}_{VAE} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - D_{KL}[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] \tag{2}$$

In order to differentiate and optimize the lower bound $\mathcal{L}_{VAE}$, following the core idea of VAEs, we use a neural network framework for the encoder $q(\mathbf{z}|\mathbf{x})$ for better approximation. Similar to previous works [12], we assume that both the prior and posterior of the latent variables are Gaussian, i.e., $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ and $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the variational mean and standard deviation respectively, which can be calculated with a multilayer perceptron (MLP). Precisely, given the BoWs vector $\mathbf{x}$ of the historical tips, we first project it to a hidden space:

$$\mathbf{h}_{enc} = relu(\mathbf{W}_{xh}\mathbf{x} + \mathbf{b}_{xh}) \tag{3}$$

where $\mathbf{h}_{enc} \in \mathbb{R}^{d_h}$, $\mathbf{W}_{xh}$ and $\mathbf{b}_{xh}$ are the neural parameters. $relu(\mathbf{x}) = max(0, \mathbf{x})$ is the activation function. Then the Gaussian parameters $\boldsymbol{\mu} \in \mathbb{R}^K$ and $\boldsymbol{\sigma} \in \mathbb{R}^K$ can be obtained via a linear transformation based on $\mathbf{h}_{enc}$:

$$\begin{aligned} \boldsymbol{\mu} &= \mathbf{W}_{h\mu}\mathbf{h}_{enc} + \mathbf{b}_{h\mu} \\ \log(\boldsymbol{\sigma}^2) &= \mathbf{W}_{h\sigma}\mathbf{h}_{enc} + \mathbf{b}_{h\sigma} \end{aligned} \tag{4}$$

In order to make the sampling operation differentiable, the latent variable $\mathbf{z} \in \mathbb{R}^K$ can be calculated using the reparameterization trick:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad \mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\varepsilon} \tag{5}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^K$ is an auxiliary noise variable. This is the encoding process, and we denote all the parameters of this state as Θ_{Enc}.

Given the latent variable $\mathbf{z}$, a new vector $\mathbf{x}'$ is generated via the conditional distribution $p(\mathbf{x}|\mathbf{z})$ according to the variational-decoder:

$$\mathbf{h}_{dec} = relu(\mathbf{W}_{zh}\mathbf{z} + \mathbf{b}_{zh}) \tag{6}$$

$$\mathbf{x}' = \sigma(\mathbf{W}_{hx}\mathbf{h}_{dec} + \mathbf{b}_{hx}) \tag{7}$$

We denote all the parameters in the decoding stage using Θ_{Dec}. Finally, based on the reparameterization trick in Equation 5, we can get the analytical representation of $\mathcal{L}_{VAE}$:

$$\begin{aligned} \log p(\mathbf{x}|\mathbf{z}) &= \sum_{i=1}^{|V|} x_i \log x'_i + (1 - x_i) \cdot \log(1 - x'_i) \\ -D_{KL}[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] &= \tfrac{1}{2} \sum_{i=1}^{K} (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \end{aligned} \tag{8}$$

For presentation clarity, we let:

$$\begin{aligned} \mathcal{L}_{Rec} &= -\log p(\mathbf{x}|\mathbf{z}) \\ \mathcal{L}_{KL} &= D_{KL}[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] \end{aligned} \tag{9}$$

and both of them need to be minimized.

We wish to employ the latent variable $\mathbf{z}$ as the distilled persona embeddings. So the quality of $\mathbf{z}$ will affect the performance of tips generation. Some previous works [9, 27, 43] have also shown that the performance of $\mathbf{z}$ is likely to be disturbed during the training procedure, especially when combining VAEs with the RNN based text generation framework. In order to enhance the performance of the typical VAEs, inspired by the ideas in [8] and [15], we employ the adversarial strategy for the training of VAEs. Generally, we design a discriminator network D_{aVAE} with a vector $\tilde{\mathbf{x}}$ as input, and the target is to recognize if $\tilde{\mathbf{x}}$ is from the true data $\mathbf{X}$ or from the generated samples $\mathbf{X}'$ by VAEs. VAEs will "fool" the discriminator D_{aVAE} by trying the best to produce high quality latent variables $\mathbf{z}$ as well as the generated sample $\mathbf{x}'$. Then the minimax game between the VAEs and the discriminator can be formulated as follows:

$$\begin{aligned} \min_{VAEs} \max_{D_{aVAE}} & \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D_{aVAE}(\mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}[\log(1 - D_{aVAE}(VAE_{Dec}(\mathbf{z})))] \\ & + \mathbb{E}_{\mathbf{z}_f\sim p(\mathbf{z})}[\log(1 - D_{aVAE}(VAE_{Dec}(\mathbf{z}_f)))] \end{aligned} \tag{10}$$

where VAE_{Dec} is the decoder component of the VAEs model. $\mathbf{z}$ is the latent variable from VAEs, and $\mathbf{z}_f$ is sampled from the prior distribution of $\mathbf{z}$.

For the design of the discriminator D_{aVAE}, we simply use a multilayer perceptron to process the data.

$$\begin{aligned} \mathbf{h}^{D_v} &= tanh(\mathbf{W}_{xh}^{D_v}\tilde{\mathbf{x}} + \mathbf{b}_{xh}^{D_v}) \\ y^{D_v} &= \sigma(\mathbf{W}_{hy}^{D_v}\mathbf{h}^{D_v} + b_{hy}^{D_v}) \end{aligned} \tag{11}$$

where $\mathbf{W}_{xh}^{D_v} \in \mathbb{R}^{d_h \times |V|}$, $\mathbf{W}_{hy}^{D_v} \in \mathbb{R}^{1 \times d_h}$, $\mathbf{b}_{xh}^{D_v} \in \mathbb{R}^{d_h}$, and $b_{hy}^{D_v} \in \mathbb{R}$. The output y^{D_v} is a real value in the range of $[0, 1]$ and the value 1 means that the sample $\tilde{\mathbf{x}}$ is from the true data. We denote all the parameters in D_{aVAE} using Θ_{D_v}. The optimization objective to be maximized for D_{aVAE} is formulated as:

$$\begin{aligned} \mathcal{L}_{D_{aVAE}} = & \log(D_{aVAE}(\mathbf{x})) \\ & + \log(1 - D_{aVAE}(VAE_{Dec}(VAE_{Enc}(\mathbf{x})))) \\ & + \log(1 - D_{aVAE}(VAE_{Dec}(\mathbf{z}_f))) \end{aligned} \tag{12}$$

Then the parameters Θ_{D_v} are updated using gradient methods:

$$\Theta_{D_v} \leftarrow \Theta_{D_v} - \nabla_{\Theta_{D_v}}(-\mathcal{L}_{D_{aVAE}}) \tag{13}$$

Conditioned on the aVAE framework, we will conduct the parameter learning for VAEs Encoder, VAEs Decoder, and discriminator D_{aVAE} using different loss functions respectively. Encoder transforms the input $\mathbf{X}$ to the persona embeddings $\mathbf{Z}$. On one side, $\mathbf{Z}$ are used to reconstruct the original input. On the other side, $\mathbf{Z}$ are used to conduct the persona-aware tips generation. So the loss signals from both the aVAE and the tips generation framework are used to conduct the optimization for Θ_{Enc}:

$$\Theta_{Enc} \leftarrow \Theta_{Enc} - \nabla_{\Theta_{Enc}}(\mathcal{L}_{KL} + \mathcal{L}_{Rec} + \mathcal{L}^z_{D_{aVAE}} + \mathcal{L}_{Tips}) \tag{14}$$

where $\mathcal{L}_{KL}$ and $\mathcal{L}_{Rec}$ are the KL divergence and reconstruction loss from Equation 8. $\mathcal{L}_{Tips}$ is the loss signal from the tips generation component. $\mathcal{L}^z_{D_{aVAE}}$ is the output of D_{aVAE}:

$$\mathcal{L}^z_{D_{aVAE}} = -\log(D_{aVAE}(VAE_{Dec}(VAE_{Enc}(\mathbf{x})))) \tag{15}$$

For the parameter optimization of VAEs Decoder, we use $\mathcal{L}_{Rec}$, $\mathcal{L}_{D_{aVAE}}$, $\mathcal{L}_{Tips}$ as the loss signals:

$$\Theta_{Dec} \leftarrow \Theta_{Dec} - \nabla_{\Theta_{Dec}}(\mathcal{L}_{\text{Rec}} + \mathcal{L}_{D_{aVAE}} + \mathcal{L}_{Tips}) \tag{16}$$

Finally, the training procedure of aVAE model is shown in Algorithm 1.

Algorithm 1 Persona embedding learning.

Input: BoWs vectors of historical tips and reviews $\mathbf{X}$.
Output: The persona embeddings $\mathbf{Z}$.

Initialize $\Theta_{Enc}, \Theta_{Dec}, \Theta_{D_v}$;
while not converged **do**
  Draw $\mathbf{x}$ from p_{data}.
  Draw $\mathbf{z}_f$ from prior $p(\mathbf{z})$.
  $\mathbf{z} = VAE_{Enc}(\mathbf{x})$
  $\mathbf{x}' = VAE_{Dec}(\mathbf{z})$
  $\mathbf{x}'_f = VAE_{Dec}(\mathbf{z}_f)$
  Get $\mathcal{L}_{Rec}$, $\mathcal{L}_{KL}$, $\mathcal{L}_{D_{aVAE}}$ according to Equation 8 and 12.
  Get $\mathcal{L}_{Tips}$ from tips generation.
  Update parameters using gradient methods:
  $\Theta_{Enc} \leftarrow \Theta_{Enc} - \nabla_{\Theta_{Enc}}(\mathcal{L}_{KL} + \mathcal{L}_{Rec} + \mathcal{L}^z_{D_{aVAE}} + \mathcal{L}_{Tips})$
  $\Theta_{Dec} \leftarrow \Theta_{Dec} - \nabla_{\Theta_{Dec}}(\mathcal{L}_{\text{Rec}} + \mathcal{L}_{D_{aVAE}} + \mathcal{L}_{Tips})$
  $\Theta_{D_v} \leftarrow \Theta_{D_v} - \nabla_{\Theta_{D_v}}(-L_{D_{aVAE}})$
end while
return $\mathbf{z}$.

Feeding the historical reviews and tips representations ($\mathbf{x}^c_u$, $\mathbf{x}^c_i$, $\mathbf{x}^s_u$, and $\mathbf{x}^s_i$) into $aVAE^c$ (for reviews) and $aVAE^s$ (for tips) respectively, we can obtain four persona embeddings $\mathbf{z}^c_u$, $\mathbf{z}^c_i$, $\mathbf{z}^s_u$, and $\mathbf{z}^s_i$. These persona embeddings will be integrated into the rating prediction component and the tips generation component later.

2.2.2 ***Sentiment and Rating Modeling.*** We regard the persona embeddings as the latent factors of users and items, and feed them into a multilayer perceptron to conduct the rating prediction. The predicted ratings will be used to control the sentiment of the generated tips.

Specifically, we first map the persona embeddings to a hidden space:

$$\mathbf{h}^r = tanh(\mathbf{W}^r_{uch}\mathbf{z}^c_u + \mathbf{W}^r_{ich}\mathbf{z}^c_i + \mathbf{W}^r_{ush}\mathbf{z}^s_u + \mathbf{W}^r_{ish}\mathbf{z}^s_i + \mathbf{b}^r_h) \quad (17)$$

where $\{\mathbf{W}^r_{uch}, \mathbf{W}^r_{ich}, \mathbf{W}^r_{ush}, \mathbf{W}^r_{ish}\} \in \mathbb{R}^{d_h \times k}$ are the mapping matrices. $\mathbf{b}^r_h \in \mathbb{R}^{d_h}$ is the bias term. $tanh$ is the hyperbolic tangent activation function. The superscript r refers to variables related to the rating prediction component. For better performance, we can add more layers of non-linear transformations into our model:

$$\mathbf{h}^r_l = \sigma(\mathbf{W}^r_{hh_l}\mathbf{h}^r_{l-1} + \mathbf{b}^r_{h_l}) \quad (18)$$

where $\mathbf{W}^r_{hh_l} \in \mathbb{R}^{d_h \times d_h}$ is the mapping matrix for the variables in the hidden layers. l is the index of a hidden layer. Assume that $\mathbf{h}^r_L$ is the output of the last hidden layer. The output layer transforms $\mathbf{h}^r_L$ into a real-valued rating $\hat{r}$:

$$\hat{r} = \mathbf{W}^r_{hr}\mathbf{h}^r_L + b^r \quad (19)$$

where $\mathbf{W}^r_{hr} \in \mathbb{R}^{1 \times d_h}$ and $b^r \in \mathbb{R}$. We formulate the optimization of the parameters Θ_r as a regression problem and the loss function is formulated as:

$$\mathcal{L}^r = \frac{1}{2|\mathcal{X}|}\sum_{u \in \mathcal{U}, i \in \mathcal{I}}(\hat{r}_{u,i} - r_{u,i})^2 \quad (20)$$

where $\mathcal{X}$ represents the training set. $r_{u,i}$ is the ground truth rating assigned by the user u to the item i.

The predicted rating is a real value, not a vector, for example, $\hat{r}_{u,i} = 4.321$. In order to incorporate the rating information into the tips generation component, we cast it into an integer 4, and add a vectorization process to obtain the vector representation of rating $\hat{r}_{u,i}$. If the rating range is $[0, 5]$, we will get the rating vector $\hat{\mathbf{r}}_{u,i} = (0, 0, 0, 0, 1, 0)^T$.

2.2.3 ***External Persona Memory.*** In addition to represent persona information using the latent embeddings from aVAE, we design an external persona memory for directly storing the persona related words for both the current user u and the current item i. To build the memory, we first collect all the words for the current user u and the current item t from their historical tips. We add a filtering process to remove the stop-words and the low-frequency words. Then we get a local vocabulary storing the indices of the persona words. Recall that we have a global word embedding $\mathbf{E}$. Then we can get a sub-matrix from $\mathbf{E}$ according to the word indices. We regard this sub-matrix as persona memory. We employ Pointer Networks to retrieve persona information from the memory when generating tips. The details are described in Section 2.3.3.

2.3 Abstractive Tips Generation

2.3.1 ***Overview of Tips Generation.*** The right part of Figure 2 depicts our tips generation model. The basic element is a RNN based sequence modeling component. Pointer Networks (attention modeling and copy mechanism) is introduced to conduct the memory reading. Context information plays an important role in the task of text generation. We combine the persona embeddings and the sentiment information as the context information and construct the context vector which can control the tips text generation. At the training state, we also design a discriminator D_{Tips} to assess the quality of the generated tips. The assess value will be propagated to the RNN models to assist the parameter learning. At the operational or testing stage, we use a beam search algorithm [13] for decoding and generating the best tips given a trained model.

2.3.2 ***Sequence Modeling.*** Assume that $\mathbf{h}^s_t$ is the sequence hidden state at the time t. It depends on the input at the time t and the previous hidden state $\mathbf{h}^s_{t-1}$:

$$\mathbf{h}^s_t = f(\mathbf{h}^s_{t-1}, s_t) \quad (21)$$

$f(\cdot)$ can be the vanilla RNN, Long Short-Term Memory (LSTM) [10], or Gated Recurrent Unit (GRU) [5]. Considering that GRU has comparable performance but with less parameters and more efficient computation, we employ GRU as the basic model in our sequence modeling framework. In the case of GRU, the state updates are processed according to the following operations:

$$\begin{aligned}
\mathbf{r}^s_t &= \sigma(\mathbf{W}^s_{sr}\mathbf{s}_t + \mathbf{W}^s_{hr}\mathbf{h}^s_{t-1} + \mathbf{b}^s_r) \\
\mathbf{z}^s_t &= \sigma(\mathbf{W}^s_{sz}\mathbf{s}_t + \mathbf{W}^s_{hz}\mathbf{h}^s_{t-1} + \mathbf{b}^s_z) \\
\mathbf{g}^s_t &= \tanh(\mathbf{W}^s_{sh}\mathbf{s}_t + \mathbf{W}^s_{hh}(\mathbf{r}^s_t \odot \mathbf{h}^s_{t-1}) + \mathbf{b}^s_h) \\
\mathbf{h}^s_t &= \mathbf{z}^s_t \odot \mathbf{h}^s_{t-1} + (1 - \mathbf{z}^s_t) \odot \mathbf{g}^s_t
\end{aligned} \quad (22)$$

where $\mathbf{s}_t \in \mathbf{E}$ is the embedding vector for the word s_t of the tips and the vector is also learnt from our framework. $\mathbf{r}^s_t$ is the reset gate, $\mathbf{z}^s_t$ is the update gate. $\odot$ denotes element-wise multiplication.

In order to conduct the persona-aware tips generation, we combine all the persona embeddings and the sentiment information as

the context information and construct the context vector. Specifically, we initialize the hidden state $\mathbf{h}_0$ using the persona embeddings and the sentiment information:

$$\mathbf{h}_0^s = tanh(\mathbf{W}_{uch}^s \mathbf{z}_\mathbf{u}^\mathbf{c} + \mathbf{W}_{ich}^s \mathbf{z}_\mathbf{i}^\mathbf{c} + \mathbf{W}_{ush}^s \mathbf{z}_\mathbf{u}^\mathbf{s} + \mathbf{W}_{ish}^s \mathbf{z}_\mathbf{i}^\mathbf{s} + \mathbf{W}_{rh}^s \hat{\mathbf{r}} + \mathbf{b}_h^s) \tag{23}$$

where $\{\mathbf{z}_*^*\}$ are the persona embeddings. $\hat{\mathbf{r}}$ is the vectorization for the predicted rating $\hat{r}$. $\mathbf{W}$ and $\mathbf{b}$ are the neural parameters.

After getting all the sequence hidden states based on GRU, we feed them to the final output layer to predict the word sequence in tips.

$$\hat{\mathbf{s}}_{t+1} = \varsigma(\mathbf{W}_{hs}^s \mathbf{h}_t^s + \mathbf{b}^s) \tag{24}$$

where $\mathbf{W}_{hs}^s \in \mathbb{R}^{d \times |\mathcal{V}|}$ and $\mathbf{b}^s \in \mathbb{R}^{|\mathcal{V}|}$. $\varsigma(\cdot)$ is the softmax function. Then the word with the largest probability is the decoding result for the step $t+1$:

$$w_{t+1}^* = \arg\max_{w_i \in \mathcal{V}} \hat{\mathbf{s}}_{t+1}^{(w_i)} \tag{25}$$

At the training stage, we use negative log-likelihood (NLL) as the loss function, where I_w is the vocabulary index of the word w:

$$\mathcal{L}_{Tips} = -\sum_{w \in Tips} \log \hat{\mathbf{s}}^{(I_w)} \tag{26}$$

Note that $\mathcal{L}_{Tips}$ is also used in the persona modeling component to train the aVAE models.

At the testing stage, given a trained model, we employ the beam search algorithm [13] to find the best sequence S^* having the maximum log-likelihood.

$$S^* = \arg\max_{S \in \mathcal{S}} \sum_{w \in S} \log \hat{\mathbf{s}}^{(I_w)} \tag{27}$$

2.3.3 ***Exploiting Persona Memory***. Recall that in Section 2.2.3, we build a local personal vocabulary V_{ui} for the user u and the item i. The persona memory $\mathbf{M}_{ui}$ is extracted from the word embedding $\mathbf{E}$ using the word indices in V_{ui}. Inspired by [1], we exploit the idea of attention modeling to conduct the addressing and reading operations on the memory $\mathbf{M}_{ui}$. We can obtain the GRU hidden state $\mathbf{h}_t^s$ according to Equation (21). Then the attention weights at the time step t are calculated based on the relationship between $\mathbf{h}_t^s$ with all the word embeddings in $\mathbf{M}_{ui}$. Let $a_{i,j}$ be the attention weight between $\mathbf{h}_i^s$ and $\mathbf{m}_j$, which can be calculated using:

$$\begin{aligned} a_{i,j} &= \frac{\exp(e_{i,j})}{\sum_{j'=1}^{|V_{ui}|} \exp(e_{i,j'})} \\ e_{i,j} &= \mathbf{v}_a{}^T \tanh(\mathbf{W}_{hh}^s \mathbf{h}_i^s + \mathbf{W}_{hh}^m \mathbf{m}_j + \mathbf{b}_a) \end{aligned} \tag{28}$$

where $\mathbf{W}_{hh}^s \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{W}_{hh}^m \in \mathbb{R}^{d_w \times d_h}$, $\mathbf{b}_a \in \mathbb{R}^{d_h}$, and $\mathbf{v}_a \in \mathbb{R}^{d_h}$. The attention context is obtained by the weighted linear combination of all the word embeddings in $\mathbf{M}_{ui}$:

$$\mathbf{c}_t = \sum_{j'=1}^{|V_{ui}|} a_{t,j'} \mathbf{m}_{j'} \tag{29}$$

The final hidden state $\mathbf{h}_t^{s_2}$ is the output of the second decoder GRU layer, jointly considering the word $\mathbf{s}_t$, the previous hidden state $\mathbf{h}_{t-1}^{s_2}$, and the attention context $\mathbf{c}_t$:

$$\mathbf{h}_t^{s_2} = GRU_2(\mathbf{h}_{t-1}^{s_2}, \mathbf{s}_t, \mathbf{c}_t) \tag{30}$$

Then we can use $\mathbf{h}_t^{s_2}$ as the input to Equation (24) to conduct the decoding operation.

Besides using attention modeling to address and read the persona information from the the persona memory $\mathbf{M}$, we also employ the idea of Pointer Networks [36] to copy the target words from the memory to form the tips. At the state t, we can obtain the attention weights (distribution) $\mathbf{a}_{t,:}$ on the persona memory $\mathbf{M}_{ui}$. We project $\mathbf{a}_{t,:}$ to a $|V|$-sized vector $\hat{\mathbf{s}}_{t+1}^p$ according to the word indices in V_{ui}. Then we design a soft gate to decide that the word s_{t+1} should be generated or be copied from the memory:

$$p_g = \sigma(\mathbf{v}_p^T(\mathbf{W}_{hp}^s \mathbf{h}_t^{s_2} + \mathbf{W}_{sp}^s \mathbf{s}_t + \mathbf{W}_{cp}^s \mathbf{c}_t + \mathbf{b}_p)) \tag{31}$$

where $\mathbf{v}_p \in \mathbb{R}^{d_h}$ and $p_g \in (0,1)$. We merge the copy signal $\hat{\mathbf{s}}_{t+1}^p$ and the original output $\hat{\mathbf{s}}_{t+1}$ according to the gate p_g:

$$\hat{\mathbf{s}}'_{t+1} = p_g \times \hat{\mathbf{s}}_{t+1} + (1 - p_g) \times \hat{\mathbf{s}}_{t+1}^p \tag{32}$$

Then the tips sampling process can be conducted on $\hat{\mathbf{s}}'_{t+1}$.

2.3.4 ***Tips Quality Discriminator***. Some previous works [40, 41] show that adversarial training strategy is beneficial to the text generation problem. To further improve the performance, we also employ this training strategy in our framework.

The tips discriminator D_{Tips} is a multilayer perceptron with the persona embeddings, the rating information, and the tips sequence as the input. The input tips sequence can be the ground truth S or the tips $\hat{S}$ generated by the system. We propose a Bidirectional-GRU model to conduct the representation learning for S and $\hat{S}$:

$$\mathbf{h}^S = \overrightarrow{\mathbf{h}^S} || \overleftarrow{\mathbf{h}^S} \tag{33}$$

Then we combine all the information according to:

$$\mathbf{h}^q = \tanh(\mathbf{W}_{sh}^q \mathbf{h}^S + \mathbf{W}_{uch}^q \mathbf{z}_\mathbf{u}^\mathbf{c} + \mathbf{W}_{ich}^q \mathbf{z}_\mathbf{i}^\mathbf{c} + \mathbf{W}_{ush}^q \mathbf{z}_\mathbf{u}^\mathbf{s} + \mathbf{W}_{ish}^q \mathbf{z}_\mathbf{i}^\mathbf{s} + \mathbf{W}_{rh}^q \hat{\mathbf{r}} + \mathbf{b}_h^q)$$

Finally, we add a softmax output layer to let the model output a binary category variable:

$$\mathbf{y}^q = \varsigma(\mathbf{W}_{hy}^q \mathbf{h}^q + \mathbf{b}_y^q) \tag{34}$$

We treat the ground truth S as the positive instance and the sampled sequence $\hat{S}$ as the negative instance. So we directly let the first dimension of $\mathbf{y}^q$ represent the positive label. We define the value function as $V(S) = \mathbf{y}_{[0]}^q$. We utilize the REINFORCE [38] method to integrate the tips quality signal $V(S)$ into the tips generation framework to conduct the parameter learning. The details can be found in the existing works [18, 40, 41].

3 EXPERIMENTAL SETUP

3.1 Datasets

In our experiments, we use five datasets from different domains to evaluate our framework. The ratings of all these datasets are integers in the range of $[1, 5]$. There are four datasets from Amazon 5-core[3]: **Electronics**, **Movies & TV**, **Clothing, Shoes and Jewelry**, and **Home and Kitchen**. We regard the field "summary" as tips, and the number of tips texts is the same with the number of reviews. Another dataset is from **Yelp** Challenge[4]. It is also a large-scale dataset consisting of restaurant reviews and tips. We filter out the words with low term frequency in the tips and review texts,

[3]http://jmcauley.ucsd.edu/data/amazon
[4]https://www.yelp.com/dataset_challenge

Table 2: Overview of the datasets.

	Electr	Movies	Home	Clothing	Yelp
users	191,522	123,340	66,212	39,085	115,781
items	62,333	49,823	27,991	22,794	60,224
reviews	1,684,779	1,693,441	550,461	277,521	1,393,257
$\|\mathcal{V}\|$	37,999	82,805	23,950	16,297	82,805

and build a vocabulary $\mathcal{V}$ for each dataset. We show the statistics of our datasets in Table 2.

3.2 Evaluation Metrics

For the evaluation of abstractive tips generation, the ground truth s_h is the tips written by the user. We use *ROUGE* [24] as our evaluation metric with standard options[5]. It is a classical evaluation metric in the field of text summarization [24]. It counts the number of overlapping units between the generated tips and the ground truth written by users. Assuming that s is the generated tips, g_n is n-gram, $C(g_n)$ is the number of n-grams in $\tilde{s}$ (s_h or s), $C_m(g_n)$ is the number of n-grams that appear in both s and s_h, then the ROUGE-N score for s is defined as follows:

$$ROUGE\text{-}N(s) = \sum_{g_n \in s_h} C_m(g_n) / \sum_{g_n \in \tilde{s}} C(g_n) \tag{35}$$

When $\tilde{s} = s_h$, we can get $ROUGE_{recall}$, and when $\tilde{s} = s$, we get $ROUGE_{precision}$. We use Recall, Precision, and F-measure of ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L), and ROUGE-SU4 (R-SU4) to evaluate the quality of the generated tips.

For the evaluation of rating prediction, we employ two metrics: Mean Absolute Error (*MAE*) and Root Mean Square Error (*RMSE*). Both of them are widely used for rating prediction in recommender systems. Given a predicted rating $\hat{r}_{u,i}$ and a ground-truth rating $r_{u,i}$ from the user u for the item i, the RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2} \tag{36}$$

where N indicates the number of ratings between users and items. Similarly, MAE is calculated as follows:

$$MAE = \frac{1}{N} \sum_{u,i} |r_{u,i} - \hat{r}_{u,i}| \tag{37}$$

3.3 Comparative Methods

Abstractive tips generation: We compare our framework PATG with the following baseline and state-of-the-art methods:

- **NRT** [21]: It is a recent multi-task learning framework for rating prediction and abstractive tips generation achieving state-of-the-art performance. Latent factors for users and items are learnt during the training procedure, and are used as the context information for tips generation. NRT does not consider the persona information.
- **LexRank** [7] is a classical method in the field of text summarization. Because we have obtained all the historical tips for the current user and item, then the problem can be regarded as a multi-document summarization problem. LexRank can extract a sentence as the final tips. Note that we give an advantage of this method since the ground truth ratings are used to conduct the filtering.
- **CTR**$_t$: **C**ollaborative **T**opic **R**egression (CTR) [37] is proposed for rating prediction. It contains a topic model component and it can generate topics for items. Then the most topic-similar sentence from the item historical tips is extracted as the tips.
- **HFT**$_t$: **H**idden **F**actors and Hidden **T**opics [26] utilizes a topic modeling technique to model the review texts for rating prediction. Then we can design a tips extraction method HFT$_t$ using the similar technique in CTR$_t$.

Rating prediction: We compare of rating prediction performance with the following baseline methods:

- **NMF**: **N**on-negative **M**atrix **F**actorization [16]. It only uses the rating matrix as the input.
- **PMF**: **P**robabilistic **M**atrix **F**actorization [32]. Gaussian distribution is introduced to model the latent factors for users and items.
- **LRMF**: **L**earning to **R**ank with **M**atrix **F**actorization [34]. It combines a list-wise learning-to-rank algorithm with matrix factorization to improve recommendation.
- **SVD++**: It extends **S**ingular **V**alue **D**ecomposition by considering implicit feedback information for latent factor modeling [14].
- **URP**: **U**ser **R**ating **P**rofile modeling [25]. Topic models are employed to model the user preference from a generative perspective. It still only uses the rating matrix as input.
- The baseline methods used in tips quality evaluation: **NRT** [21], **CTR** [37], **HFT** [26].

Ablation experiments: In order to demonstrate the performance of each component of our framework, we conduct the ablation experiments on the dataset Home. We compare the performance of our integrated model PATG with the models without the some designed components. We set that "A" denotes the aVAE model, "M" represents the persona memory and the Pointer Networks, and "D" represents the tips quality discriminator D_{Tips}. Then the method "**PATG w/o A, M, D**" means that A, M, and D are all removed and we only use the standard VAE for persona modeling.

3.4 Experimental Settings

Each dataset is divided into three subsets: 80%, 10%, and 10%, for training, validation, and testing, respectively. All the parameters of our model are tuned with the validation set. After the tuning process, the number of latent factors k is set to 10 for NMF and SVD++. The number of topics K is set to 50 for the methods using topic models. The number of dimension for the persona embeddings is set to 100. The dimension of the hidden size is 400. In our framework, the number of layers for the rating regression model is 2, and for the tips generation model is 1. We set the beam size $\beta = 5$, and the maximum length $\eta = 20$. All the neural matrix parameters in hidden layers and RNN layers are initialized from a uniform distribution between $[-0.1, 0.1]$. We also regard the word embedding $\mathbf{E}$ used in the tips generation component as a neural parameter. Adadelta [42] is used for gradient based optimization.

[5]ROUGE-1.5.5.pl -n 4 -w 1.2 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0

4 RESULTS AND DISCUSSIONS

4.1 Research Questions

The research questions in our experiments are as follows:

- **RQ1**: What is the performance of PATG in persona-aware abstractive tips generation? (Section 4.2)
- **RQ2**: Can the persona embeddings improve the performance of rating prediction? (Section 4.3)
- **RQ3**: What is the performance of each component of PATG, such as VAEs, aVAE, and the persona memory? (Section 4.4)
- **RQ4**: Can the model generate tips that are complying with the persona information? (Section 4.5.)
- **RQ5**: Can the model generate tips that are really controlled by ratings? (Section 4.6.)

4.2 Abstractive Tips Generation (RQ1)

The evaluation results of tips generation of our model and the comparative methods are given in Table 3. In order to capture more details, we report Recall, Precision, and F-measure (in percentage) of ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU4. Our model achieves the best performance in most of the metrics among all the five datasets. NRT does not consider persona information when generating tips. It only utilizes the learnt latent factors for users and items as the context information. Compared with NRT, our proposed framework PATG obtains better performance on all the metrics, which demonstrates that the consideration of persona information can indeed improve the tips generation performance. We also conduct statistical significance test comparing PATG and NRT and the results indicate that the improvements are significant with $p < 0.05$.

From the results, we also find that our model obtains dramatic improvements on the metric of ROUGE Precision, especially compared with the methods of LexRank, HFT_t, and CTR_t. The main reasons are that those methods are all extraction-based which just extract some original sentences from the original reviews or tips as the final tips. Therefore, the obtained tips are much longer with more noisy and redundant information. In contrast, our framework PATG as well as NRT are abstractive tips generation methods. PATG can generate more concise sentences which not only guarantee the recall metric, but also obtain better precision performance. This also fits the essential spirit of Tips.

4.3 Rating Prediction (RQ2)

Recall that we also design an auxiliary component, i.e. **rating prediction** to capture the sentiment information and control the sentiment of the generated tips, which is also an important aspect of the task of tips generation. Therefore we design some experiments to evaluate the performance of this component. The rating prediction results are given in Table 4. Our model consistently outperforms the best under both MAE and RMSE metrics on all datasets, thus it verifies that the generated persona embeddings are not only effective for generating better tips in the main task, but also useful for predicting accurate ratings in the auxiliary task. Statistical significance of differences between the performance of PATG and the recent method NRT is tested using a two-tailed paired t-test. The result shows that PATG is significantly better than NRT.

4.4 Ablation Analysis (RQ3)

Considering that we design various of components to tackle the corresponding problems of our task, and different components play different roles on our framework. In order to demonstrate the necessity and the performance of each component, we conduct the ablation experiments on the dataset "Home". The results are shown in Table 5. Recall that "A" denotes the aVAE model, "M" represents the persona memory and the Pointer Networks, and "D" represents the tips quality discriminator D_{Tips}. It is obvious that persona modeling based on aVAE (A) can improve the tips generation performance. The persona memory and Pointer Networks (M) are very helpful to the effectiveness of our framework. The tips quality discriminator (D) can also contribute to the better performance. Among all the components, aVAE (A) as well as the persona memory and pointer network (M) contribute more to the improvements of the performance.

4.5 Persona Controlled Generation (RQ4)

The main problem setting of this work is to generate persona-aware tips. In order to demonstrate the quality of the generated tips, we selected some real cases generated by our PATG from different domains for some users and items. The results are listed in Table 6. Although our model generates tips in an abstractive way, tips' linguistic quality is quite good. The **persona properties** of the generated tips match well with the ground truth. For example, in the first case, the generated tips is "This is a great hat for the price.", and the ground truth is "Thanks nice quality excellent price great deal.". Both of the sentences contain the terms "great" and "price". In the third case, the generated tips and the ground truth have a large overlapping with the terms "replace my old", and "processor". Interestingly, sometimes the framework can select some synonyms when conducting tips generation. For instance, the generated tips of the fourth case contains terms "bought" and "for my husband". The ground truth contains "purchased" and "for a male". Moreover, we also choose some generated tips with negative sentiment to conduct the sentiment correlation analysis. Take the generated tips "Please do not buy this coffee maker." as an example (the last case in Table 6), our model predicts a rating of 2.01, which clearly shows a consistent sentiment. The ground truth tips of this example is " They are still overpriced and all but worthless.", which also conveys a negative sentiment. The generated tips "The bottom line of the thin man." and the ground truth "Pretty dark story in book or movie form." are just describing some facts, with a neutral rating 3. Sometimes the overlapping between the generated tips and the ground truth is small, but they still convey similar information.

4.6 Rating Controlled Generation (RQ5)

Recall that in addition to the persona embeddings as context information, rating information is also incorporated to control the sentiment of the generated tips. In order to show such ability of our framework, we design an experiment on the domain "Home" to demonstrate the rating controlled tips generation. Specifically, during the prediction, we manually set the rating from 1 to 5 as the sentiment context to control the generation, and meanwhile, we create a new user and a new item with **0** persona embeddings. This setting mimics a **cold start** case: what the tips will look like

Table 3: ROUGE evaluation on the five datasets from different domains.

Dataset	Method	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
Electronics	LexRank	10.97	12.93	11.58	0.95	1.05	0.97	9.96	11.70	10.50	3.08	3.91	3.22
	HFT_t	12.86	12.22	12.35	1.10	1.00	1.03	11.65	11.09	11.19	3.43	3.10	3.14
	CTR_t	12.69	11.72	12.02	1.13	1.05	1.07	11.65	10.74	11.02	3.45	3.06	3.14
	NRT	12.79	17.55	13.85	1.86	2.77	2.08	11.80	15.99	12.70	4.18	6.42	4.45
	PATG	**13.00**	**19.26**	**14.52***	**2.29**	**3.12**	**2.44***	**11.91**	**17.42**	**13.24***	**4.50**	**7.44**	**4.89***
Movies&TV	LexRank	11.10	13.50	11.89	1.06	1.29	1.12	10.02	12.12	10.70	3.25	4.33	3.46
	HFT_t	11.64	10.26	11.33	1.78	1.36	1.46	11.42	8.72	9.67	4.63	3.00	3.28
	CTR_t	11.37	10.33	10.68	1.43	1.31	1.34	10.40	9.44	9.76	3.17	2.73	2.84
	NRT	12.12	20.06	14.17	2.29	3.53	2.55	11.13	18.25	12.98	4.09	8.15	4.79
	PATG	**12.46**	**21.22**	**14.63***	**2.38**	**3.88**	**2.67***	**11.51**	**19.25**	**14.73***	**6.04**	**8.76**	**6.33***
Home	LexRank	12.91	15.47	13.77	1.73	2.06	1.82	11.72	13.97	12.46	3.93	5.02	4.15
	HFT_t	13.32	12.72	12.80	1.33	1.23	1.25	12.25	11.73	11.79	3.63	3.33	3.34
	CTR_t	14.30	13.21	13.55	1.73	1.50	1.58	13.14	12.11	12.43	4.18	3.66	3.78
	NRT	11.51	19.91	13.64	1.95	3.47	2.30	10.64	18.23	12.57	3.77	8.24	4.51
	PATG	12.21	**21.46**	**14.61***	**2.32**	**4.32**	**2.78***	11.32	**19.65**	**13.48***	4.03	**8.71**	**4.82***
Clothing	LexRank	13.31	12.73	12.85	1.06	1.02	1.02	11.97	11.43	11.54	3.47	3.24	3.26
	HFT_t	13.31	12.73	12.85	1.06	1.02	1.02	11.97	11.43	11.54	3.47	3.24	3.26
	CTR_t	13.79	13.82	13.37	1.26	1.23	1.22	12.54	12.14	12.16	3.70	3.52	3.49
	NRT	13.52	18.91	14.75	2.11	2.95	2.31	12.36	17.04	13.39	4.58	7.04	4.86
	PATG	**14.45**	**21.49**	**16.14***	**2.49**	**3.77**	**2.79***	**13.09**	**19.24**	**14.55***	**4.93**	**8.39**	**5.39***
Yelp	LexRank	9.19	12.09	10.28	1.07	1.33	1.15	8.45	11.13	9.45	2.65	3.90	3.01
	HFT_t	10.47	10.21	10.26	0.91	0.87	0.88	9.56	9.31	9.35	2.70	2.57	2.59
	CTR_t	10.68	10.51	10.51	0.98	0.94	0.96	9.70	9.53	9.54	2.77	2.68	2.68
	NRT	10.98	17.42	12.71	1.82	3.03	2.13	9.96	15.76	11.51	3.48	6.48	4.05
	PATG	**12.05**	**19.15**	**14.02***	**2.15**	**3.44**	**2.47***	**10.94**	**17.21**	**12.66***	**3.96**	**7.15**	**4.57***

The "*" marker denotes that PATG achieves better performance than NRT with statistical significance test with $p < 0.05$.

Table 4: MAE and RMSE values for rating prediction.

	Electronics		Movies		Yelp		Clothing		Home	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LRMF	1.986	2.208	1.891	2.136	1.721	1.982	1.936	2.179	2.028	2.248
PMF	1.139	1.553	0.911	1.307	1.133	1.538	2.355	2.724	1.395	1.780
NMF	0.869	1.266	0.809	1.155	0.961	1.136	0.887	1.257	0.830	1.220
SVD++	0.841	1.226	0.778	1.122	1.957	1.299	0.829	1.169	0.786	1.164
URP	0.875	1.185	0.797	1.101	0.973	1.246	0.876	1.136	0.831	1.135
CTR	0.903	1.154	0.863	1.116	1.051	1.285	0.847	1.094	0.826	1.086
HFT	0.813	1.117	0.769	1.041	0.940	1.191	0.805	1.080	0.773	1.058
NRT	0.823	1.108	0.751	1.038	0.935	1.187	0.828	1.102	0.779	1.058
PATG	**0.747***	**1.016***	**0.740***	**1.015***	**0.866***	**1.134***	**0.714***	**0.987***	**0.694***	**0.997***

* denotes that PATG achieves better performance than NRT [21] with statistical significance test with $\alpha = 0.01$.

without knowing any information of users and items. Then the manual rating and the **0** based persona embeddings are fed into the framework to conduct tips generation. The results are shown in Table 7. Here we set the beam size to 5 so we obtain 5 decoded tips ranked by the likelihood in descending order. It is obvious that when $r > 1$, our framework can generated reasonable tips controlled by ratings. The generated tips show monotone language style, which looks odd, but in fact it is not, because we did not give any persona information of users and items as input (refer to Table 6 for generated tips with rich text such as product information). Moreover, note that this is an artificial scenario which does not compile well with the real case. Table 7 shows that when $r = 1$, the framework does not generate negative tips. We investigate the training dataset and find that the proportion of rating-1 records is much smaller (e.g. a quarter of rating-4 and one-thirteenth of

Table 5: Ablation experiments on the dataset Home. R-* represents the F1-Measure of ROUGE-*.

System	R-1	R-2	R-L	R-SU4
PATG w/o A, M, D	13.76	2.27	12.64	4.45
PATG w/o M, D	13.99	2.61	12.95	4.71
PATG w/o D	14.32	2.72	13.30	4.81
PATG	14.51	2.72	13.48	4.81

Table 6: Examples of the predicted ratings and the generated tips for some users and items. The first line of each group shows the generated rating and tips. The second line shows the ground truth.

Rating	Tips
5.10	***This is a great hat for the price.***
5	Thanks nice quality excellent price great deal.
5.08	***This is a great pitcher.***
5	Beautiful pitcher makes a great vase.
5.17	***I bought this food processor to replace my old one.***
4	I got this about a month ago to replace my old food processor.
4.99	***These shoes are so comfortable and I bought these for my husband.***
5	Comfortable good looking shoes purchased for a male that walks a lot.
4.81	***This is a great movie.***
5	Amazing love great movie and all teen shold see it.
2.57	***The bottom line of the thin man.***
3	Pretty dark story in book or movie form.
2.01	***Please do not buy this coffee maker.***
1	They are still overpriced and all but worthless.

rating-5), which may cause our model under-fitting for generating rating-1 tips.

5 RELATED WORK

Abstractive text generation is a challenging task. Recently, sequence modeling based on the gated recurrent neural networks such as Long Short-Term Memory (LSTM) [10] and Gated Recurrent Unit (GRU) [5] demonstrates high capability in text generation related tasks, such as abstractive summarization [19, 28, 31], dialogue systems [3, 33] and image caption generation [39].

In the area of recommendation systems, some researchers also apply LSTM or GRU based RNN models on abstractive text generation. Tang et al. [35] propose a framework to generate context-aware reviews. Sentiments and products are encoded into a continues semantic representation and use RNN to conduct the decoding and generation. Dong et al. [6] regard users, products, and rating as attribute information and employ a attention modeling based sequence modeling framework to generate reviews. Ni et al. [29] propose to combine collaborative filtering with generative networks to jointly perform the tasks of item recommendation and review generation. Low-dimensional user preferences and item properties are combined with a character-level LSTM model to conduct the review generation. Yao et al. [40] employ the adversarial strategy to make the generated review indistinguishable from human written

Table 7: Rating controlled tips generation in a cold start scenario. $\hat{r}$ is the rating usd to control the sentiment.

$\hat{r}$	Tips
5	This is a great product. I bought this for my mom and she loves it. I bought this for my daughter and she loves it. I bought this for my husband and she loves it. I bought this for my daughter.
4	This is a good product. I bought this for my daughter for christmas. I bought this for my daughter and she loved it. I bought this for my mom and she loved it. I bought this for my daughter for christmas and she loves it.
3	Not as good as my old one. This is a good product. Not as good as my old one. I bought this for my daughter for christmas. I bought this for my mom and she loved it.
2	Not as good as I expected. Not as good as the original. Not as good as my old one. I bought this for my daughter for christmas. This is a good product.
1	This is a good product. I bought this for my daughter for christmas. This is the third one i bought. This is the third one i had. I bought this for my daughter for christmas and she loves it.

ones so that can improve the performance of review generation. Although research works have been proposed for review generation, there are very few works investigating tips generation. Li et al. [21] propose a unified framework to jointly conduct rating prediction and abstractive tips generation. Latent factors for users and items are learnt from the multi-task learning framework and are fed into the tips generation framework as context information.

However, few works consider persona modeling and sentiment detection jointly in their frameworks. Hu et al. [11], Liao et al. [23] revised the variational auto-encoders (VAEs) based text generation model and can control the sentiment and tense of the generated reviews. But they still do not consider persona information in their model. Li et al. [17] propose two methods to conduct the persona modeling for text generation in the area of dialog systems, but dialog systems have different characteristics with recommendation system. Moreover, they do not consider the sentiment information. Different with these previous works, our proposed framework can jointly consider the persona information and the sentiment signal when conducting the abstractive tips generation.

6 CONCLUSIONS

We propose a framework PATG to address the problem of persona-aware tips generation. A framework based on adversarial variational auto-encoders (aVAE) is exploited for persona modeling from the historical tips and reviews. We also design an external persona memory for directly storing the persona related words for the current user and item. The distilled persona embeddings are used as latent factors and are fed into the rating prediction component for detecting sentiment. Then the persona embeddings and the

sentiment information are incorporated into a recurrent neural networks (RNN) based tips generation component to control the tips generation. Experimental results show that our framework achieves better performance than the state-of-the-art models on abstractive tips generation.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3, Jan (2003), 993–1022.

[3] Deng Cai, Yan Wang, Victoria Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2018. Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory. *arXiv preprint arXiv:1809.05296* (2018).

[4] Dallas Card, Chenhao Tan, and Noah A Smith. 2017. A Neural Framework for Generalized Topic Models. *arXiv preprint arXiv:1705.09296* (2017).

[5] Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *EMNLP* (2014), 1724–1734.

[6] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL*, Vol. 1. 623–632.

[7] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR* 22 (2004), 457–479.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.

[9] Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. 2017. Z-Forcing: Training Stochastic Recurrent Networks. In *NIPS*. 6716–6726.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[11] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *ICML*. 1587–1596.

[12] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

[13] Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas*. Springer, 115–124.

[14] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. ACM, 426–434.

[15] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *ICML*. 1558–1566.

[16] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*. 556–562.

[17] Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. In *ACL*, Vol. 1. 994–1003.

[18] Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070* (2018).

[19] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep Recurrent Generative Decoder for Abstractive Text Summarization. In *EMNLP*. 2091–2100.

[20] Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017. Salience Estimation via Variational Auto-Encoders for Multi-Document Summarization. In *AAAI*. 3497–3503.

[21] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural Rating Regression with Abstractive Tips Generation for Recommendation. In *SIGIR*. ACM, 345–354.

[22] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*. ACM, 305–314.

[23] Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam, and Tong Zhang. 2018. QuaSE: Sequence Editing under Quantifiable Guidance. In *EMNLP*. 3855–3864.

[24] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out-ACL Workshop*. 74–81.

[25] Benjamin M Marlin. 2003. Modeling user rating profiles for collaborative filtering. In *NIPS*. 627–634.

[26] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. ACM, 165–172.

[27] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. In *ICML*. 2391–2400.

[28] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 280–290.

[29] Jianmo Ni, Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2017. Estimating Reactions and Recommending Products with Generative Models of Reviews. In *IJCNLP*, Vol. 1. 783–791.

[30] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*. ACM, 485–494.

[31] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*. 379–389.

[32] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization.. In *NIPS*. 1–8.

[33] Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *ACL*, Vol. 1. 1577–1586.

[34] Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys*. 269–272.

[35] Jian Tang, Yifan Yang, Sam Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware Natural Language Generation with Recurrent Neural Networks. *arXiv preprint arXiv:1611.09900* (2016).

[36] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*. 2692–2700.

[37] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*. ACM, 448–456.

[38] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3-4 (1992), 229–256.

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. 2048–2057.

[40] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. 2017. Automated Crowdturfing Attacks and Defenses in Online Review Systems. In *CCS*. ACM, 1143–1158.

[41] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

[42] Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[43] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*. 654–664.