Generate Natural Language Explanations for Recommendation

Hanxiong Chen
Rutgers University
hanxiong.chen@rutgers.edu

Xu Chen
Tsinghua University
xu-ch14@mails.tsinghua.edu.cn

Shaoyun Shi
Tsinghua University
shisy17@mails.tsinghua.edu.cn

Yongfeng Zhang
Rutgers University
yongfeng.zhang@rutgers.edu

ABSTRACT

Providing personalized explanations for recommendations can help users to understand the underlying insight of the recommendation results, which is helpful to the effectiveness, transparency, persuasiveness and trustworthiness of recommender systems. Current explainable recommendation models mostly generate textual explanations based on pre-defined sentence templates. However, the expressiveness power of template-based explanation sentences are limited to the pre-defined expressions, and manually defining the expressions require significant human efforts.

Motivated by this problem, we propose to generate free-text natural language explanations for personalized recommendation. In particular, we propose a hierarchical sequence-to-sequence model (HSS) for personalized explanation generation. Different from conventional sentence generation in NLP research, a great challenge of explanation generation in e-commerce recommendation is that not all sentences in user reviews are of explanation purpose. To solve the problem, we further propose an auto-denoising mechanism based on topical item feature words for sentence generation. Experiments on various e-commerce product domains show that our approach can not only improve the recommendation accuracy, but also the explanation quality in terms of the offline measures and feature words coverage. This research is one of the initial steps to grant intelligent agents with the ability to explain itself based on natural language sentences.

KEYWORDS

Recommender System; Collaborative Filtering; Explainable Recommendation; Natural Language Generation

ACM Reference format:
Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate Natural Language Explanations for Recommendation. In *Proceedings of SIGIR 2019 Workshop on ExplainAble Recommendation and Search, Paris, France, July 25, 2019 (EARS'19),* 10 pages.
DOI: 10.475/123_4

EARS'19, Paris, France

DOI: 10.475/123_4

1 INTRODUCTION

Recommender systems are playing an important role in many online applications. They provide personalized suggestions to help user select the most relevant items based on their preferences. Collaborative Filtering (CF) has been one of the most successful approaches to generate recommendations based on historical user behaviors [29]. However, the recently popular latent representation approaches to CF – including both shallow or deep models – can hardly explain their rating prediction and recommendation results to users.

Researchers in very early stages have noticed that appropriate explanations are important to recommendation systems [11], which can help to improve the system effectiveness, transparency, persuasiveness and trustworthiness. As a result, researchers have looked into explainable recommendation systems in the recent years [2, 3, 8, 16, 27, 38, 39, 44, 45], which can not only provide users with the recommendation lists, but also intutive explanations about why these items are recommended.

Recommendation explanations can be provided in many different forms, and among the many, a frequently used one is textual sentence explanation. Current textual explanation generation models can be broadly classified into two categories – template-based methods and retrieval-based methods. Template-based models, such as [8, 38, 45], define one or more explanation sentence templates, and then fill different words into the templates according to the corresponding recommendation so as to generate different explanations. Such words could be, for example, item feature words that the target user is interested in. However, template-based method requires extensive human efforts to define different templates for different scenarios, and it limits the expressive power of explanation sentences to the pre-defined templates. Retrieval-based methods such as [2], on the other hand, attempt to retrieve particular sentences from user reviews as the explanations of a recommendation, which improves the expression diversity of explanation sentences. However, the explanations are limited to existing sentences and the model cannot produce new sentences for explanation.

Considering these problems, we propose to conduct explainable recommendation by generating free-text natural language explanations, meanwhile keep a high prediction accuracy. There exist three key challenges to build and evaluate a personalized natural language explanation system. 1) Data bias – the most commonly used text resources for training explainable recommender systems are user-generated reviews. Although the reviews are plentiful, informative and contain valuable information about users opinions and product features [19, 45], they can be very noisy and not all the sentences in a review are of explanation purpose. Take Figure

1 as an example, only the underlined sentence is really commenting about the product. To train a good explanation generator, our model should have the ability of auto-denoising so as to focus on the training of explanation sentences. 2) Personalization – since different users may pay attention to different product features, a good explainable recommendation system should have the ability to provide tailored explanations for different users according to the features that the user cares about. 3) Evaluation – although explainable recommendation has been widely researched in recent years, our understanding is still limited regarding which metric(s) is appropriate to evaluate the explainability of explanations. Recent research adopt readability measures in NLP (such as ROUGE score) for evaluation, but since explainability is not equivalent to readability, only generating readable sentences is not sufficient, and we need to take the effectiveness of recommendation into consideration. This problem involves deep understandings of natural language, and it also contributes technical merit to natural language processing research.

Motivated by these challenges, we propose a hierarchical sequence-to-sequence model (HSS) with auto-denoising for personalized recommendation and natural language explanation generation. In particular, the paper makes the following contributions:

• We design a hierarchical generation model, which is able to collaboratively learn over multiple sentences from different users for explanation sentence generation.

• Based on item feature words extracted from reviews, we design a feature-aware attention model to implicitly select explanation sentences from reviews for model learning, and we further introduce a feature attention model to enhance the feature-level personality of the explanations.

• We adopt three offline metrics – BLEU score, ROUGE score and feature coverage – to evaluate the quality of the generated explanations. The first two metrics are classical measures for neural machine translation and text summarization. BLEU score is precision-based while ROUGE score is relatively recall-based. They are complement to each other and it would be reasonable to report both scores to reflect the quality of machine generated text. We also use feature words coverage to show how well a model can capture the real user personalized preferences. In the meanwhile, the feature words coverage is a possible measure of the explainability of the generated explanation sentences.

In the following, we first review some related work in Section 2, and then explain the details of our framework in Section 3. We describe the three offline experimental results to verify the performance of the proposed approach in terms of rating prediction and explanation in Section 4. In Section 5, we will analyze the results and make discussions about what we learned from the experiments. Finally, we conclude this work and provide our visions of the research in Section 6.

2 RELATED WORK

Collaborative filtering (CF) [31] has been an important approach to modern personalized recommendation systems. Early collaborative filtering methods adopted intuitive and explainable methods, such as user-based [28] or item-based [30] collaborative filtering, which makes recommendation based on similar users or similar items. Later approaches to CF more and more advanced to more accurate but less transparent latent factor approaches, beginning from various matrix factorization algorithms [14, 24, 32, 34], to more recent deep learning and neural modeling approaches [9, 10, 37, 42, 43, 46]. Though effective in ranking and rating prediction, the latent nature of these approaches makes it difficult to explain the recommendation results to users, which motivated the research on explainable recommendation [44].

Researchers have explored various approaches towards model-based explainable recommendation. Since user textual reviews are informative and better reflect user preferences, a lot of research explored the possibility of incorporating user reviews to improve the recommendation quality [1, 18, 20, 41, 46] and recommendation explainability [2, 3, 8, 16, 27, 38, 45], which helps to enhance the effectiveness, transparency, trustworthiness and persuasiveness of recommendation system [11, 45].

Early approaches to explainable recommendation models generate explanations based on pre-defined explanation templates. For example, Zhang et al [45] proposed an explicit factor model (EFM), which generates explanations by filling a sentence template with the item features that a user is interested in. However, generating explanation sentences in this way needs extensive human efforts to define various templates in different scenarios. Moreover, the predefined templates limit the expressive power of explanation sentences. Li et al [16] leveraged neural rating regression and text generation to predict the user ratings and user-generated tips for recommendation, which helps to improve the prediction accuracy and the effectiveness of recommendation results. However, not all of the tips are of explanation purposes for the recommendations because they do not always explicitly comment about the product features. To alleviate the problem, Costa et al [5] attempted to train generation models based on user reviews and automatically generate fake reviews as explanations. One problem here is that not all of the sentences in the user reviews are appropriate for explanation purposes, because users may write sentences that are irrelevant to the corresponding item, which makes it difficult to generate explanations when the user reviews are too long with too much noise. Considering these deficiencies, we propose an auto-denoising mechanism for text generation and produce personalized natural language explanations for personalized recommendations.

Recently, deep neural network models have been used in various natural language processing tasks, such as question answering [47] and text summarization [25]. A well trained neural network could learn lower-dimension dense representations to capture grammatical and semantical generalizations [7]. This property of neural network is useful for natural language generation tasks. Recurrent neural network (RNN) [22] has shown notable success in sequential modeling tasks. The long short-term memory unit (LSTM) [12] and gated recurrent unit (GRU) [4] are among the most commonly used neural networks for natural language modeling to avoid the gradient vanishing problem when dealing with long sequences. A demonstration of potential utility of recurrent networks for natural language generation was provided by [33], which used a character-level LSTM model for the generation of grammatical English sentences. Character-level models can obviate the Out-of-Vocabulary(OOV) problem and reduce the vector representation

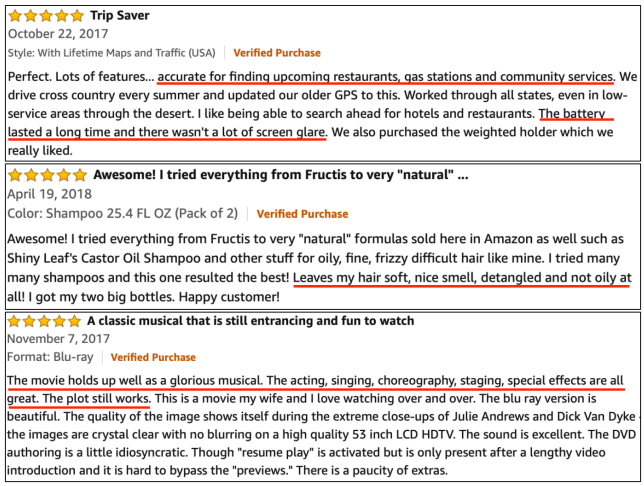


Figure 1: An example of user reviews in e-commerce. The sentences with red underlines are good for explanations.

spaces for language modeling. However, they are generally outperformed by word-level models [23]. Considering the performance of these two modeling strategies, our proposed approach, in particular, works on word-level with GRU to generate natural language explanations.

3 THE FRAMEWORK

3.1 Overview

An explainable recommender system can not only give an accurate prediction of rating score by given a user and an item, but also generate explanations to interpret the recommendation results. In our framework, we have two major modules: a rating prediction module and a natural language explanation generation module. Both modules take shared user and item latent factors as input. Since the input space is shared, the extra information can be utilized from the other module during training process to improve the general performance of our framework. During the testing stage, only user and item latent factors as well as the extracted feature words information are provided.

At the training stage, the training data consists of users, items, user generated reviews and ratings. We use $\mathcal{X}$ to represent the training dataset; $\mathcal{U}$ and $\mathcal{I}$ are user set and item set respectively; $Review$ is the set of sentences in the user generated reviews; $\mathcal{R}$ represents the set of user ratings; $\mathcal{K}$ is the feature words set, where $\mathcal{K}$ is the subset of vocabulary $\mathcal{V}$. We have $\mathcal{X} = \{\mathcal{U}, \mathcal{I}, Review, \mathcal{R}, \mathcal{K}\}$. The key notations in this paper are listed in Table 1.

In the rating regression module, only the user latent factors $\mathbf{U}$ and item latent factors $\mathbf{V}$ are given as the input. Then the multi-layer perceptron would project these latent factors into a single value as the rating prediction. After that, we calculate the mean square error loss and optimize the loss function.

In the personalized natural language explanation generation module, we design a hierarchical GRU to map the user and item latent factors into a sequence of words. The overview of our framework is shown in Figure 2. The hierarchical GRU contains a context GRU and a sentence GRU. Context GRU is used to generate the initial hidden state for sentence GRU to generate the sequence of words. The attention model is employed to improve the personalization of the generated sentences. It can be interpreted as which feature word or feature word should we pay more attention to when generating the current sentence. Since not all the words in the vocabulary are good for explanations and not all the feature words are suitable for each specific user item pair, we expect the model to learn to generate a more related and personalized explanation sentences by applying attention model. Moreover, we design an auto-denoising mechanism by applying a supervised factor on the corresponding generated sentence loss function. The key point here is that we believe the sentence with higher proportion of feature words would be more important for training the model. The effect of those sentences with less or zero proportion of feature words would automatically be weakened during training process by applying zero or very small supervised factor on their loss function.

Finally, all the neural network parameters, user and item latent factors, word embedding in both modules are learned by a multi-task learning approach. The model can be trained through back-propagation algorithms.

Table 1: A summary of key notations in this work.

Notation	Explanation
$\mathcal{X}$	training dataset
$\mathcal{U}$	user set
$\mathcal{I}$	item set
$\mathcal{V}$	vocabulary
$\mathcal{K}$	feature words set
$\mathcal{S}$	The set of generated sequences
$Review$	The set of sentences in the reviews
$\mathcal{R}$	The set of user ratings
$\mathbf{U}$	The set of user latent factors
$\mathbf{V}$	The set of item latent factors
$\mathbf{u}$	user latent factor
$\mathbf{v}$	item latent factor
$\mathbf{k}$	feature word embedding
$\mathbf{o}$	attentive feature-aware vector
Θ	The set of neural network parameters
β_i	The supervised factor of the i-th sentence
d	latent factor dimension
$r_{u,i}$	rating of user u to item i
$tanh$	hyperbolic tangent activation function
σ	sigmoid activation function
ϕ	rectified linear unit activation function
ς	softmax function

3.2 Neural Rating Regression

The goal of doing neural rating regression is to make rating predictions by given user and item latent factors. We borrow the idea from paper [16] which is to learn a function $f_r(\cdot)$ to project user latent factors $\mathbf{U}$ and item latent factors $\mathbf{V}$ to rating scores $\hat{\mathbf{r}}$. Here $f_r(\cdot)$ is represented as a multi-layer perceptron (MLP):

$$\hat{\mathbf{r}} = \text{MLP}(\mathbf{U}, \mathbf{V}) \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{d \times m}$ and $\mathbf{V} \in \mathbb{R}^{d \times n}$ are in different latent vector spaces; m is the number of users and n is the number of items; d is the latent factor dimension for both user and item representations. We first map user and item latent factors into a hidden state:

$$\mathbf{h}^r = tanh(\mathbf{W}^r_{uh}\mathbf{u} + \mathbf{W}^r_{vh}\mathbf{v} + \mathbf{b}^r_h) \tag{2}$$

where $\mathbf{W}^r_{uh} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}^r_{vh} \in \mathbb{R}^{d \times d}$; $\mathbf{b}^r_h \in \mathbb{R}^{d \times 1}$ is the bias term. We add more layers and use tanh activation function to do non-linear transformation to improve the performance of rating prediction:

$$\mathbf{h}^r_l = tanh(\mathbf{W}^r_{hh_l}\mathbf{h}^r_{l-1} + \mathbf{b}^r_{h_l}) \tag{3}$$

where $\mathbf{W}^r_{hh_l} \in \mathbb{R}^{d \times d}$ is a mapping matrix; l is the index of hidden layer. We denote the last hidden layer as $\mathbf{h}^r_L$. The output layer maps the last hidden state into a predicted rating score $\hat{r}$:

$$\hat{r} = \mathbf{W}^r_{hh_L}\mathbf{h}^r_L + b^r_{h_L} \tag{4}$$

where $\mathbf{W}^r_{hh_L} \in \mathbb{R}^{1 \times d}$. The objective function of this rating regression problem is defined as:

$$\mathcal{L}^r = \frac{1}{|\mathcal{X}|} \sum_{u \in \mathcal{U}, i \in \mathcal{I}} (\hat{r}_{u,i} - r_{u,i})^2 \tag{5}$$

$\hat{r}_{u,i}$ is the predicted rating score by user u given item i and $r_{u,i}$ is the corresponding ground truth. We can optimize this objective function to learn neural network parameters Θ as well as user and item latent representations $\mathbf{U}$ and $\mathbf{V}$.

3.3 Personalized Natural Language Explanation Generation

The key point of doing this work is to generate personalized natural language explanations. Although some research works have already implemented deep neural models to generating reviews [6] or tips [16], not many researchers work on explanation generation. In this section, we will introduce: 1) auto-denosing strategy; 2) feature-aware attention for personalized explanation generation; 3) hierarchical GRU model for sentence generation.

3.3.1 Auto-denoising. User review usually contains multiple sentences. However, not all of them are good representations of user's purchase intention. Our goal is to promote the quality of generated explanation text by introducing a supervised factor to control the training process, so that our model can learn from those more important sentences while ignoring those useless sentences. To implement this idea, we first extract all the feature words by using toolkit Sentires [1], represented as $\mathcal{K}$ and $\mathcal{K} \subseteq \mathcal{V}$, from the data set. Then the supervised factor of the i-th sentence in the review is calculated as:

$$\beta_i = \frac{N^i_k}{N^i_w} \tag{6}$$

where N^i_k is the number of feature words in the i-th sentence; N^i_w is the total number of words in the i-th sentence. We can multiply this supervised factor to the loss function of this sentence to control the training process. We believe that the sentence with higher feature words proportion would be more important. The effect of those sentences with lower or zero proportion of feature words would be automatically weakened by multiplying zero or quite small factor on their loss function.

3.3.2 Feature-aware Attention. Feature words are the words which describe the features of a product. For example, "memory", "screen", "sensitivity" can be feature words in electronics dataset. However, "use", "good", "day" are not feature words, since they are not used to describe the feature of an item. Since users may pay different attention to these feature words and each product may only relate to some feature words, inspired by [40], we implement a feature-aware attention mechanism to improve the personality. Mathematically, given a hidden state $\mathbf{h}_t$ and the $i-$th feature word embedding $\mathbf{k}_i$, the attention score of the feature word $\mathbf{k}_i$ at time t is computed as:

$$\begin{aligned} \mathbf{x}_i &= \mathbf{h}_t; \mathbf{k}_i \\ a(i, \mathbf{h}_t) &= \mathbf{w}^T_2 \phi(\mathbf{W}^a_1 \mathbf{x}_i + \mathbf{b}^a_1) + b^a_2 \end{aligned} \tag{7}$$

where $\mathbf{x}_i \in \mathbb{R}^{2d \times 1}$ is the concatenation of the hidden vector at time t and the i-th feature word vector; $\mathbf{W}^a_1 \in \mathbb{R}^{d \times 2d}$ is the mapping matrix for the first layer network; $\mathbf{b}^a_1 \in \mathbb{R}^{d \times 1}$ is the first layer bias; $\mathbf{w}_2 \in \mathbb{R}^{d \times 1}$ and $b^a_2 \in \mathbb{R}$ are the neural parameters for the second layer; $\phi(\cdot)$ is the ReLU activation function which is defined as:

$$\phi(x) = max(0, x) \tag{8}$$

The final attention weights are obtained by normalizing above attentive scores using softmax, which can be interpreted as how much attention do we pay to the feature word in term of corresponding hidden state during the training process.

$$\alpha(i, \mathbf{h}_t) = \frac{exp(a(i, \mathbf{h}_t))}{\sum_{i=1}^{|\mathcal{K}|} exp(a(i, \mathbf{h}_t))} \tag{9}$$

Finally, the attentive feature-aware vector at time t is calculated as:

$$\mathbf{o}_t = \sum_{i=1}^{|\mathcal{K}|} \alpha(i, \mathbf{h}_t)\mathbf{k}_i \tag{10}$$

This attentive feature-aware vector will be used to compute the initial hidden state for generating the i-th sentence in GRU_{wrd}, which will be introduced in the following subsection.

3.3.3 Context-level GRU (GRU_{ctx}). As shown in Figure 2, the review sentences are generated by GRU_{wrd}, which will be introduced in the next subsection. However, the initial hidden states are given by GRU_{ctx}. By leveraging this hierarchical recurrent neural network, we can generate multiple sentences by given one pair of user and item latent factors. Since each generated sentence has its own loss function, we are able to apply the auto-denoising strategy mentioned above to reduce the effects of unrelated sentences in the user generated reviews during training process.

Suppose that for each user and item pair, there are n sentences in the review. Then we have n context representations.

$$C = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$$

We use C to denote the collection of all the context representations and $\mathbf{C}_i$ denotes a specific context representation. When a sentence is generated, the context representation would be updated by the following equation:

$$\mathbf{C}_n = GRU_{ctx}(\mathbf{C}_{n-1}, \mathbf{h}^w_{n-1,L}) \tag{11}$$

[1] http://yongfeng.me/software/

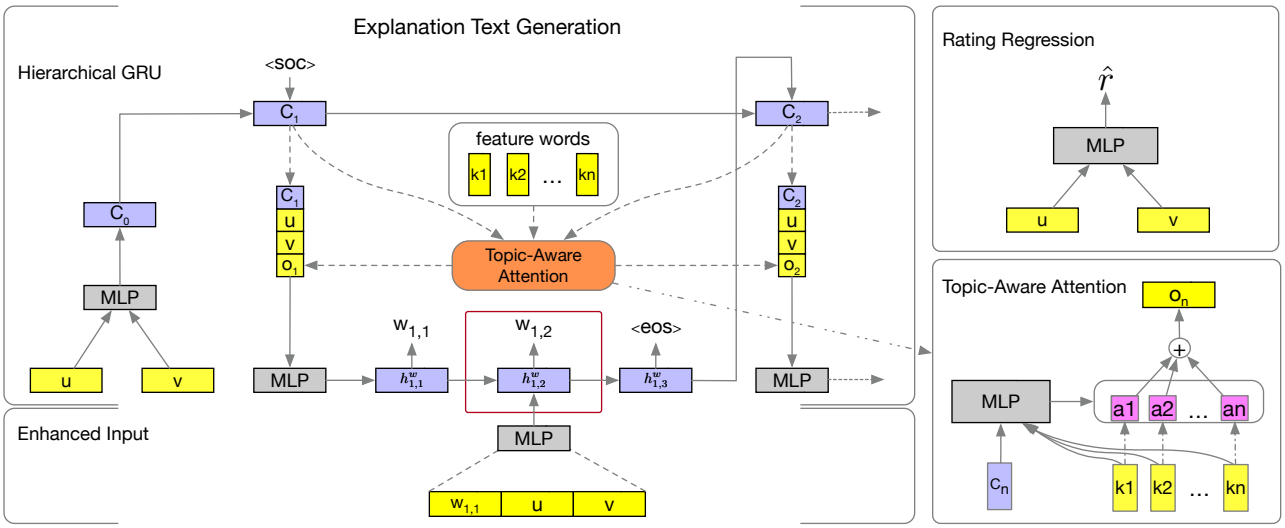


Figure 2: Overview of our HSS model. There are two major modules—explanation text generation module and rating regression module. The yellow boxes represent latent factors, such as user latent factor u, item latent factor v, word vector w; blue boxes represent hidden states; gray boxes represent multi-layer perceptron; pink boxes represent the attention weights for each feature word vector.

$\mathbf{C}_{n-1}$ is the previous context representation; $\mathbf{h}^w_{n-1,L}$ is the last hidden state of the GRU_{wrd}. Then the GRU_{ctx} state is updated by the following operations:

$$\begin{aligned}
\mathbf{r}^c_n &= \sigma(\mathbf{W}^c_{hr}\mathbf{h}^w_{n-1,L} + \mathbf{W}^c_{cr}\mathbf{C}_{n-1} + \mathbf{b}^c_r)\\
\mathbf{z}^c_n &= \sigma(\mathbf{W}^c_{hz}\mathbf{h}^w_{n-1,L} + \mathbf{W}^c_{cz}\mathbf{C}_{n-1} + \mathbf{b}^c_z)\\
\mathbf{g}^c_n &= \tanh(\mathbf{W}^c_{hg}\mathbf{h}^w_{n-1,L} + \mathbf{W}^c_{cg}(\mathbf{r}^c_n \odot \mathbf{C}_{n-1}) + \mathbf{b}^c_g)\\
\mathbf{C}_n &= \mathbf{z}^c_n \odot \mathbf{C}_{n-1} + (1 - \mathbf{z}^c_n) \odot \mathbf{g}^c_n
\end{aligned} \tag{12}$$

To start the whole process, we utilize the user latent factor $\mathbf{u}$ and item latent factor $\mathbf{v}$ to initialize the first hidden state $\mathbf{C}_0$.

$$\mathbf{C}_0 = \phi(\mathbf{W}^{c_0}_u\mathbf{u} + \mathbf{W}^{c_0}_v\mathbf{v} + \mathbf{b}^{c_0}) \tag{13}$$

3.3.4 Word-level GRU (GRU_{wrd}). This part is to generate the words for explanation sentences. The main idea can be describbed as follow:

$$p(w_{n,t}|w_{n,1}, w_{n,2}, \ldots, w_{n,t-1}) = \varsigma(\mathbf{h}^w_{n,t}) \tag{14}$$

where $w_{n,t}$ is the t-th word of the n-th review sentence. $\varsigma(\cdot)$ is the softmax function which is defined as follow:

$$\varsigma(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{15}$$

$\mathbf{h}^w_{n,t}$ is the sequence hidden state of the n-th sentence at the time t. It depends on the previous hidden state $\mathbf{h}^w_{n,t-1}$ and the current input $\mathbf{w}_{n,t}$:

$$\mathbf{h}^w_{n,t} = f(\mathbf{h}^w_{n,t-1}, \mathbf{w}_{n,t}) \tag{16}$$

The $f(\cdot)$ can be LSTM, GRU or Vanilla RNN. Here we utilize GRU for efficiency consideration. The states are updated by following operations:

$$\begin{aligned}
\mathbf{r}^w_{n,t} &= \sigma(\mathbf{W}^w_{wr}\mathbf{w}_{n,t} + \mathbf{W}^w_{hr}\mathbf{h}^w_{n,t-1} + \mathbf{b}^w_r)\\
\mathbf{z}^w_{n,t} &= \sigma(\mathbf{W}^w_{wz}\mathbf{w}_{n,t} + \mathbf{W}^w_{hz}\mathbf{h}^w_{n,t-1} + \mathbf{b}^w_z)\\
\mathbf{g}^w_{n,t} &= \tanh(\mathbf{W}^w_{wg}\mathbf{w}_{n,t} + \mathbf{W}^w_{hg}(\mathbf{r}^w_{n,t} \odot \mathbf{h}^w_{n,t-1}) + \mathbf{b}^w_g)\\
\mathbf{h}^w_{n,t} &= \mathbf{z}^w_{n,t} \odot \mathbf{h}^w_{n,t-1} + (1 - \mathbf{z}^w_{n,t}) \odot \mathbf{g}^w_{n,t}
\end{aligned} \tag{17}$$

where $\mathbf{r}^w_{n,t}$ is the reset gate; $\mathbf{z}^w_{n,t}$ is the update gate; $\odot$ represents element-wise multiplication; *tanh* denotes hyperbolic tangent activation function; $\mathbf{w}_{n,t}$ can simply to be the vector representation of the word $w_{n,t}$, which is the word in the n-th sentence in the review at time t. However, we expect to bring more personalized information into the text generation model. Inspired by [35] we concatenate word embedding of word w at time t with user embedding $\mathbf{u}$ and item embedding $\mathbf{v}$, to get an enhanced input embedding $\mathbf{s}_{n,t}$. Then we feed this embedding into a multi-layer perceptron to produce input vector $\mathbf{w}_{n,t}$:

$$\begin{aligned}
\mathbf{s}_{n,t} &= \mathbf{e}_{n,t}; \mathbf{u}; \mathbf{v}\\
\mathbf{h}_s &= \phi(\mathbf{W}_s\mathbf{s}_{n,t} + \mathbf{b}_s)
\end{aligned} \tag{18}$$

where $\mathbf{e}_{n,t}$ is the vector representation of word w in the n-th sentence at time t; $\mathbf{h}_s$ is the hidden state after doing non-linear transformation on the enhanced embedding. We can add more layers and finally feed the output of the last layer hidden state $\mathbf{h}^s_L$ into an output layer to get the input vector $\mathbf{w}_{n,t}$:

$$\mathbf{w}_{n,t} = \mathbf{W}^s_L\mathbf{h}^s_L + \mathbf{b}^s_L \tag{19}$$

where $\mathbf{W}_s \in \mathbb{R}^{d\times 3d}$, $\mathbf{W}^s_L \in \mathbb{R}^{d\times d}$; $\mathbf{b}_s$ and $\mathbf{b}^s_L$ are in $\mathbb{R}^d$.

To start the explanation sentences generation process, we need an initial hidden state. We use the output of GRU_{ctx} $\mathbf{C}_n$, user latent factor $\mathbf{u}$, item latent factor $\mathbf{v}$ and the i-th sentence feature-aware attentive context vector $\mathbf{o}_n$ together to compute the initial hidden state $\mathbf{h}^w_{n,0}$:

$$\mathbf{h}^w_{n,0} = \mathbf{W}^i_{n,2}{}^T\phi(\mathbf{W}^i_{n,1}(\mathbf{C}_n; \mathbf{u}; \mathbf{v}; \mathbf{o}_n) + \mathbf{b}^i_{n,1}) + \mathbf{b}^i_{n,2} \tag{20}$$

where $\mathbf{W}^i_{n,1} \in \mathbb{R}^{d\times 4d}$, $\mathbf{b}^i_{n,1} \in \mathbb{R}^{d\times 1}$, $\mathbf{W}^i_{n,2} \in \mathbb{R}^{d\times d}$, $\mathbf{b}^i_{n,2} \in \mathbb{R}^{d\times 1}$. The feature-aware attentive context vector $\mathbf{o}_n$ is calculated as described in subsection 3.3.2, where the hidden state $\mathbf{h}_t$ is replaced with $\mathbf{C}_n$ and the feature-aware attentive context vector is represented as $\mathbf{o}_n$ instead of $\mathbf{o}_t$. This can be interpreted as how much attention do the model pay to the feature words when generating the n-th explanation sentence. The equation (20) uses two layers neural network to calculate initial hidden state for GRU_{wrd}. You can choose to add more layers here.

By obtaining the $\mathbf{h}^w_{n,0}$, GRU can conduct the sequence decoding process. After obtaining all the hidden states of the sequences, we then feed them into a final output layer to predict the word sequence in the review.

$$\hat{\mathbf{y}}_{t+1} = \varsigma(\mathbf{W}^w_h \mathbf{h}^w_t + \mathbf{b}^w) \tag{21}$$

$\varsigma(\cdot)$ is softmax function which was defined in Equation 15; $\mathbf{h}^w_t \in \mathbb{R}^{d\times l}$ is the hidden state matrix, where l is the length of the sequence; $\mathbf{W}^w_h \in \mathbb{R}^{|\mathcal{V}|\times d}$; $\hat{\mathbf{y}}_{t+1}$ can be considered as a multinomial distribution over vocabulary $\mathcal{V}$ on review text. Then the model can generate the next word w^*_{t+1} from $\hat{\mathbf{y}}_{t+1}$ by selecting the one with the largest probability. Here we use w_i to indicate the i-th word in vocabulary. Then we have

$$w^*_{t+1} = \underset{w_i \in V}{\operatorname{argmax}} \hat{\mathbf{y}}^{(w_i)}_{t+1} \tag{22}$$

To train the model, we use Negative Log-Likelihood as the loss function. Our goal is to make the words in the review have higher probabilities than others. Here I_w is the index of word w in the vocabulary $\mathcal{V}$. The loss function of the $i-$th sentence is represented as:

$$\mathcal{L}^s_i = -\sum_{w\in Review} \log \hat{\mathbf{y}}^{(I_w)} \tag{23}$$

In the testing stage, we introduce beam search to search for the best sequence s^* with maximum log-likelihood.

$$s^* = \underset{s\in\mathcal{S}}{\operatorname{argmax}} \sum_{w\in s} \log \hat{\mathbf{y}}^{(I_w)} \tag{24}$$

$\mathcal{S}$ is the set of generated sequences. $|\mathcal{S}|$ is the beam size.

3.4 Multi-task Learning

The framework contains two major modules. We integrate both parts into one multi-task learning process. The final objective function is defined as:

$$\mathcal{J} = \min_{\mathbf{U},\mathbf{V},\mathbf{E},\Theta} \left(\mathcal{L}^r + \sum_{i=1}^{|Review|} \beta_i \mathcal{L}^s_i + \lambda(||\mathbf{U}||^2_2 + ||\mathbf{V}||^2_2 + ||\Theta||^2_2)\right) \tag{25}$$

where $\mathbf{E} \in \mathbb{R}^{d\times|\mathcal{V}|}$ is the word embedding matrix; Θ is the set of neural parameters; λ is the penalty weight; $\mathcal{L}^r$ is the rating regression loss function; β_i is the supervised factor of the i-th sentence; $\beta_i\mathcal{L}^s_i$ is the weighted loss function of the i-th generated sentence for auto-denoising.

Table 2: Statistics of the datasets in our experiments.

	Electronics	Beauty
#Users	45,224	5,122
#Items	61,687	11,616
#Reviews	744,453	90,247
#features	434	518
$\mid\mathcal{V}\mid$	20,568	7,152
sparsity	99.999%	99.998%

4 EXPERIMENTS

4.1 Datasets

Our datasets are built upon Amazon 5-core [2] [21] which includes user generated reviews and metadata spanning from May 1996 to July 2014 without duplicated records. The dataset covers 24 different categories and we select **Electronics** and **Beauty** two datasets to cover different domains and different scales in our experiment. Instead of using original 5-core version, we filter the dataset by selecting the users who has at least 10 shopping records. The reason of doing this filtering operation is that the model would not be well trained to learn the personalized preference for those users with very few reviews. After the original 5-core data is filtered, we move those records with less item frequencies into training set to get avoid of cold start issue in testing stage. For review text preprocessing, we keep all the punctuation and numbers in the raw text and we do not remove long sentences by setting a length threshold. In other words, our dataset is noisy which is challenging for text generation models. The "Electronics" dataset contains 45,224 users, 61,687 items, 744,453 reviews and 434 extracted feature words; the "Beauty" dataset is a smaller dataset which contains 5,122 users, 11,616 items, 90,247 reviews and 518 extracted feature words. The statistical details of our datasets are in Table 2.

We filter out the words with frequency lower than ten to build the vocabulary $\mathcal{V}$. Then the whole dataset is splitted into three subsets: training, validation and testing (80%/10%/10%).

4.2 Rating Regression Evaluation

4.2.1 ***Baselines.*** To evaluate the performance of rating prediction, we compare our HSS model with three methods, namely BiasedMF, SVD++ and DeepCoNN. The first two methods only utilize the ratings information and the third method involves user generated review for rating prediction.

- **BiasedMF** [14]: Biased Matrix Factorization. It only uses rating matrix to learn two low-rank user and item matrices to do rating prediction. By adding biases into plain matrix factorization model, it is able to depict the independent interaction of a user or an item on a rating value.
- **SVD++** [13]: It extends Singular Value Decomposition by integrating implicit feedback into latent factor modeling.
- **DeepCoNN**[46]: Deep Cooperative Neural Networks. This is a state-of-art deep learning method that exploits user reviews information to jointly model user and item. The

[2] http://jmcauley.ucsd.edu/data/amazon

author has shown that their model significantly outperforms some strong topic modeling based methods such as **HFT**[20], **CTR**[36] and **CDL**[37]. We use the implementation by [2] in our experiments.

4.2.2 ***Evaluation Metric.*** To evaluate the performance of rating prediction, we employ the well-known Root Mean Square Error(RMSE) as our evaluation metric. Given a rating prediction $\hat{r}_{u,i}$ and the ground truth $r_{u,i}$, the RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u \in \mathcal{U}, i \in \mathcal{V}} (r_{u,i} - \hat{r}_{u,i})^2} \tag{26}$$

where N is the total number of observations.

4.3 Explanation Sentence Generation Evaluation

4.3.1 ***Baseline.*** To evaluate the performance of text generation module, we compare our work with **Att2SeqA**[15]. This work is to automatically generate product reviews by given user, item and corresponding rating information. Their model treat user, item and rating as attributes and encode the three attributes into latent factors through multi-layer perceptron. Then the decoder would take the encoded latent factor as the initial hidden state of LSTM for review generation. In our implementation, we also use the two-level stacked LSTM for text generation as the paper proposed. There are three reasons for choosing this model as our baseline:

- *Similar input*: both our **HSS** and their **Att2SeqA** would learn user and item latent factor as the input for text generation. The difference is that their model would take rating information as the direct input while our model would learn to predict the rating score.
- *Use attention mechanism*: their model introduces attention mechanism to enhance the text generation quality while our model also uses attention model to improve the personality of the generated explanations.
- *Use review data*: both methods use user generated reviews for training the model. The difference is that their model is to learn from user written review to automatically generate fake reviews while our model is to generate explanation sentences.

Considering these three reasons, we believe that this model is the most suitable and competitive model for comparison.

There is another related model called **NRT** proposed in [16]. In that paper, they also do rating regression and text generation simultaneously. However, their goal is to generate tips. The data source used by their model is the summary in Amazon dataset. The summary can be treated as the title of a user review. It only contains one short sentence expresses the general feeling of a user to a product such as "So good", "Excellent", "I don't like it". Since the summaries or tips are too general to depict the features of an item that a user is preferred, we cannot use summaries for training an explanation generation model. The **NRT** model is very useful for simulating user feelings on a specific item. However, considering the differences from data source and the designing purposes, we would not use this model as our baseline.

4.3.2 ***Evaluation Metrics.*** We use three evaluation metrics to evaluate generated explanation sentence quality: BLEU[26], ROUGE[17] and feature words coverage.

- *BLEU*: this is a precision-based measure which is used for automatically evaluating machine generated text quality. It measures how well a machine generated text (candidate) matches a set of human reference texts by counting the percentage of n-grams in the machine generated text overlapping with the human references. The precision score for n-gram is calculated as:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{ngram \in C} Count_{clip}(ngram)}{\sum_{C' \in \{Candidates\}} \sum_{ngram' \in C'} Count(ngram')}$$

  where $Count_{clip}$ means that the count of each word in the machine generated text is truncated to not exceed the largest count observed in any single reference for that word. For more details, please refer to the paper [26].
- *ROUGE*: this is another classical evaluation metric for evaluating machine generated text quality. It is a recall-related measure which shows how much the words in the human reference texts appear in the machine generated text. The ROUGE-N is computed as:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{References\}} \sum_{ngram \in S} Count_{match}(ngram)}{\sum_{S \in \{References\}} \sum_{ngram \in S} Count(ngram)}$$

  where $Count_{match}(ngram)$ is the maximum number of n-grams co-occurring in a machine generated text and a set of human reference texts. In our experiments, we use recall, precision and F-measure of ROUGE-1(uni-gram), ROUGE-2(bi-gram), ROUGE-L(longest common subsequence) and ROUGE-SU4(skip gram) to evaluate the quality of generated explanation sentences. We use the standard option [3] for evalutaion.
- *Feature words coverage*: this measure is to reflect how well our model can capture the user personalized preferences. Assuming that the number of feature words in the human reference texts is N_r and the number of covered feature words in the machine generated sentences is N_c, the feature words coverage is calculated as:

$$Coverage_{feature} = \frac{N_r}{N_c}$$

  We use this measure to reflect how well our model generated sentences can capture the users personalized preferences. In the meanwhile, this is also the measure we use to evaluate the explainability of the generated explanation sentences.

4.4 Experimental Settings

In our HSS model, we use 300 as the dimension of user and item latent factors. The dimension of hidden size and word vector are set to 300. The number of layers for rating regression model is 4 and for explanation generation is 3. The training batch size is 100. We add gradient clip on GRU_{ctx} and GRU_{sen} by setting the norm of gradient clip to 1.0. The L2 regularization weight parameter $\lambda = 0.001$, dropout rate is 0.1. The beam size is set to 4 for both

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

Table 3: RMSE values for rating regression

	Electronics	Beauty
BiasedMF	1.096	1.030
SVD++	1.104	1.034
DeepCoNN	**1.089**	1.028
HSS	1.090	**1.027**

our model and the baseline model. All the linear layers parameter matrices are initialized from a normal distribution with mean is 0, standard deviation is 0.05. The parameter matrices in GRU are initialized with random orthogonal matrices. We set the learning rate to 0.002. The opitmizer is SGD with momentum equal to 0.9.

For the Att2SeqA model, we set the user, item and rating latent factor to 64. The hidden size and word vector size is 512. Training batch size is set to 100. The dropout rate is 0.2. Learning rate is 0.002. The optimizer is RMSprop with alpha equal to 0.95.

For both HSS and Att2SeqA, we set the length of the generated sequence to 100 but only keep the first two sentences by searching for the tag of the end of sentence "EOS". The remainder of the generated sequence would be discarded. We use these two sentences for evaluation. The reason why we do this is because a shorter explanation would be easier for users to get the point of a specific item quickly. However, if the explanation is too short, for example one sentence, that one sentence may not cover enough information to improve the recommendation quality. We think two is a reasonable good length for explanation sentences.

5 RESULTS AND DISCUSSIONS

5.1 Ratings

Our HSS model can not only generate natural language explanation sentences, but also provide predicted rating scores. The results of rating prediction of our model and baseline models are given in Table 3. It shows that our model can outperforms all the baselines on Beauty dataset. On Electronics dataset, the RMSE score of our HSS is better than BiasedMF and SVD++. Although the performance is not better than the state-of-art model DeepCoNN, the result is still comparable. In general, the topic-based deep neural network model DeepCoNN and HSS are better than tradition collaborative filtering based methods. It is because that DeepCoNN and HSS takes user reviews to improve the representation ability of user and latent factors, while the traditional methods only use rating information.

The difference between our HSS and DeepCoNN is the way of using the reivew data. In our HSS, we use GRU to learn to generate a sequence of words. The review data is used for maximizing the log likelihood of generated words. The DeepCoNN maps the user review content into a set of word embedding. Then pass the word embedding into convolution layers, max-pooling layer and fully connected layers to map the word embedding into a rating score. Although the way of using the review data is different, the experimental results on both models show that it is helpful to make use of user review information to improve the recommendation performance.

Table 4: BLEU-1 (B-1), BLEU-4 (B-4) and Feature words coverage (FC) on Electronics and Beauty dataset (in percentage)

	Electronics			Beauty		
	B-1	B-4	FC	B-1	B-4	FC
Att2SeqA	7.32	2.17	2.16	8.54	1.61	1.69
HSS	**12.36**	**4.17**	**6.74**	**9.55**	**3.49**	**6.05**

5.2 Personalized Explanation Sentence Generation Quality

In order to evaluate the quality of generated sentences, we report recall, precision and F-measure of ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-SU4. The results are shown in Table 5–6. According to the results, our model almost outperforms the baseline model on all the measures and only the recall on ROUGE-SU4 is slightly lower than the baseline model. From the results we can see that both models achieve good recall scores on all the measures except for ROUGE-2. One possible reason is that both models employ attention mechanism during the sequence generation process. The experiment result reflect that by adding attention context vector on word generation process can help to generate the sentences which are more related to the user and the product.

One difference between our model and the Att2SeqA model is that we implement the attention model by leveraging feature words. We believe that not all the feature words are related to a specific item and each user has their own preferred features. Considering this property in the e-commerce scenario, we calculate the attention weights on each of the feature word embedding with the context level hidden state on current time stamp. Then we apply the attention weights on each of the feature words embedding and integrate all the weighted embedding into a attentive context vector. This context vector represents how much attention do the model pay to each feature word when generating current sentence. However, Att2SeqA model obtains the attentive context vector with user, item and rating latent factors, which are the attributes as mentioned in the paper [15]. Then the author combines this context vector with the output of GRU on each time stamp to predict the next word. Since their attention mechanism is not for improving the feature word coverage, our model get much higher score on feature words coverage as shown in Table 4. In another word, to do attention on feature words do help the model to cover more feature words in the generated sentences.

Another observation is that our model gives much higher precision score than the baseline model. It means that our model generated sequences can hit much more words in human reference texts than those generated by Att2SeqA. As shown in Table 4, the BLEU-score, which is a precision-based metric for text generation evaluation, also gives a higher score to HSS than Att2SeqA.

5.3 Multi-sentence Generation Performance

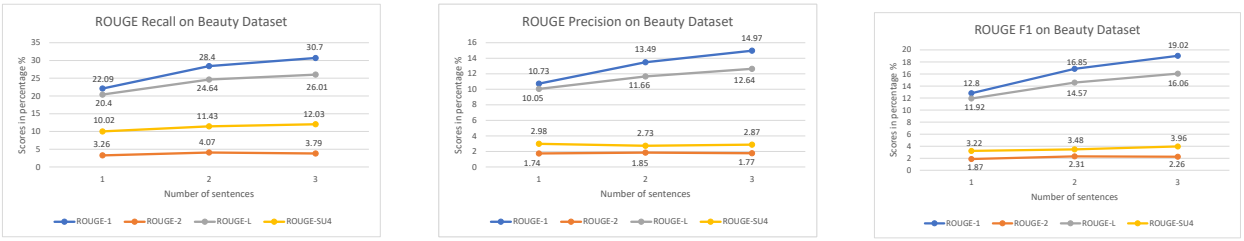
Our model has the ability of generating multiple sentences. To evaluate the multiple sentences generation quality, we do experiments on Beauty dataset. We choose the number of sentences in the range of 1 to 3 during trainig and testing stages. For example, when the number of sentences is set to 1, we only use the first sentence of each review to train the model. During the testing stage, we only generate one sentence and calculate ROUGE and BLEU score

Table 5: ROUGE score on Electronics dataset (in percentage)

	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	recall	precision	F1	recall	precision	F1	recall	precision	F1	recall	precision	F1
Att2SeqA	22.80	7.79	10.19	0.45	0.14	0.18	19.93	6.77	8.85	9.26	1.07	1.38
HSS	**26.76**	**15.72**	**18.36**	**3.01**	**1.77**	**2.05**	**22.51**	**13.31**	**15.47**	**9.69**	**3.51**	**4.10**

Table 6: ROUGE score on Beauty dataset (in percentage)

	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-SU4		
	recall	precision	F1	recall	precision	F1	recall	precision	F1	recall	precision	F1
Att2SeqA	26.55	8.67	12.03	0.70	0.19	0.27	22.96	7.57	10.46	**11.54**	1.31	1.91
HSS	**28.40**	**13.49**	**16.85**	**4.07**	**1.85**	**2.31**	**24.64**	**11.66**	**14.57**	11.43	**2.73**	**3.48**



(a) ROUGE Recall (b) ROUGE Precision (c) ROUGE F1

Figure 3: ROUGE scores change on the number of generated sentences

based on the first sentence in the human reference text. We report the changing of recall, precision and F-measure of ROUGE scores with respect to the number of sentences in Figure 3(a), 3(b) and 3(c). From the results, we can see that our model can have better recall on all the measures when to generate more than one sentence. The ROUGE precision score on multiple sentences generation is slightly lower than the one sentence case. A possible reason is that the more sentences involved in the training and testing, the more challenging for the generation model to cover the information in the human reference texts.

5.4 Case study

One thing we need to claim is that we do not do the length alignment on the review data. That means some review only contains one sentence while some of them may contains 2 or more sentences. For each sentence, the length also varies. This is a big challenge to the RNN-based sentence generation model. One reason is that the training of quite long sentences would suffer from gradient vanishing problem which would be hard for deep neural network to learning the parameters. Our hierarchical GRU model could help to solve this problem. It is because that our context level GRU could capture the long dependency so that the length of sequence for each generation process is reduced. The experiment results verify that our model has the ability of generating multiple sentences.

In Table 7 we list some generated explanations which cover the good sentences with explainability, sentence with feature word but not quite fluent, bad sentence with wrong description of the item. For the last example, the wrong description means that the item is a wireless router but the sentence is not describing the item correctly. This is a common issue we encountered during the experiments. A possible reason is that the dataset is very sparse so the corresponding item vector is not well trained, which result in the wrong description issue.

6 CONCLUSIONS AND FUTURE WORK

In this work, we proposed a deep learning framework called **HSS**, which can not only give accurate rating predictions but also generate explanation sentences to improve the effectiveness and trustworthiness of the recommender system. For rating prediction, our model can outperform the CF-based BiasedMF and SVD++ algorithms and achieve a comparable result to the state-of-art DeepCoNN model. For the explanation generation module, we design a hierarchical GRU with feature-aware attention mechanism to generate personalized explanation sentences. We also introduced an auto-denosing method to reduce the effect of unrelated sentences in trainig process. In the future, we expect to do research work to solve the wrong description issue mentioned in the previous section. We will also apply this framework on other datasets to test its robustness.

REFERENCES

[1] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 147–154.

[2] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1583–1592.

[3] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 305–314.

Table 7: Example of generated sentences. The feature words are marked in bold.

Description	Explanation Sentences
good explanation on Beauty	The **bottle** is very **light** and the **smell** is very strong.
good explanation on Electronics	The **price** is great. The **sound quality** is great
cover feature words but not fluent	The **scent** is a good **product**. I have to use this **product**. I have used to use the **hair**.
fluent but wrong description	the **price** is a great. The **sound** is great

[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[5] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. 2018. Automatic Generation of Natural Language Explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*. ACM, 57.

[6] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1. 623–632.

[7] Albert Gatt and Emiel Krahmer. 2018. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61 (2018), 65–170.

[8] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.

[9] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. *IJCAI* (2018).

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

[11] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 241–250.

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

[13] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

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

[15] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Learning through dialogue interactions by asking questions. *ICLR* (2017).

[16] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 345–354.

[17] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[18] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 105–112.

[19] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 773–782.

[20] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[21] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[22] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[23] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)* 8 (2012).

[24] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[25] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, and others. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[27] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the tenth ACM international conference on web search and data mining*. ACM, 485–494.

[28] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 175–186.

[29] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[30] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[31] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[32] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2005. Maximum-margin matrix factorization. In *Advances in neural information processing systems*. 1329–1336.

[33] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 1017–1024.

[34] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Investigation of various matrix factorization methods for large recommender systems. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 553–562.

[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] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.

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

[38] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. *arXiv preprint arXiv:1806.03568* (2018).

[39] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1543–1552.

[40] Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic Aware Neural Response Generation.. In *AAAI*, Vol. 17. 3351–3357.

[41] Yinqing Xu, Wai Lam, and Tianyi Lin. 2014. Collaborative filtering incorporating review text and co-clusters of hidden user communities and item groups. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 251–260.

[42] Shuai Zhang, Lina Yao, and Aixin Sun. 2018. Deep learning based recommender system: A survey and new perspectives. *Comput. Surveys* (2018).

[43] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1449–1458.

[44] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends in Information Retrieval* (2018).

[45] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 83–92.

[46] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 425–434.

[47] Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. *arXiv preprint arXiv:1506.06490* (2015).