Neural Rating Regression with Abstractive Tips Generation for Recommendation*

Piji Li
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong.
pjli@se.cuhk.edu.hk

Zihao Wang
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong.
zhwang@se.cuhk.edu.hk

Zhaochun Ren
Data Science Lab, JD.com, North Star Century Center, Beijing, China
renzhaochun@jd.com

Lidong Bing
AI Lab, Tencent Inc.
Hi-tech Park, Nanshan District, Shenzhen, China
lyndonbing@tencent.com

Wai Lam
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong.
wlam@se.cuhk.edu.hk

ABSTRACT

Recently, some E-commerce sites launch a new interaction box called Tips on their mobile apps. Users can express their experience and feelings or provide suggestions using short texts typically several words or one sentence. In essence, writing some tips and giving a numerical rating are two facets of a user's product assessment action, expressing the user experience and feelings. Jointly modeling these two facets is helpful for designing a better recommendation system. While some existing models integrate text information such as item specifications or user reviews into user and item latent factors for improving the rating prediction, no existing works consider tips for improving recommendation quality. We propose a deep learning based framework named **NRT** which can simultaneously predict precise ratings and generate abstractive tips with good linguistic quality simulating user experience and feelings. For abstractive tips generation, gated recurrent neural networks are employed to "translate" user and item latent representations into a concise sentence. Extensive experiments on benchmark datasets from different domains show that NRT achieves significant improvements over the state-of-the-art methods. Moreover, the generated tips can vividly predict the user experience and feelings.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Recommender systems**; *Collaborative filtering*;

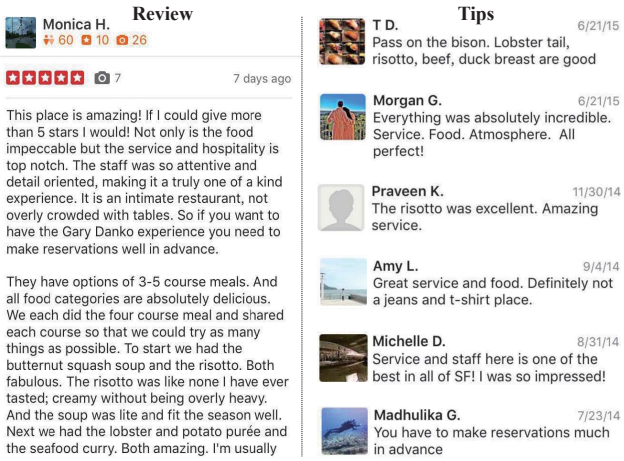


Figure 1: Examples of reviews and tips selected from the restaurant "Gary Danko" on Yelp. Tips are more concise than reviews and can reveal user experience, feelings, and suggestions with only a few words. Users will get conclusions about this restaurant immediately after scanning the tips with their mobile phones.

KEYWORDS

Rating Prediction; Tips Generation; Deep Learning.

1 INTRODUCTION

With the explosive growth of Internet information, recommendation systems have been playing an increasingly important role in on-line E-commerce and applications in a variety of areas, including music streaming service such as Spotify[1] and Apple Music, movie rating such as IMDB[2], video streaming service such as Netflix and Youtube, job recommendation such as LinkedIn[3], and product recommendation such as Amazon. Many recommendation methods are based on Collaborative Filtering (CF) which mainly makes use

*The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414) and the Microsoft Research Asia Urban Informatics Grant FY14-RES-Sponsor-057. This work is also affiliated with the CUHK MoE-Microsoft Key Laboratory of Human-centric Computing and Interface Technologies.

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

ACM ISBN 978-1-4503-5022-8/17/08…$15.00
https://doi.org/10.1145/3077136.3080822

[1] http://www.spotify.com
[2] http://www.imdb.com
[3] http://www.linkedin.com

of historical ratings [14, 15, 18, 22, 31, 33, 35]. Recently, some approaches also consider text information in addition to the rating data [1, 21, 23, 26, 40, 49]. After some investigations, we observe that the text information in most recommendation tasks can be generally classified into two types: item specifications [40–42] and user reviews [1, 21, 23, 26, 46, 47, 49]. Item specifications are the text information for describing the attributes or properties of the items. For example, in article recommendation such as CiteULike[4], it refers to titles and abstracts of papers. In product recommendation such as Amazon, it refers to product descriptions and technical specification information. The second type is user reviews which are written by users to explain why they like or dislike an item based on their usage experiences. Multi-faceted information can be extracted from reviews and used as user preferences or item features, which otherwise cannot be obtained from the overall ratings [5]. Although both types of text data are found to be useful for the recommendation task, they have some inherent limitations. Concretely, the former cannot reflect users' experience and preference, and the latter is usually too long and suffers from noise.

Recently, some E-commerce sites such as Yelp[5] launch a new interaction box called **Tips** on their mobile platforms. As shown in Figure 1, the left column is a review from the user "Monica H.", and tips from several other users are shown on the right column. In the review text, Monica first generally introduced the restaurant, and then narrated her dining experience in detail. In the tips text, users expressed their experience and feelings plainly using short texts, such as "The risotto was excellent. Amazing service.". They also provide some suggestions to other people directly in several words, such as "You have to make reservations much in advance." In contrast to item specifications and user reviews, tips have several characteristics: (1) tips are typically single-topic nuggets of information, and shorter than reviews with a length of about 10 words on average; (2) tips can express user experience, feelings, and suggestions directly; (3) tips can give other people quick insights, saving the time of reading long reviews. In essence, writing some tips and giving a numerical rating are two facets of a user's product assessment action, expressing the user experience and feelings. Jointly modeling these two facets is helpful for designing a better recommendation system.

Existing models only integrate text information such as item specifications [40–42] and user reviews [1, 21, 23, 26, 46, 47, 49] to enhance the performance of latent factor modeling and rating prediction. To our best knowledge, we are the first to consider tips for improving the recommendation quality. We aim at developing a model that is capable of conducting the latent factor modeling and rating prediction, and more importantly, it can generate tips based on the learnt latent factors. We do not just extract some existing sentences and regard them as tips. Conversely, we investigate the task of automatically construing a concise sentence as tips, such capability can be treated as simulating how users write tips in order to express their experience and feelings, just as if they have bought and consumed the item. Therefore, we named this task *abstractive tips generation*, where "abstractive" is a terminology from the research of text summarization [3].

[4]http://www.citeulike.org
[5]http://www.yelp.com

Generating abstractive tips only based on user latent factors and item latent factors is a challenging task. Recently, gated recurrent neural networks such as Long Short-Term Memory (LSTM) [12] and Gated Recurrent Unit (GRU) [6] demonstrate high capability in text generation related tasks [2, 30]. Moreover, inspired by [11, 41], neural network based models can help learn more effective latent factors when conducting rating prediction and improve the performance of collaborative filtering. We employ deep learning techniques for latent factor modeling, rating prediction, and abstractive tips generation. For abstractive tips generation, gated recurrent neural networks are employed to "translate" a user latent factor and an item latent factor into a concise sentence to express user experience and feelings. For neural rating regression, a multilayer perceptron network [28] is employed to project user latent factors and item latent factors into ratings. All the neural parameters in the gated recurrent neural networks and the multilayer perceptron network as well as the latent factors for users and items are learnt by a multi-task learning approach in an end-to-end training paradigm.

The main contributions of our framework are summarized below:

- We propose a deep learning based framework named NRT which can simultaneously predict precise ratings and generate abstractive tips with good linguistic quality simulating user experience and feelings. All the neural parameters as well as the latent factors for users and items are learnt by a multi-task learning approach in an end-to-end training paradigm.
- We are the first to explore using tips information to improve the recommendation quality. In essence, writing some tips and giving a numerical rating are two facets of a user's product assessment action, expressing the user experience and feelings. Jointly modeling these two facets is helpful for designing a better recommendation system.
- Experimental results on benchmark datasets show that our framework achieves better performance than the state-of-the-art models on both tasks of rating prediction and abstractive tips generation.

2 RELATED WORKS

Collaborative filtering (CF) has been studied for a long time and has achieved some success in recommendation systems [27, 37]. Latent Factor Models (LFM) based on Matrix Factorization (MF) [15] play an important role for rating prediction. Various MF algorithms have been proposed, such as Singular Value Decomposition (SVD) and SVD++ [14], Non-negative Matrix Factorization (NMF) [18], and Probabilistic Matrix Factorization (PMF) [31]. These methods map users and items into a shared latent factor space, and use a vector of latent features as the representation for users and items respectively. Then the inner product of their latent factor vectors can reflect the interactions between users and items.

The recommendation performance will degrade significantly when the rating matrix is very sparse. Therefore, some works consider text information for improving the rating prediction. Both item specifications and user reviews have been investigated. In order to use the item specifications, CTR [40] integrates PMF [31] and Latent Dirichlet Allocation (LDA) [4] into a single framework and employs LDA to model the text. Collaborative Deep Learning (CDL)

[41] employs a hierarchical Bayesian model which jointly performs deep representation learning for the specification text content and collaborative filtering for the rating matrix. For user review texts, some research works, such as HFT [23], RMR [21], TriRank [10], and sCVR [26], integrate topic models in their frameworks to generate the latent factors for users and items incorporating review texts. Moreover, TriRank and sCVR have been explicitly claimed that they can provide explanations for recommendations. However, one common limitation of them is that their explanations are simple extractions of words or phrases from the texts. In contrast, we aim at generating concise sentences representing tips, which express the feeling of users while they are reviewing an item.

Deep Learning (DL) techniques have achieved significant success in the fields of computer vision, speech recognition, and natural language processing [8]. In the field of recommendation systems, researchers have made some attempts by combining different neural network structures with collaborative filtering to improve the recommendation performance. Salakhutdinov et al. [32] employ a class of two-layer Restricted Boltzmann Machines (RBM) with an efficient learning algorithm to model user interactions and perform collaborative filtering. Considering that the training procedure of Auto-Encoders [25] is more straightforward, some research works employ auto-encoders to tackle the latent factor modeling and rating prediction [34, 39, 44]. Recently, He et al. [11] combine generalized matrix factorization and multi-layer perceptions to find better latent structures from the user interactions for improving the performance of collaborative filtering. To model the temporal dynamic information in the user interactions, Wu et al. [43] propose a recurrent recommender network which is able to predict future behavioral trajectories.

3 FRAMEWORK DESCRIPTION

3.1 Overview

The goal of recommendation, similar to collaborative filtering, is to predict a rating given a user and an item. Additionally, in our proposed task, our model also generates abstractive tips in the form of a concise sentence. At the operational stage, only a user and an item are given. There is no given review texts and obviously no tips texts.

At the training stage, the training data consists of users, items, tips texts, and review content. Table 1 depicts the notations and key concepts used in our paper. 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 sentences. As shown in Figure 2, our framework contains two major components: neural rating regression on the left and abstractive tips generation on the right. There are two crucial latent variables : user latent factors $\mathbf{U} \in \mathbb{R}^{k_u \times m}$ and item latent factors $\mathbf{V} \in \mathbb{R}^{k_v \times n}$, where m is the number of users, and n is the number of items. k_u and k_v are the latent factor dimension for users and items respectively. For neural rating regression, given the user latent factor $\mathbf{u}$ and the item latent factor $\mathbf{v}$, a multi-layer perceptron network based regression model is employed to project $\mathbf{u}$ and $\mathbf{v}$ to a real value via several layers of non-linear transformations.

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
C	set of reviews
$\mathcal{S}$	set of tips
C_{ctx}	context for tips decoder
$\mathbf{U}$	user latent factors
$\mathbf{V}$	item latent factors
$\mathbf{E}$	word embeddings
$\mathbf{H}$	neural hidden states
$\mathbf{u}$	user latent factor
$\mathbf{v}$	item latent factor
$\mathbf{W}$	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
$tanh$	hyperbolic tangent function

For abstractive tips generation, we design a sequence decoding model based on a gated recurrent neural network called Gated Recurrent Unit (GRU) [6] to "translate" the combination of a user latent factor $\mathbf{u}$ and an item latent factor $\mathbf{v}$ into a sequence of words, representing tips. Moreover, two kinds of context information generated based on $\mathbf{u}$ and $\mathbf{v}$ are also fed into the sequence decoder model. One is the hidden variable from the rating regression component, which is used as sentiment context information. The other is the hidden output of a generative model for review texts. At the operational or testing stage, we use a beam search algorithm [13] for decoding and generating the best tips given a trained model. All the neural parameters and the latent factors for users, items, and words are learnt by a multi-task learning approach. The model can be trained efficiently by an end-to-end paradigm using back-propagation algorithms [29].

3.2 Neural Rating Regression

The aim of the neural rating regression component is to conduct representation learning for the user factor $\mathbf{u}$ and the item factor $\mathbf{v}$ mentioned above. In order to predict a rating, we need to design a model that can learn the function $f_r(\cdot)$ which can project $\mathbf{u}$ and $\mathbf{v}$ to a real-valued rating $\hat{r}$:

$$\hat{r} = f_r(\mathbf{u}, \mathbf{v}) \tag{1}$$

In most of the existing latent factor models, $f_r(\cdot)$ is represented by the inner product of $\mathbf{u}$ and $\mathbf{v}$, or adds a bias item for the corresponding user and item respectively:

$$\hat{r} = \mathbf{u}^T \mathbf{v} + b_u + b_v + b \tag{2}$$

It is obvious that the rating is calculated by a linear combination of user latent factors, item latent factors, and bias. The learnt latent

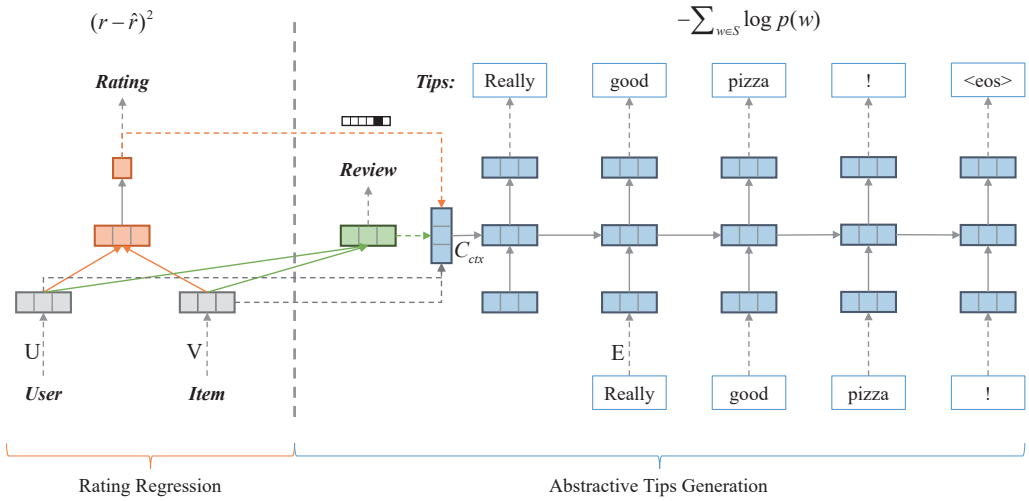


Figure 2: Our proposed framework NRT for rating regression and abstractive tips generation.

factors may not capture the complex structure implied in the user historical interactions. Recently, some research works on representation learning from different fields, such as computer vision [9, 16], natural language processing [17, 24], and knowledge base completion [36], demonstrate that non-linear transformations will enhance the representation ability. Moreover, most latent factor models assume that users and items or even text information are in the same vector space and share the same latent factors. Actually, user, item, and text information are different kinds of objects with different characteristics. Modeling them in the same vector space would lead to limitations.

As shown in left part in Figure 2, we let user latent factors $\mathbf{U} \in \mathbb{R}^{k_u \times m}$ and item latent factors $\mathbf{V} \in \mathbb{R}^{k_v \times n}$ in different vector space, where k_u and k_v are the latent factor dimension for users and items respectively. m and n are the number of users and items respectively. In order to model the relationship between users and items, one may consider to use a neural tensor network [36] to describe the interactions between users and items, such as $\mathbf{u}^T \mathbf{W} \mathbf{v}$, where $\mathbf{W} \in \mathbb{R}^{k_u \times d \times k_v}$. However, our investigation shows that such tensor network has too many parameters resulting in difficulty for handling large-scale datasets commonly found in recommendation applications. Therefore, we employ a multi-layer perceptron network to model the interactions between users and items, and map user latent factors and item latent factors into real-valued ratings.

Specifically, we first map latent factors to a shared hidden space:

$$\mathbf{h}^r = \sigma(\mathbf{W}^r_{uh}\mathbf{u} + \mathbf{W}^r_{vh}\mathbf{v} + \mathbf{b}^r_h) \tag{3}$$

where $\mathbf{W}^r_{uh} \in \mathbb{R}^{d \times k_u}$ and $\mathbf{W}^r_{vh} \in \mathbb{R}^{d \times k_v}$ are the mapping matrices for user latent factors and item latent factors respectively. $\mathbf{b}^r_h \in \mathbb{R}^d$ is the bias term. d is the dimension of the hidden vector $\mathbf{h}^r$. The superscript r refers to variables related to the rating prediction component. $\sigma(\cdot)$ is the sigmoid activation function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

This non-linear transformation can improve the performance of the rating prediction. For better performance, we can add more layers of non-linear transformations into our model:

$$\mathbf{h}^r_l = \sigma(\mathbf{W}^r_{hh_l}\mathbf{h}^r_{l-1} + \mathbf{b}^r_{h_l}) \tag{5}$$

where $\mathbf{W}^r_{hh_l} \in \mathbb{R}^{d \times d}$ 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 $\hat{r}$:

$$\hat{r} = \mathbf{W}^r_{hr}\mathbf{h}^r_L + b^r \tag{6}$$

where $\mathbf{W}^r_{hr} \in \mathbb{R}^d$ and $b^r \in \mathbb{R}$.

In order to optimize the latent factors $\mathbf{U}$ and $\mathbf{V}$, as well as all the neural parameters Θ, we formulate it as a regression problem and the loss function is formulated as:

$$\mathcal{L}^r = \frac{1}{2\,|\mathcal{X}|} \sum_{u \in \mathcal{U}, i \in \mathcal{I}} (\hat{r}_{u,i} - r_{u,i})^2 \tag{7}$$

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

3.3 Neural Abstractive Tips Generation

Generating abstractive tips only based on user latent factors and item latent factors is a challenging task. As mentioned above, abstractive tips generation is different from review content summarization and explainable topic words extraction. At the operational stage, the input only consists of a user and an item, but without any text information. After obtaining the user latent factor $\mathbf{u}$ and the item latent factor $\mathbf{v}$ from the matrices $\mathbf{U}$ and $\mathbf{V}$, we should design a strategy to "translate" these two latent vectors into a fluent sequence of words. Recently, gated recurrent neural networks such as Long Short-Term Memory (LSTM) [12] and Gated Recurrent Unit (GRU) [6] demonstrate high capability in text generation related tasks [2, 30]. Inspired by these works and 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. The right part of Figure 2 depicts our tips generation model.

The major idea of sequence modeling for tips generation can be expressed as follows:

$$p(s_t|s_1, s_2, \ldots, s_{t-1}, C_{ctx}) = \varsigma(\mathbf{h}_t^s) \tag{8}$$

where s_t is the t-th word of the tips s. C_{ctx} denotes the context information which will be described in the following sections. $\varsigma(\cdot)$ is the softmax function and defined as follows:

$$\varsigma(\mathbf{x}^{(i)}) = \frac{e^{\mathbf{x}^{(i)}}}{\sum_{k=1}^{K} e^{\mathbf{x}^{(k)}}} \tag{9}$$

$\mathbf{h}_t^s$ is the sequence hidden state at the time t and it depends on the input at the time t and the previous hidden state $\mathbf{h}_{t-1}^s$:

$$\mathbf{h}_t^s = f(\mathbf{h}_{t-1}^s, s_t) \tag{10}$$

Here $f(\cdot)$ can be the vanilla RNN, LSTM, or GRU. In the case of GRU, the state updates are processed according to the following operations:

$$\begin{aligned} \mathbf{r}_t^s &= \sigma(\mathbf{W}_{sr}^s \mathbf{s}_t + \mathbf{W}_{hr}^s \mathbf{h}_{t-1}^s + \mathbf{b}_r^s) \\ \mathbf{z}_t^s &= \sigma(\mathbf{W}_{sz}^s \mathbf{s}_t + \mathbf{W}_{hz}^s \mathbf{h}_{t-1}^s + \mathbf{b}_z^s) \\ \mathbf{g}_t^s &= \tanh(\mathbf{W}_{sh}^s \mathbf{s}_t + \mathbf{W}_{hh}^s (\mathbf{r}_t^s \odot \mathbf{h}_{t-1}^s) + \mathbf{b}_h^s) \\ \mathbf{h}_t^s &= \mathbf{z}_t^s \odot \mathbf{h}_{t-1}^s + (1 - \mathbf{z}_t^s) \odot \mathbf{g}_t^s \end{aligned} \tag{11}$$

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. $\mathbf{r}_t^s$ is the reset gate, $\mathbf{z}_t^s$ is the update gate. $\odot$ denotes element-wise multiplication. $tanh$ is the hyperbolic tangent activation function.

As shown in Figure 2, when $t = 1$, the sequence model has no input information. Therefore, we utilize the context information C_{ctx} to initialize $\mathbf{h}_0^s$. Context information is very crucial in a sequence decoding framework, which will directly affect the performance of sequence generation. In the field of neural machine translation [45], context information includes the encoding information of the source input and the decoding attention information from the source. In the field of neural summarization [19, 30], the context is the encoded document information. In our framework, the corresponding user u and item i are the input from which we design two kinds of context information for tips generation: predicted rating $\hat{r}_{u,i}$ and the generated hidden variable for the review text $\mathbf{h}_L^c$.

For the input, we just find the user latent factor and the item latent factor from the matrices $\mathbf{U}$ and $\mathbf{V}$:

$$\mathbf{u} = \mathbf{U}(:, u), \mathbf{v} = \mathbf{V}(:, i) \tag{12}$$

For the context of rating information, we can employ the output of the rating regression component in Section 3.2. Specifically, after getting the predicted rating $\hat{r}_{u,i}$, for example, $\hat{r}_{u,i} = 4.321$, we cast it into an integer 4, and add a step of vectorization. Then we get the vector representation of rating $\hat{r}_{u,i}$. If the rating range is $[0, 5]$, we will get the rating vector $\hat{\mathbf{r}}_{u,i}$:

$$\hat{\mathbf{r}}_{u,i} = (0, 0, 0, 0, 1, 0)^T \tag{13}$$

$\hat{\mathbf{r}}_{u,i}$ is used as the context information to control the sentiment of the generated tips.

Another context information is from review texts. One should note that review texts cannot be used as the input directly. The reason is that at the testing state, there are no review information. We only make use of reviews to enhance the representation ability of the latent vectors $\mathbf{U}$ and $\mathbf{V}$. We develop a standard generative model for review texts based on a multi-layer perceptron. For review content $c_{u,i}$ written by the user u to the item i, the generative process is defined as follows. We first map the user latent vector $\mathbf{u}$ and the item latent factor $\mathbf{v}$ into a hidden space:

$$\mathbf{h}^c = \sigma(\mathbf{W}_{uh}^c \mathbf{u} + \mathbf{W}_{vh}^c \mathbf{v} + \mathbf{b}_h^c) \tag{14}$$

It is obvious that we can also add more layers of non-linear transformation into the generative hidden layers. Assume that $\mathbf{h}_L^c$ is the output of the last hidden layer. We add the final generative layer to map $\mathbf{h}_L^c$ into a $|\mathcal{V}|$-size vector $\hat{\mathbf{c}}$, where $\mathcal{V}$ is the vocabulary of words in the reviews and the tips:

$$\hat{\mathbf{c}} = \varsigma(\mathbf{W}_{hc}^c \mathbf{h}_L^c + \mathbf{b}^c) \tag{15}$$

where $\mathbf{W}_{hc}^c \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{b}^c \in \mathbb{R}^{|\mathcal{V}|}$. $\varsigma(\cdot)$ is the softmax function defined in Equation 9. In fact we can regard $\hat{\mathbf{c}}$ as a multinomial distribution defined on $\mathcal{V}$. Therefore, we can draw some words from $\hat{\mathbf{c}}$ and generate the content of the review $c_{u,i}$. We let $\mathbf{c}$ be the ground truth of $c_{u,i}$. $\mathbf{c}^{(k)}$ is the term frequency of the word k in $c_{u,i}$. We employ the likelihood to evaluate the performance of this generative process. For convenience, we use the Negative Log-Likelihood (NLL) as the loss function:

$$\mathcal{L}^c = -\sum_{k=1}^{|\mathcal{V}|} \mathbf{c}^{(k)} \log \hat{\mathbf{c}}^{(k)} \tag{16}$$

One characteristic of the design of our model is that both the rating and review texts are generated from the same user latent factors $\mathbf{U}$ and item latent factors $\mathbf{V}$, i.e., $\mathbf{U}$ and $\mathbf{V}$ are shared by the subtasks of rating prediction and review text generation. Thus, in the training stage, both of $\mathbf{U}$ and $\mathbf{V}$ receive the feedback from all the subtasks, which improves the representation ability of the latent factors.

After obtaining all the context information $C_{ctx} = \{\hat{\mathbf{r}}, \mathbf{h}_L^c\}$, we integrate them into the initial decoding hidden state $\mathbf{h}_0^s$ using a non-linear transformation:

$$\mathbf{h}_0^s = tanh(\mathbf{W}_{uh}^s \mathbf{u} + \mathbf{W}_{vh}^s \mathbf{v} + \mathbf{W}_{rh}^s \hat{\mathbf{r}} + \mathbf{W}_{ch}^s \mathbf{h}_L^c + \mathbf{b}_c^s) \tag{17}$$

where $\mathbf{u}$ is the user latent factor, $\mathbf{v}$ is the item latent factor, $\hat{\mathbf{r}}$ is the vectorization for the predicted rating $\hat{r}$, and $\mathbf{h}_L^c$ is the generated hidden variable from the review text. Then GRU can conduct the sequence decoding progress. After getting all the sequence hidden states, we feed them to the final output layer to predict the word sequence in tips.

$$\hat{\mathbf{s}}_{t+1} = \varsigma(\mathbf{W}_{hs}^s \mathbf{h}_t^s + \mathbf{b}^s) \tag{18}$$

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 defined in Equation 9. Then the word with the largest probability is the decoding result for the step $t + 1$:

$$w_{t+1}^* = \arg\max_{w_i \in \mathcal{V}} \hat{\mathbf{s}}_{t+1}^{(w_i)} \tag{19}$$

At the training stage, we also use NLL as the loss function, where I_w is the vocabulary index of the word w:

$$\mathcal{L}^s = -\sum_{w \in Tips} \log \hat{\mathbf{s}}^{(I_w)} \tag{20}$$

Algorithm 1 Beam search for abstractive tips generation

Input: Beam size β, maximum length η, user id u, item id v, and tips generation model $\mathcal{G}$.
Output: β best candidate tips.

Initialize $\Pi = \emptyset$, $\pi[0:\beta-1] = \mathbf{0}$, $\Pi_p = \emptyset$, $\pi_p = \mathbf{0}$, $t = 0$;
Get user latent factor and item latent factor: $\mathbf{u} = \mathbf{U}(:,u)$ and $\mathbf{v} = \mathbf{V}(:,v)$
while $t < \eta$ **do**
  Generate β new states based on Π: $\{\hat{\mathbf{s}}_t\}_0^{\beta-1} = \mathcal{G}(\Pi)$
  for i from 0 to β **do**
    Uncompleted sequence $s_i \leftarrow \Pi(i)$
    Top-β words $\{w_0, w_1, \ldots, w_{\beta-1}\} \leftarrow \beta\text{-}\arg\max_{w_i \in \mathcal{V}} \hat{\mathbf{s}}_{t_i}^{(w_i)}$
    for each word w_j **do**
      Concatenation: $\Pi_p.inseart(s_i + w_j)$
      Likelihood: $\pi_p.inseart(\pi[i] + \log \hat{\mathbf{s}}_{t_i}^{(w_j)})$
    end for
  end for
  Get the top-β sequences with largest likelihood: $\{s\}_0^{\beta-1}, \{l\}_0^{\beta-1} = \beta\text{-}\arg\max_{s \in \Pi_p, l \in \pi_p} l$
  $\Pi \leftarrow \{s\}_0^{\beta-1}$, $\pi \leftarrow \{l\}_0^{\beta-1}$, $\Pi_p = \emptyset$, $\pi_p = \mathbf{0}$
  $t \leftarrow t + 1$
end while
return Π, π.

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

$$s^* = \arg\max_{s \in \mathcal{S}} \sum_{w \in s} \log \hat{\mathbf{s}}^{(I_w)} \tag{21}$$

The details of the beam search algorithm is shown in Algorithm 1.

3.4 Multi-task Learning

We integrate all the subtasks of rating prediction and abstractive tips generation into a unified multi-task learning framework whose objective function is:

$$\mathcal{J} = \min_{\mathbf{U},\mathbf{V},\mathbf{E},\Theta} (\lambda_r \mathcal{L}^r + \lambda_c \mathcal{L}^c + \lambda_s \mathcal{L}^s + \lambda_n (\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2 + \|\Theta\|_2^2)) \tag{22}$$

where $\mathcal{L}^r$ is the rating regression loss from Equation 7, $\mathcal{L}^c$ is the review text generation loss from Equation 16, and $\mathcal{L}^r$ is the tips generation loss from Equation 20. Θ is the set of neural parameters. λ_r, λ_c, λ_s, and λ_n are the weight proportion of each term. The whole framework can be efficiently trained using back-propagation in an end-to-end paradigm.

4 EXPERIMENTAL SETUP

4.1 Research Questions

We list the research questions we want to investigate in this paper:

- **RQ1**: What is the performance of NRT in rating prediction tasks? Does it outperform the state-of-the-art models? (See Section 5.1.)
- **RQ2**: What is the performance of NRT in abstractive tips generation? Can the generated tips express user experience and feelings? (See Section 5.2)
- **RQ3**: What is the relationship between predicted ratings and the sentiment of generated tips? (See Section 5.3)

We conduct extensive experiments to investigate the above research questions.

Table 2: Overview of the datasets.

	Books	Electronics	Movies&TV	Yelp-2016
users	603,668	192,403	123,960	684,295
items	367,982	63,001	50,052	85,533
reviews	8,887,781	1,684,779	1,697,533	2,346,227
$\lvert\mathcal{V}\rvert$	258,190	70,294	119,530	111,102

4.2 Datasets

In our experiments, we use four standard benchmark datasets from different domains to evaluate our model. The ratings of these datasets are integers in the range of $[0, 5]$. There are three datasets from Amazon 5-core[6]: **Books**, **Electronics**, and **Movies & TV**. "Books" is the largest dataset among all the domains. It contains 603,668 users, 367,982 items, and 8,887,781 reviews. We regard the field "summary" as tips, and the number of tips texts is same with the number of reviews.

Another dataset is from **Yelp** Challenge 2016[7]. It is also a large-scale dataset consisting of restaurant reviews and tips. The number of users is 684,295, which is the largest among all the datasets. Therefore this dataset is also the most sparse one. Tips are included in the dataset. For samples without tips, the first sentence of review texts is extracted and regarded as tips.

We filter out the words with low term frequency in the tips and review texts, and build a vocabulary $\mathcal{V}$ for each dataset. We show the statistics of our datasets in Table 2.

4.3 Evaluation Metrics

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:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2} \tag{23}$$

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

$$MAE = \frac{1}{N} \sum_{u,i} |r_{u,i} - \hat{r}_{u,i}| \tag{24}$$

For the evaluation of abstractive tips generation, the ground truth s_h is the tips written by the user for the item. We use *ROUGE* [20] as our evaluation metric with standard options[8]. It is a classical

[6] http://jmcauley.ucsd.edu/data/amazon
[7] https://www.yelp.com/dataset_challenge
[8] 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

evaluation metric in the field of text summarization [3, 20]. 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 co-occurring in s and s_h, then the ROUGE-N score for s is defined as follows:

$$ROUGE\text{-}N(s) = \sum_{g_n \in s_h} C_m(g_n) / \sum_{g_n \in \tilde{s}} C(g_n) \qquad (25)$$

When $\tilde{s} = s_h$, we can get $ROUGE_{recall}$, and when $\tilde{s} = s$, we get $ROUGE_{presicion}$. 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.

4.4 Comparative Methods

To evaluate the performance of rating prediction, we compare our model with the following methods:

- **RMR**: **R**atings **M**eet **R**eviews [21]. It utilizes a topic modeling technique to model the review texts and achieves significant improvements compared with other strong topic modeling based methods.
- **CTR**: **C**ollaborative **T**opic **R**egression [40]. It is a popular method for scientific articles recommendation by solving a one-class collaborative filtering problem. Note that CTR uses both ratings and item specifications.
- **NMF**: **N**on-negative **M**atrix **F**actorization [18]. It only uses the rating matrix as the input.
- **PMF**: **P**robabilistic **M**atrix **F**actorization [31]. Gaussian distribution is introduced to model the latent factors for users and items.
- **LRMF**: **L**earning to **R**ank with **M**atrix **F**actorization [35]. 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 [22]. Topic models are employed to model the user preference from a generative perspective. It still only uses the rating matrix as input.

For abstractive tips generation, we find that no existing works can generate abstractive tips purely based on latent factors of users and items. In order to evaluate the performance and conduct comparison with some baselines, we refine some existing methods to make them capable of extracting sentences for tips generation as follows.

LexRank [7] is a classical method in the field of text summarization. We add a preprocessing procedure to prepare the input texts for LexRank, which consists of the following steps: (1) Retrieval: For the user u, we first retrieve all her reviews C_u from the training set. For the item i, we use the same method to get C_i. (2) Filtering: Assuming that the ground truth rating for u and i is $r_{u,i}$, then we remove all the reviews from C_u and C_i whose ratings are not equal to $r_{u,i}$. The reviews whose words only appear in one set are also removed. (3) Tips extraction: We first merge C_u and C_i to get $C_{u,i}$, then the problem can be regarded as a multi-document summarization problem. LexRank can extract a sentence from $C_{u,i}$ as the final tips. Note that we give an advantage of this method since the ground truth ratings are used.

Table 3: Baselines and methods used for comparison.

Acronym	Gloss	Reference
NRT	Neural rating and tips generation	Section 3
Rating prediction		
RMR	Ratings meet reviews model	[21]
CTR	Collaborative topic regression model	[40]
NMF	Non-negative matrix factorization	[18]
PMF	Probabilistic matrix factorization	[31]
LRMF	List-wise learning to rank for item ranking	[35]
SVD++	Factorization meets the neighborhood	[14]
URP	User rating profile modeling using LDA	[22]
Tips generation		
LexRank	Pagerank for summarization	[7]
CTR_t	CTR for tips topic extraction	[40]
RMR_t	RMR for tips topic extraction	[21]

CTR contains a topic model component and it can generate topics for items. So the topic related variables are employed to extract tips: (1) We first get the latent factor θ_i for item i, and draw the topic z with the largest probability from θ_i. Then from ϕ_z, which is a multinomial distribution of z on $\mathcal{V}$, we select the top-50 words with the largest probability. (2) The most similar sentence from $C_{u,i}$ is extracted as the tips. This baseline is named $\mathbf{CTR}_t$. Another baseline method $\mathbf{RMR}_t$ is designed in the same way.

Finally, we list all the methods and baselines in Table 3.

4.5 Experimental Settings

Each dataset is divided into three subsets: 80%, 10%, and 10%, for training, validation, and testing, receptively. All the parameters of our model are tuned with the validation set. After the tuning process, we set the number of latent factors $k = 10$ for LRMF, NMF, PMF, and SVD++. We set the number of topics $K = 50$ for the methods using topic models. In our model NRT, we set $K = 300$ for user latent factors, item latent factors, and word latent factors. The dimension of the hidden size is 400. The number of layers for the rating regression model is 4, and for the tips generation model is 1. We set the beam size $\beta = 4$, and the maximum length $\eta = 20$. For the optimization objective, we let the weight parameters $\lambda_r = \lambda_c = \lambda_s = 1$, and $\lambda_n = 0.0001$. The batch size for mini-batch training is 200.

All the neural matrix parameters in hidden layers and RNN layers are initialized from a uniform distribution between $[-0.1, 0.1]$. Adadelta [48] is used for gradient based optimization. Our framework is implemented with Theano [38] on a single Tesla K80 GPU.

5 RESULTS AND DISCUSSIONS

5.1 Rating Prediction (RQ1)

The rating prediction results of our framework NRT and comparative models on all datasets are given in Table 4. It shows that our model consistently outperforms all comparative methods under both MAE and RMSE metrics on all datasets. From the comparison,

Table 4: MAE and RMSE values for rating prediction.

	Books		Electronics		Movies		Yelp-2016	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LRMF	1.939	2.153	2.005	2.203	1.977	2.189	1.809	2.038
PMF	0.882	1.219	1.220	1.612	0.927	1.290	1.320	1.752
NMF	0.731	1.035	0.904	1.297	0.794	1.135	1.062	1.454
SVD++	0.686	0.967	0.847	1.194	0.745	1.049	1.020	1.349
URP	0.704	0.945	0.860	1.126	0.764	1.006	1.030	1.286
CTR	0.736	0.961	0.903	1.154	0.854	1.069	1.174	1.392
RMR	0.681	0.933	0.822	1.123	0.741	1.005	0.994	1.286
NRT	**0.667***	**0.927***	**0.806***	**1.107***	**0.702***	**0.985***	**0.985***	**1.277***

*Statistical significance tests show that our method is better than RMR [21].

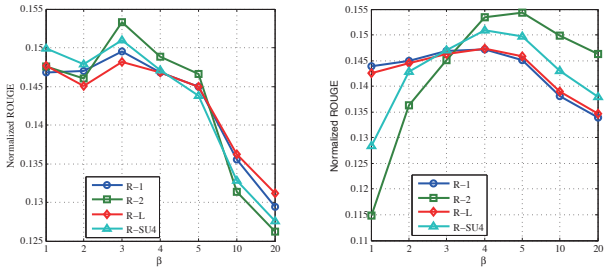
we notice that the topic modeling based methods CTR and RMR are much better than LRMF, NMF, PMF, and SVD++. The reason is that CTR and RMR consider text information such as item specifications and user reviews to improve the representation quality of latent factors, while the traditional CF-based models (e.g. LRMF, NMF, PMF, and SVD++) only consider the rating matrix as the input. Statistical significance of differences between the performance of NRT and RMR, the best comparison method, is tested using a two-tailed paired t-test. The result shows that NRT is significantly better than RMR.

Except jointly learning the tips decoder, we did not apply any sophisticated linguistic operations on the texts of reviews and tips. Jointly modeling the tips information is already very helpful for recommendation performance. In fact, tips and its corresponding rating are two facets of product assessment by a user on an item, namely, the qualitative facet and the quantitative facet. Our framework NRT elegantly captures this information with its multi-task learning model. Therefore the learnt latent factors are more effective.

5.2 Abstractive Tips Generation (RQ2)

Our NRT model can not only solve the rating prediction problem, but also generate abstractive tips simulating how users express their experience and feelings. The evaluation results of tips generation of our model and the comparative methods are given in Table 5∼Table 8. 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 the metrics of Precision and F1-measure among all the four datasets. On the dataset of Movies&TV, NRT also achieves the best Recall for all ROUGE metrics.

For most of the datasets, our NRT model does not outperform the baselines on Recall. There are several reasons: (1) The ground truth tips used in the training set are very short, only about 10-word length on average. Naturally, the model trained using this dataset cannot generate long sentence. (2) The mechanism of typical beam search algorithm makes the model favor short sentences. (3) The comparison models are extraction-based approaches and these models favor to extract long sentence, although we add a length (i.e., 20 words) restriction on them.



(a) Electronics. **(b) Movies&TV.**

Figure 3: Effectiveness of beam size β on the validation set.

We investigate the performance of different beam size β used in the beam search algorithm. The relationship between ROUGE and β on two validation sets of Electronics and Movies&TV is shown in Figure 3. We test $\beta \sim \{1 \sim 5, 10, 20\}$ and find that when $\beta = 3 \sim 5$ our model can achieve the best performance of tips generation.

Inspired by [45], we make use of Length-Normalization (LN) to adjust the log-probability in the beam search algorithm to make the beam search algorithm also consider long sentences:

$$LN(s) = \frac{(n + |s|)^{\alpha}}{(n + 1)^{\alpha}} \tag{26}$$

where s is the decoded sequence, $n = 2$, and $\alpha = 0.6$. We conduct several experiments to verify the effectiveness of LN. The comparison results are shown in Table 9, where F1-measures of ROUGE evaluation metrics are reported. It is obvious that our model NRT with LN is much better than the one without LN.

5.3 Case Analysis (RQ3)

For the purpose of analyzing the linguistic quality and the sentiment correlation between the predicted ratings and the generated tips, we selected some real cases form different domains. The results are listed in Table 10. Although our model generates tips in an abstractive way, tips' linguistic quality is quite good.

For the sentiment correlation analysis, we also choose some generated tips with negative sentiment. Take the tips "Not as good as i expected." as an example, our model predicts a rating of 2.25, which clearly shows the consistent sentiment. The ground truth

Table 5: ROUGE evaluation on dataset Books.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
LexRank	12.94	12.02	12.18	2.26	2.29	2.23	11.72	10.89	11.02	4.13	4.15	4.02
RMR_t	13.80	11.69	12.43	1.79	1.57	1.64	12.54	10.55	11.25	4.49	3.54	3.80
CTR_t	14.06	11.85	12.62	2.03	1.80	1.87	12.68	10.64	11.35	4.71	3.71	3.99
NRT	10.30	**19.28**	**12.67**	1.91	**3.76**	**2.36**	9.71	**17.92**	**11.88**	3.24	**8.03**	**4.13**

Table 6: ROUGE evaluation on dataset Electronics.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
LexRank	13.42	13.48	12.08	1.90	2.04	1.83	11.72	11.48	10.44	4.57	4.51	3.88
RMR_t	15.68	11.32	12.30	2.52	2.04	2.15	13.37	9.61	10.45	5.41	3.72	3.97
CTR_t	15.81	11.37	12.38	2.49	1.92	2.05	13.45	9.62	10.50	5.39	3.63	3.89
NRT	13.08	**17.72**	**13.95**	**2.59**	**3.36**	**2.72**	11.93	**16.01**	**12.67**	4.51	**6.69**	**4.68**

Table 7: ROUGE evaluation on dataset Movies&TV.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
LexRank	13.62	14.11	12.37	1.92	2.09	1.81	11.69	11.74	10.47	4.47	4.53	3.75
RMR_t	14.64	10.26	11.33	1.78	1.36	1.46	12.62	8.72	9.67	4.63	3.00	3.28
CTR_t	15.13	10.37	11.57	1.90	1.42	1.54	13.02	8.77	9.85	4.88	3.03	3.36
NRT	**15.17**	**20.22**	**16.20**	**4.25**	**5.72**	**4.56**	**13.82**	**18.36**	**14.73**	**6.04**	**8.76**	**6.33**

Table 8: ROUGE evaluation on dataset Yelp-2016.

Methods	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
LexRank	11.32	11.16	11.04	1.32	1.34	1.31	10.33	10.16	10.06	3.41	3.38	3.26
RMR_t	11.17	10.25	10.54	2.25	2.16	2.19	10.22	9.39	9.65	3.88	3.66	3.72
CTR_t	10.74	9.95	10.19	2.21	2.14	2.15	9.91	9.19	9.41	3.96	3.64	3.70
NRT	9.39	**17.75**	**11.64**	1.83	**3.39**	**2.22**	8.70	**16.27**	**10.74**	3.01	**7.06**	**3.78**

Table 9: Effectiveness of Length-Normalization (LN). R-* refers to ROUGE-*.

Dataset	Method	R-1	R-2	R-L	R-SU4
Electronics	NRT w/o LN	13.36	2.65	12.34	4.56
	NRT	**13.72**	**2.68**	**12.57**	**4.66**
Movies&TV	NRT w/o LN	14.86	3.72	13.76	5.46
	NRT	**15.21**	**4.00**	**13.90**	**5.71**

tips of this example is "Jack of all trades master of none. ", which also conveys a negative sentiment. One interesting observation is that its ground truth rating is the full mark 5, which we guess, may be clicked by a fat finger. Nevertheless, our model can generate a consistent sentiment between this case's rating and tips. Another generated tips "What a waste of time and money." with a negative predicted rating of 1.46 also demonstrates this property.

There are also some bad cases. For example, the predicted rating of the generated tips "Not bad for the price." is 4.34, which is a positive polarity. But the sentiment of the generated tips is neutral, consistent with the ground truth. Generally speaking, our model can achieve satisfactory performance on both rating prediction and abstractive tips generation.

6 CONCLUSIONS

We propose a deep learning based framework named **NRT** which can simultaneously predict precise ratings and generate abstractive tips with good linguistic quality simulating user experience and feelings. For abstractive tips generation, GRU with context information is employed to "translate" user and item latent factors into a concise sentence. All the neural parameters as well as the latent factors for users and items are learnt by a multi-task learning approach in an end-to-end training paradigm. Experimental results on benchmark datasets show that NRT achieves better performance than the state-of-the-art models on both tasks of rating prediction and abstractive tips generation. The generated tips can vividly predict the user experience and feelings.

Table 10: Examples of the predicted ratings and the generated tips. The first line of each group shows the generated rating and tips. The second line shows the ground truth.

Rating	Tips
4.64	***This is a great product for a great price.***
5	Great product at a great price.
4.87	***I purchased this as a replacement and it is a perfect fit and the sound is excellent.***
5	Amazing sound.
4.69	***I have been using these for a couple of months.***
4	Plenty of wire gets signals and power to my amp just fine quality wise.
4.87	***One of my favorite movies.***
5	This is a movie that is not to be missed.
4.07	***Why do people hate this film.***
4	Universal why didnt your company release this edition in 1999.
2.25	***Not as good as i expected.***
5	Jack of all trades master of none.
1.46	***What a waste of time and money.***
1	The coen brothers are two sick bastards.
4.34	***Not bad for the price.***
3	Ended up altering it to get rid of ripples.

REFERENCES

[1] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *RecSys*. ACM, 147–154.

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

[3] Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive Multi-Document Summarization via Phrase Selection and Merging. In *ACL*. 1587–1597.

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

[5] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[6] 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.

[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, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[9] 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.

[10] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*. ACM, 1661–1670.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

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

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

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.

[17] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*. 1188–1196.

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

[19] 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.

[20] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop*, Vol. 8.

[21] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*. 105–112.

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

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

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[25] Andrew Ng. 2011. Sparse autoencoder. *CS294A Lecture notes* 72 (2011), 1–19.

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

[27] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. *Introduction to recommender systems handbook*. Springer.

[28] Frank Rosenblatt. 1961. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Technical Report. DTIC Document.

[29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

[30] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

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

[32] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*. ACM, 791–798.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. ACM, 285–295.

[34] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. ACM, 111–112.

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

[36] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

[37] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.

[38] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (2016).

[39] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR* 11 (2010), 3371–3408.

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

[41] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. ACM, 1235–1244.

[42] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*. 415–423.

[43] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM*. 495–503.

[44] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. ACM, 153–162.

[45] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).

[46] Yinqing Xu, Wai Lam, and Tianyi Lin. 2014. Collaborative filtering incorporating review text and co-clusters of hidden user communities and item groups. In *CIKM*. ACM, 251–260.

[47] Yinqing Xu, Bei Shi, Wentao Tian, and Wai Lam. 2015. A Unified Model for Unsupervised Opinion Spamming Detection Incorporating Text Generality. In *IJCAI*. 725–731.

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

[49] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*. ACM, 425–434.