Generate Neural Template Explanations for Recommendation

Lei Li
Hong Kong Baptist University
Hong Kong, China
csleili@comp.hkbu.edu.hk

Yongfeng Zhang
Rutgers University
New Brunswick, USA
yongfeng.zhang@rutgers.edu

Li Chen
Hong Kong Baptist University
Hong Kong, China
lichen@comp.hkbu.edu.hk

ABSTRACT

Personalized recommender systems are important to assist user decision-making in the era of information overload. Meanwhile, explanations of the recommendations further help users to better understand the recommended items so as to make informed choices, which gives rise to the importance of explainable recommendation research. Textual sentence-based explanation has been an important form of explanations for recommender systems due to its advantage in communicating rich information to users. However, current approaches to generating sentence explanations are either limited to predefined sentence templates, which restricts the sentence expressiveness, or opt for free-style sentence generation, which makes it difficult for sentence quality control. In an attempt to benefit both sentence expressiveness and quality, we propose a Neural Template (NETE) explanation generation framework, which brings the best of both worlds by learning sentence templates from data and generating template-controlled sentences that comment about specific features. Experimental results on real-world datasets show that NETE consistently outperforms state-of-the-art explanation generation approaches in terms of sentence quality and expressiveness. Further analysis on case study also shows the advantages of NETE on generating diverse and controllable explanations.

CCS CONCEPTS

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

KEYWORDS

Recommender Systems; Explainable Recommendation; Natural Language Generation; Neural Template Explanation

ACM Reference Format:
Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate Neural Template Explanations for Recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411992

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411992

1 INTRODUCTION

Recommender systems tackle the information overload problem by finding items that users may be interested in from a large collection of products. Over the years, many effective recommendation algorithms, such as user/item-based collaborative filtering [33, 34], matrix factorization [20, 21] and deep neural networks [15, 43, 48] have been proposed to improve recommendation accuracy. Recently, increasing attention has been paid to generating explanations for recommendations [3–5, 7–9, 12, 39, 44, 45], because it has been shown that providing explanations can help users to make better and/or faster decisions, increase the system's ease of use and enjoyment, and gain user trust in the system [36, 44]. Among various explanation styles (e.g., images [7, 18] and item neighbors [26, 32]), text explanation [3, 5, 8, 9, 12, 39, 45] has been widely studied because of the abundant textual data that online marketplace websites, such as TripAdvisor, Yelp and Amazon, could offer.

There are two major approaches to generating text explanations: *template-based* and *natural language generation*. In template-based approaches, a sentence template is usually predefined (e.g., "*You might be interested in [feature], on which this product performs well*") [45], with the slot(s) to be filled by means of matrix/tensor factorization [38, 45] or attention mechanism [12]. However, template-based approaches require manually defined sentence templates, which are expensive to create, and restrict the expressiveness of sentence explanations. For example, all items features are described as "*performs well*" in the above template, which cannot reflect the special property of different item features.

Natural language generation approach, due to its flexibility in sentence styles, has obtained research interests recently, with the goal of automatically generating flexible free-text explanations learned from user-generated contents such as user reviews. For instance, Attribute-to-Sequence (Att2Seq) [11], a state-of-the-art review generation method, produces a variety of expressions (see the example sentences in Table 1). Another typical method, Neural Rating and Tips generation (NRT) [25], aims to generate short and concise sentences. However, there are two important issues yet to be addressed in current natural language generation approaches. First, since the models are trained on user generated contents, the topics of the generated sentences may be irrelevant to the recommended item (e.g., "*I'm not sure if I need to go back*"). Second, due to the lack of variety in generative signals, a large proportion of the generated sentences may be very similar or even identical, which makes the explanations not personalized to the target users and items. These problems amount to the importance of quality control in natural language generation approaches to explainable recommendation, since poor explanations may bring negative effects to the user receptiveness of recommendations and the overall user experience in recommender systems [36].

In this work, we propose a **Ne**ural **Te**mplate (NETE) approach to explainable recommendation[1], in order to generate both expressive and high quality explanations by bridging the benefits of template and generation approaches. Essentially, it is a neural generation method, where the generated sentences are template-shaped for quality control (e.g, "*the rooftop/harbor is a great place to stay*"), because the generation process is implicitly guided by a "neural template" that are adaptive to the given features, so that more targeted and specific explanations can be generated. Table 1 shows two example sentences generated by NETE, which are more relevant to the ground truth among the comparative methods.

Technically, we propose a new recurrent neural network architecture named Gated Fusion Recurrent Unit (GFRU), which incorporates the neural templates into the explanation generation process. Specifically, the GFRU in our NETE model consists of three components: two Gated Recurrent Units (GRU) that are responsible for generating the item feature word and the sentence context words (i.e., the template), respectively, as well as a Gated Fusion Unit (GFU) that decides which GRU's word to be emitted at each time step. After the generation process, the context words will constitute the "neural template". Furthermore, to increase the variety of the generated sentence explanations, we explore a single-task learning approach instead of traditional multi-task learning approach to model training. Basically, during the training phase, the recommendation task and the explanation generation task are trained separately in a sequential manner.

We not only evaluate the generated sentence explanations in terms of traditional text quality measures (such as BLEU [31] and ROUGE [27] scores), but also evaluate how well the sentences really explain the recommendations, which is a contribution that previous work mostly ignored. To achieve this goal, we design four metrics to evaluate the generated sentences, including the ratio of unique sentences, the ratio of matched features, the ratio of feature coverage, as well as the feature diversity.

Key contributions of the paper are summarized as follows:

- We propose NETE, which can generate template-controlled explanations for both expressiveness and quality. To the best of our knowledge, this is the first work to bridge the merits of template-based and neural generation approaches to explainable recommendation.
- We explore a single-task training method instead of traditional multi-task learning methods, which helps to increase the variety of sentence generation, since the joint optimal solution may not always exist for different tasks.
- We evaluate the generated explanations in terms of both traditional sentence quality measures and measures that specifically care about the explainability of the sentences.
- Experiments on real-world datasets provide substantial evidence that NETE is capable of producing more diverse, high-quality, and controllable natural language explanations.

In the following, we will first review related work in section 2, and then introduce our neural template generation model in section 3 and section 4. Section 5 introduces the experimental setup, while interpretation of the results are provided in section 6. We conclude the work with future directions in section 7.

[1]Codes and datasets are available at https://github.com/lileipisces/NETE

Table 1: Examples of explanation sentences by state-of-the-art neural generation methods (Att2Seq [11] and NRT [25]) and our NETE method. The reference sentence is the ground-truth explanation extracted from user reviews.

Reference	They have a huge ***variety*** of things.
NRT	The food is good.
Att2Seq	I'm not sure if I need to go back.
NETE	They have a **variety** of things to choose from.
Reference	The black garlic ***ramen*** was good as well.
NRT	The food is good.
Att2Seq	The food was great.
NETE	The **ramen** was delicious.

2 RELATED WORK

There are two major directions on explainable recommendation research, i.e., human-computer interaction approaches that investigate human perception on different types of explanations [6, 13, 23], and machine learning approaches, which design algorithms to provide recommendation explanations. We mostly focus on the second approach, in particular, generating textual sentence explanations. On one hand, template-based explanation methods [12, 38, 45] have been widely used, which adopt a predefined template to create explanations. However, since the templates are predefined and fixed, the approach may hinder the diversity and flexibility of explanations. On the other hand, some retrieval-based methods [3, 4] present to users a few reviews selected from the target item's review collection as explanation. However, the selected reviews could be too long and may contain information that are irrelevant to the item or the user's interests [44], which may confuse the user. Therefore, some research works have shifted to retrieve sentences instead of the whole review as explanations [8, 39], but they are still limited to adopting existing sentences and cannot create new contents.

The above limitations motivate the development of neural generation methods. In natural language processing, the encoder-decoder framework has been widely used in different tasks, such as machine translation [10], conversational systems [30, 42], and text summarization [2]. Despite its popularity, researchers have pointed out that they tend to generate too general sentences, which lack concrete meanings and therefore are less useful to users [30, 42, 47]. To address this problem, researchers have introduced keywords [30, 42] or a group of attributes [47] as input to the generation model, so as to improve the expressiveness of the generated contents.

Besides sequence-to-sequence generation frameworks, we can also adopt table-to-text generation frameworks [41], such as review generation [11, 37, 40], tips generation [25], and explanation generation [5, 9], where the input data include users, items and ratings. Although our model's architecture is similar to these methods, its generation settings on textual data are quite different from them. Specifically, our method generates an explanation that discusses about at least one concrete feature of the item, while previous methods usually adopt a textual review [11, 37, 40], its first sentence [9], or the review title [25] for generation, which could be irrelevant to the recommended item.

Moreover, because the outputs from neural generation methods may not be easy to control [41], some works have leveraged attention mechanism [28] or copy mechanism [14] to force the models to include some specific words. However, these techniques cannot

guarantee which specific word to be included. In comparison, our NETE approach is able to place a specific feature in the generated sentence, maintaining the controllability of the explanations by learning templates to guide the generation process.

3 THE NEURAL TEMPLATE (NETE) MODEL

Our proposed **Ne**ural **Te**mplate (NETE) model consists of two modules for recommendation and explanation, respectively. An overview of the model is given in Figure 1. The goal of recommendation module is to predict a rating $\hat{r}_{u,i}$, given a user u and an item i. Meanwhile, based on a feature $f_{u,i}$ of the item that is of the user's interest, our model can generate a template-controlled sentence, which is realized by our proposed gated fusion recurrent unit (GFRU), as an explanation to the recommendation. The input feature $f_{u,i}$ could be an arbitrary feature that we want the generated explanation to talk about. Depending on the application scenario, it can be either manually set by the user u, or predicted by a feature prediction model. In section 4, we will provide a simple point-wise mutual information (PMI)-based approach for feature prediction. In summary, the training data comprise of users, items, ratings, features and explanation sentences, while during the testing stage, only users, items and features are needed.

3.1 Personalized Recommendation

Traditionally, rating prediction task is achieved by the inner product between the user and item latent factors [21], but its bi-linear nature may make it difficult to model complex user-item interactions [15]. Therefore, we adopt non-linear transformations that have been shown to have better representation ability in different fields, such as computer vision [22] and natural language processing [29]. More specifically, we employ multi-layer perceptron (MLP) with L hidden layers to capture the interactions between users and items, as shown in the left part of Figure 1. Formally, given the IDs of user u and item i, we can obtain their latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ (also called representations or embeddings), and then the recommendation module is defined as:

$$\begin{cases} \mathbf{z}_1 = \sigma(\mathbf{W}_1[\mathbf{p}_u, \mathbf{q}_i] + \mathbf{b}_1) \\ \mathbf{z}_2 = \sigma(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2) \\ \dots \\ \mathbf{z}_L = \sigma(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L) \end{cases} \quad \text{and } \hat{r}_{u,i} = \mathbf{w}_r\mathbf{z}_L + b_r \tag{1}$$

where $[\cdot,\cdot]$ denotes the concatenation of vectors, $\sigma(\cdot)$ is the sigmoid activation function, $\mathbf{W}_x \in \mathbb{R}^{2d\times 2d}$ and $\mathbf{b}_x \in \mathbb{R}^{2d}$ are weight matrices and bias vectors in the hidden layers, while $\mathbf{w}_r \in \mathbb{R}^{2d}$ and $b_r \in \mathbb{R}$ correspond to the weight and bias parameters in the final linear layer, respectively.

To minimize the difference between ground truth ratings and the predicted ones, we adopt the mean squared error loss as its objective function:

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

where $\mathcal{T}$ is the training data set, and $r_{u,i}$ denotes the ground truth rating that user u assigned to item i. In this way, the randomly initialized latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ can be updated via back-propagation.

For personalized recommendation, we can predict ratings for each user's unobserved items, and recommend the top ranked items.

3.2 Explanation Generation

The module for explanation generation (illustrated in the right part of Figure 1) is compatible with any neural recommendation module, since it only leverages the predicted ratings from the recommendation module to adjust the sentiment of the generated explanations. In the following, we first introduce the encoder-decoder framework for text generation, and then present our proposed gated fusion recurrent unit (GFRU), which is able to generate template-controlled explanations.

3.2.1 ***Encoder-decoder.*** The explanation generation problem can be formulated as a table-to-text generation task [41], where the table contains a user, an item, and possibly other attributes, e.g., a rating between the user and item. We use user u's representation $\mathbf{p}_u \in \mathbb{R}^d$ and item i's representation $\mathbf{q}_i \in \mathbb{R}^d$ as the inputs of the encoder, so that the decoded word sequence can be personalized to different user-item pairs. Moreover, we also consider the predicted rating $\hat{r}_{u,i}$, so as to enforce the sentiment control on the generated explanation. Concretely, suppose the rating scale is 1 to 5, following common practice in recommender systems and sentiment analysis, we map $\hat{r}_{u,i}$ to -1 if the rating is less than 3, otherwise +1. We then represent this sentiment using the corresponding representation $\mathbf{s}_{u,i} \in \mathbb{R}^d$ (there are only two vectors, representing positive and negative sentiment, respectively).

During the training phase, we use the sentiment associated with the given feature in a sentence instead of the rating to create the sentiment representation, because we find that the feature sentiment is more consistent with the user's perception on the item than the overall rating. We will introduce how to obtain the feature sentiments in section 4. To encode the inputs into a vector, we employ MLP with one hidden layer as the encoder,

$$\mathbf{h}_0 = \tanh(\mathbf{W}_e[\mathbf{p}_u, \mathbf{q}_i, \mathbf{s}_{u,i}] + \mathbf{b}_e) \tag{3}$$

where $\mathbf{W}_e \in \mathbb{R}^{n\times 3d}$ and $\mathbf{b}_e \in \mathbb{R}^n$ are model parameters, and $\tanh(\cdot)$ denotes the hyperbolic tangent function.

The encoded vector $\mathbf{h}_0$ is used as the initial hidden state of the decoder. Hidden states of the other time steps can be computed by recurrently feeding the word representation of the $(t-1)$-th input word $\mathbf{x}_{t-1}$ into the decoder,

$$\mathbf{h}_t = g(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) \tag{4}$$

where the hidden vector $\mathbf{h}_{t-1}$ encodes the information of previously generated words, and the decoder $g(\cdot)$ can be recurrent neural networks (RNN), long short-term memory (LSTM) networks [17], or gated recurrent units (GRU) [10]. In this paper, we adopt GRU as the decoder, because it shows competitive performance with much better computational efficiency than LSTM [25].

During decoding, the decoder recurrently produces a word based on previously generated words, which can be expressed as,

$$p(y_t|y_{<t}, \mathbf{h}_0) = \text{softmax}_{y_t}(\mathbf{W}_v\mathbf{h}_t + \mathbf{b}_v) \tag{5}$$

where $\text{softmax}(\cdot)$ denotes the softmax function, $\mathbf{W}_v \in \mathbb{R}^{|\mathcal{V}|\times n}$ and $\mathbf{b}_v \in \mathbb{R}^{|\mathcal{V}|}$ are model parameters, $y_{<t}$ represents words produced before time step t, and y_t is the word predicted at the current time step. At time step t, the decoder takes in the hidden vector $\mathbf{h}_t$ and maps it into a $|\mathcal{V}|$-sized vector, where $\mathcal{V}$ is the vocabulary of words in the dataset. This vector can be regarded as the probability

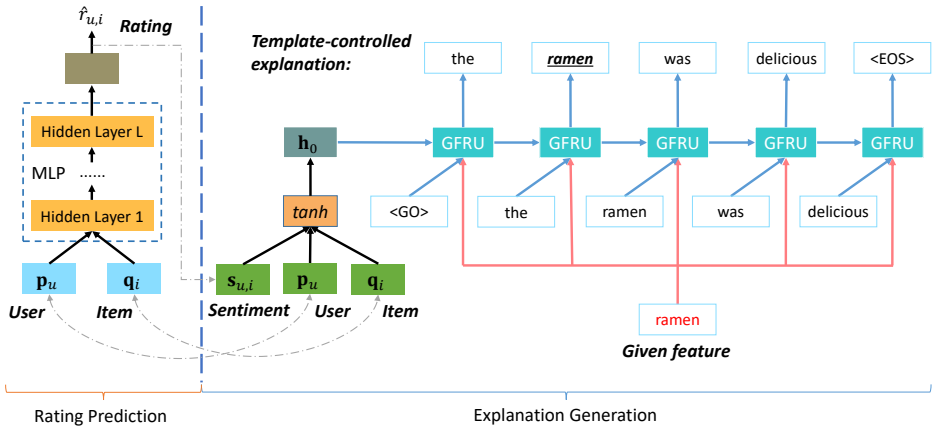

Figure 1: Overview of our proposed model NETE that consists of two basic modules for rating prediction (left) and explanation generation (right), respectively. Given a feature, the GFRU component in our model is able to generate a template-controlled explanation that contains the feature. The two modules are trained separately in a sequential manner. The gray dotted curves show the relationship between variables in the multi-task learning situation.

distribution over the vocabulary, from which a word y_t with the largest probability is sampled.

3.2.2 ***Gated Fusion Recurrent Unit.*** Although vanilla decoder can be adopted to generate explanations, its generation is uncontrollable (results and discussions can be found in Section 6). To enforce controllability to the decoder, we propose to fuse a feature into the decoding process at each time step, which is realized by our proposed Gated Fusion Recurrent Unit (GFRU). Thus, Equation (4) can be reformulated as,

$$\mathbf{h}_t = g(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_f) \tag{6}$$

where $\mathbf{x}_f$ is the representation of the feature $f \in \mathcal{F}$, and $\mathcal{F} \subset \mathcal{V}$.

Specifically, our GFRU consists of three components: two GRUs and a gated fusion unit (GFU) [1]. We treat the feature and the neural template as two types of information, so we use two GRUs to process them, respectively, which are finally merged by GFU. During explanation generation process, the *context GRU* takes the previously generated word as input, and the *feature GRU* takes the given feature at each time step, as shown in Figure 2. The GFU then combines the outputs from the two GRUs to emit a final hidden state that is used to predict the next word.

Let $\mathbf{h}_{t-1} \in \mathbb{R}^n$ be the last hidden state of our GFRU, $\mathbf{x}_{t-1} \in \mathbb{R}^d$ be the representation of the last generated word. The hidden state of *context GRU*, $\mathbf{h}_t^\alpha = g^\alpha(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$, can be computed as follows,

$$\begin{cases} \mathbf{z}_t^\alpha = \sigma(\mathbf{W}_z^\alpha[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}] + \mathbf{b}_z^\alpha) \\ \mathbf{r}_t^\alpha = \sigma(\mathbf{W}_r^\alpha[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}] + \mathbf{b}_r^\alpha) \\ \widetilde{\mathbf{h}}_t^\alpha = \tanh(\mathbf{W}_h^\alpha[\mathbf{x}_{t-1}, \mathbf{r}_t^\alpha \odot \mathbf{h}_{t-1}] + \mathbf{b}_h^\alpha) \\ \mathbf{h}_t^\alpha = \mathbf{z}_t^\alpha \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t^\alpha) \odot \widetilde{\mathbf{h}}_t^\alpha \end{cases} \tag{7}$$

where $\mathbf{W}_x^\alpha \in \mathbb{R}^{n \times (d+n)}$ and $\mathbf{b}_x^\alpha \in \mathbb{R}^n$ are model parameters, $\mathbf{z}_t^\alpha$ and $\mathbf{r}_t^\alpha$ control how much of the past information to keep and forget, respectively, and $\odot$ denotes element-wise multiplication. Accordingly, with the last hidden state of GFRU $\mathbf{h}_{t-1} \in \mathbb{R}^n$ and the representation of the feature $\mathbf{x}_f \in \mathbb{R}^d$, the hidden state of the *feature GRU* is defined as $\mathbf{h}_t^\beta = g^\beta(\mathbf{x}_f, \mathbf{h}_{t-1})$. Notice that, the two GRUs do not share parameters, so we use superscripts α and β to differentiate them.

Then, we integrate the two types of decoding information, i.e., $\mathbf{h}_t^\alpha$ and $\mathbf{h}_t^\beta$, into the final hidden state $\mathbf{h}_t$ via the GFU. The computing equations are as follows,

$$\begin{cases} \hat{\mathbf{h}}_t^\alpha = \tanh(\mathbf{W}_\alpha \mathbf{h}_t^\alpha) \\ \hat{\mathbf{h}}_t^\beta = \tanh(\mathbf{W}_\beta \mathbf{h}_t^\beta) \\ k = \sigma(\mathbf{w}_k[\hat{\mathbf{h}}_t^\alpha, \hat{\mathbf{h}}_t^\beta]) \\ \mathbf{h}_t = (1-k) \odot \mathbf{h}_t^\alpha + k \odot \mathbf{h}_t^\beta \end{cases} \tag{8}$$

where $\mathbf{W}_\alpha \in \mathbb{R}^{n \times n}$, $\mathbf{W}_\beta \in \mathbb{R}^{n \times n}$ and $\mathbf{w}_k \in \mathbb{R}^{2n}$ are parameters to be learned. From Equation (8), we can see that the automatically learned weight k controls the decoding information of the two GRUs to the output of the whole unit. When k is small, the output of the unit mainly comes from the *context GRU* for producing a template-controlled word sequence. Conversely, when it is large, the whole unit relies on *feature GRU* to fill the feature in the template. By introducing the GFRU that can include a feature in the output sequence, we can improve the controllability of the explanation generation process.

3.2.3 ***Objective Function.*** To train the module of explanation generation, we draw on the widely used cross-entropy loss as our objective function, and compute the loss for each user-item pair in the training set,

$$\mathcal{L}_e = \frac{1}{|\mathcal{T}|} \sum_{u,i \in \mathcal{T}} \frac{1}{|S_{u,i}|} \sum_{t=1}^{|S_{u,i}|} -\log p(y_t) \tag{9}$$

where $S_{u,i}$ is the ground-truth explanation for the user u and item i pair, $|S_{u,i}|$ is its length in number of words, and $p(y_t)$ denotes the predicted probability of word y_t from Equation (5).

3.3 Model Training

In general, our explainable recommendation framework (Figure 1) involves two modules for two tasks – the recommendation task and the explanation task. Previous works usually jointly train the

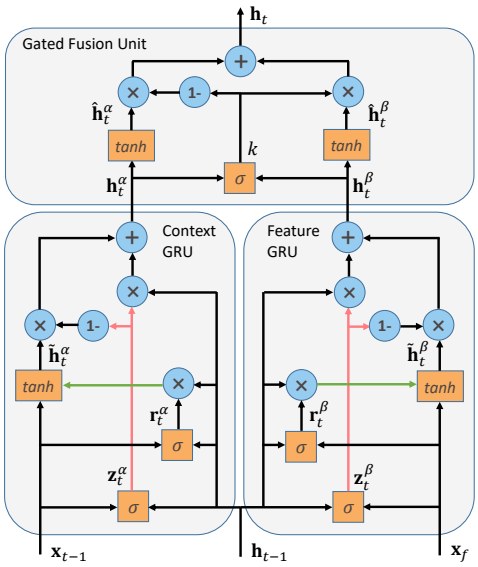

Figure 2: The structure of our proposed GFRU decoder with three components. The word at the current time step and the feature are processed by the bottom two GRUs, respectively, whose outputs are merged by the GFU component, which produces a final hidden state for the current step.

two tasks by default in a multi-task learning framework. However, little research is done on studying if and how the two tasks are compatible in a joint learning framework.

To investigate the influence of different learning methods on the recommendation and explanation tasks, we explore two different training protocols: 1) a single-task learning framework, which trains the two tasks separately in a sequential manner, and 2) a multi-task learning framework, which integrates the two tasks into a joint loss function.

For the single-task learning framework, we first optimize the objective function of the recommendation task (Eq. (2)) based on the user-item pairs in the training data. After that, we optimize the objective function of the explanation task (Eq. (9)). Notice that, since the two tasks are separated, the representations of users and items (i.e., $\mathbf{p}_u$ for user u and $\mathbf{q}_i$ for item i) in the two tasks are different sets of latent vectors. During the testing stage, the predicted ratings from the recommendation module are used as the input sentiment of the explanation task.

For the multi-task learning framework, the final objective function of the two tasks becomes:

$$\mathcal{J} = \min_{\Theta}(\lambda_r \mathcal{L}_r + \lambda_e \mathcal{L}_e + \lambda_n ||\Theta||^2) \tag{10}$$

where Θ is the set of model parameters, and λ_r, λ_e and λ_n are regularization weights for different tasks. In this case, the representations of $\mathbf{p}_u$ and $\mathbf{q}_i$ of user u and item i in the two tasks are shared (see the gray dotted curves in Figure 1).

4 FEATURE PREDICTION BASED ON PMI

As we discussed in the previous section, our NETE model can generate an explanation for a given feature. The feature could be determined in different ways according to the application scenario. For example, it can be manually specified if we require the explanation to talk about the given feature, or it can be predicted from data. In this section, we provide a simple method that can predict a feature of an item that is of interest to the target user. It particularly considers the relationship between features in the user's historical reviews and those in the target item's reviews.

We first apply a sentiment analysis toolkit [46] to extract features (i.e., aspects) and their associated sentiments from user reviews, e.g., (*rooms*, *spacious*, +1) from the sentence "*The rooms are spacious*", where *rooms* is a feature word, *spacious* is an opinion word, and +1 means that the feature-opinion pair expresses a positive sentiment. We denote the collection of all extracted features as $\mathcal{F}$. As point-wise mutual information (PMI) [30, 42] has been widely used in computational linguistics to find the association between words/features, we utilize it to predict a user's interest to a feature by measuring its relationship with the user's preferred features.

Formally, given a user's feature f_u and an item's feature f_i, the PMI is computed as:

$$\text{PMI}(f_u, f_i) = \log \frac{p(f_u, f_i)}{p(f_u)p(f_i)} = \log \frac{p(f_u|f_i)}{p(f_u)} \tag{11}$$

Then, from item i's feature set $\mathcal{F}_i$, we select the feature $\hat{f}_i$ with the highest PMI score against all the features in user u's feature set $\mathcal{F}_u$ as the predicted one, i.e., $\hat{f}_i = \text{argmax}_{f \in \mathcal{F}_i} \text{PMI}(\mathcal{F}_u, f)$, where

$$\begin{aligned} \text{PMI}(\mathcal{F}_u, f) &= \log \frac{p(\mathcal{F}_u|f)}{p(\mathcal{F}_u)} \approx \log \frac{\prod_{f' \in \mathcal{F}_u} p(f'|f)}{\prod_{f' \in \mathcal{F}_u} p(f')} \\ &= \sum_{f' \in \mathcal{F}_u} \log \frac{p(f'|f)}{p(f')} = \sum_{f' \in \mathcal{F}_u} \text{PMI}(f', f) \end{aligned} \tag{12}$$

The approximation in Eq. (12) is based on the independence assumptions of the prior distribution $p(f')$ and posterior distribution $p(f'|f)$. The two assumptions may not be true, but we use them in a pragmatic way, so that feature-level PMI on user u's feature set is additive. PMI penalizes a frequently occurring feature by dividing its prior probability, which helps us filter out less informative features for producing better explanations.

By comparing predicted features with those in the testing reviews in terms of Precision and Recall, we find that the PMI-based method performs two times better than randomly selecting features from the target item's feature set. This is as expected, because the former takes users' preferences on features into consideration when computing PMI values, while the latter arbitrarily selects features of the target item without considering these information.

5 EXPERIMENTAL SETUP

5.1 Datasets

In our experiments, we use three real-world datasets from different domains, i.e., hotel, restaurant and movie, to evaluate our proposed model. For the hotel domain, we construct the dataset with reviews crawled from a travel website TripAdvisor[2]. Specifically, we implemented a crawler that collected all the user reviews from this website on every hotel located in an international city Hong Kong. To obtain users' past interactions, the crawler subsequently collected all the historical reviews of these users. We only keep English

[2]https://www.tripadvisor.com

reviews, which gives us 2,118,108 records in total. For restaurant domain, we use the dataset from Yelp Challenge 2019[3]. This publicly available dataset consists of 6,685,900 restaurant reviews written by 1,637,138 users for 192,606 businesses located in 10 metropolitan areas. The last dataset in the movie domain is from Amazon 5-core[4] Movies & TV, which contains 1,697,533 reviews by 123,960 users for 50,052 items.

Since the three datasets are very large, we further process them by recursively removing users and items with less than 20 interactions, which results in three subsets **TA-HK**, **YELP19** and **AZ-MT**. Each review record in our datasets comprises of user ID, item ID, overall rating in the scale of 1 to 5, and textual review. After extracting features from user reviews, for each record we select one sentence from the review that contains at least one feature as the ground-truth explanation. The key characteristics of the three processed datasets are presented in Table 2.

5.2 Evaluation Metrics

To measure the recommendation accuracy of different methods, we adopt four widely used metrics: Root Mean Square Error (**RMSE**) and Mean Absolute Error (**MAE**) for rating prediction, and Normalized Discounted Cumulative Gain (**NDCG**) and Hit Ratio (**HR**) for personalized ranking. For the former two metrics, a lower value indicates a better performance, while larger values are better for the latter two.

As to the explainability, we evaluate the generated explanations from two perspectives: the relevance to ground-truth sentences and the degree of personalization. For the first perspective, we adopt two commonly used metrics, **BLEU** [31] in machine translation and **ROUGE** [27] in text summarization, to evaluate the text similarity between the generated explanation and the ground-truth. We report results of BLEU-1 and BLEU-4, and use Precision, Recall and F1 of ROUGE-1 and ROUGE-2 to measure the generated sentences in different granularities. The larger BLEU and ROUGE scores are, the closer the generated text is to the ground-truth.

For the second perspective, i.e., degree of personalization, we adopt four metrics for evaluation: the ratio of unique sentences, the ratio that our explanation matches the given feature, the feature coverage ratio in all of the generated sentences, and the feature diversity across all explanations:

1. **Unique Sentence Ratio (USR)**. As discussed before, we find that the generated sentences in previous methods tend to be highly repetitive, i.e., many user-item pairs have exactly the same explanation. To examine how severe the problem is, we present this metric to calculate how many distinct sentences are generated.

$$USR = |\mathcal{S}| / N \tag{13}$$

where $\mathcal{S}$ denotes the set of generated unique explanations, and N is the number of total explanations. Notice that, only the exactly matched sentences are considered being identical, so only one of them is added to $\mathcal{S}$.

2. **Feature Matching Ratio (FMR)**. Apart from sentence-level evaluation, we also evaluate the generated explanation at feature level. Since a feature is given as input for each user-item pair in our method, we are interested in whether it is really included in the generated explanation, which can be formulated as follows:

$$FMR = \frac{1}{N} \sum_{u,i} \delta(f_{u,i} \in \hat{S}_{u,i}) \tag{14}$$

where $\hat{S}_{u,i}$ is the generated sentence for a user-item pair, $f_{u,i}$ is the given feature, and $\delta(x) = 1$ if x is true and $\delta(x) = 0$ otherwise.

3. **Feature Coverage Ratio (FCR)**. We adopt this metric to measure the features at corpus level, i.e., how many different features are shown in the produced explanations:

$$FCR = N_g / |\mathcal{F}| \tag{15}$$

where $\mathcal{F}$ is the collection of all features in the dataset, and N_g is the number of distinct features shown in the generated explanations.

4. **Feature Diversity (DIV)**. It is reasonable that the explanations for different user-item pairs do not always discuss the same feature, so we are motivated to measure feature diversity. Let $\hat{\mathcal{F}}_{u,i}$ and $\hat{\mathcal{F}}_{u',i'}$ respectively represent two sets of features contained in two generated explanations, we can compute their intersection. For each pair of feature sets corresponding to two explanations in the testing set, the averaged size of the intersection is calculated as:

$$DIV = \frac{2}{N \times (N-1)} \sum_{u,u',i,i'} \left| \hat{\mathcal{F}}_{u,i} \cap \hat{\mathcal{F}}_{u',i'} \right| \tag{16}$$

A lower DIV indicates a smaller overlap between feature sets, and thus a higher diversity. For the others, i.e., USR, FMR and FCR, the higher the scores are, the better the performance is.

Overall, the four proposed metrics measure explanations over different perspectives, and they do not duplicate with each other. Moreover, FMR can only be used to evaluate our method since it involves the given features, while other metrics, i.e., USR, FCR and DIV, can be applied to all the explanation generation methods.

Table 2: Statistics of the datasets

	TA-HK	YELP19	AZ-MT
# of users	9,765	27,147	7,506
# of items	6,280	20,266	7,360
# of reviews	320,023	1,293,247	441,783
# of features	5,069	7,340	5,399
Avg. # of reviews / user	32.77	47.64	58.86
Avg. # of reviews / item	50.96	63.81	60.02
Avg. # of words / explanation	13.01	12.32	14.14

* **TA** and **AZ** denote **TripAdvisor** and **Amazon**, respectively.

5.3 Comparative Methods

We first introduce comparative methods to evaluate the explanations, since this is our key focus in this work. Two state-of-the-art neural generation methods, i.e., Neural Rating and Tips generation (NRT) [25], as well as Attribute-to-Sequence (Att2Seq) [11], are compared with our model. We omit other neural generation models whose input sources are different from ours, which makes them not directly comparable. For example, Visually Explainable Collaborative Filtering (VECF) [7] and Multimodal Review Generation (MRG) [37] generate descriptions based on image features, while the Neural Memory Model (NMM) [40] considers the neighborhood relation between reviews for review generation. We also provide four variants of our own model for ablation analysis.

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

- **NRT**: Neural Rating and Tips generation [25]. This model adopts multi-layer perceptron (MLP) to predict a rating based on user ID and item ID, and formulates the explanation generation problem as a text (i.e., tip) summarization task. The two tasks are integrated into a multi-task learning framework. In our implementation, the explanation sentence is used as the tip.
- **Att2Seq**: Attribute-to-sequence [11] employs MLP to encode three attributes, i.e., user, item and rating, and adopts two-layer LSTM to decode the encoded representations to generate a textual review.
- **NETE-GRU**: In this variant of our method NETE, the decoder is a standard GRU rather than our proposed GFRU. By comparing with this variant, we are able to detect whether GFRU benefits the model performance.
- **NETE-MUL**: This variant integrates the two tasks of rating prediction and text generation into a multi-task learning framework (see Equation (10)), while all the other settings are the same as our model NETE. It is to investigate whether multi-task learning is the primary factor that causes generated texts to be highly repetitive.
- **NETE-GM**: This variant differs from our model NETE in two aspects, i.e., GFRU is replaced by GRU and the model is trained in a multi-task learning framework. Its structure is inherently similar to NRT, so it is devised to show the common limitation of multi-task learning.
- **NETE-PMI**: The only difference of this variant from NETE is that the input features are predicted by the PMI method introduced in Section 4, while the standard NETE model uses the feature given by the ground-truth explanation to see if our model can generate a similar sentence to comment about the feature.

To evaluate the recommendation performance, in additon to some of the above methods, i.e., NRT, NETE-GM and NETE-MUL (NETE-GRU and NETE-PMI are excluded because their recommendation module is the same as NETE's), we compare with the following three typical rating prediction methods:

- **MF**: Matrix Factorization [21]. This method characterizes users and items by latent factors and bias terms, inferred from observed interactions.
- **SVD++**: Singular Value Decomposition [20]. This method extends MF by regarding items that a user interacted with as implicit feedback, and integrates them into the latent factor modeling.
- **DeepCoNN**: Deep Cooperative Neural Networks [48]. This method models users and items by learning feature representations from aggregated user reviews using two convolutional neural networks (CNN).

5.4 Implementation Details

We randomly split each dataset into training (80%), validation (10%) and testing (10%) sets, and ensure that there is at least one instance in the training set for each user/item. We repeat the splitting process for 5 times, and report the averaged performance on the testing set, while the validation set is used for hyper-parameters tuning.

We implemented all the methods in Python. All the neural methods, i.e., DeepCoNN, Att2Seq, NRT and NETE, are implemented by TensorFlow, and optimized by Adam [19] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. For traditional models, i.e., MF and SVD++, we set the number of latent factors to 20 (optimal choice on our datasets), and search their regularization parameters and learning rates from [0.1, 0.01, 0.001]. For DeepCoNN, we set the maximum document length of aggregated reviews to 1,000 words. For all the models that make use of text, we select top 20,000 distinct words with the largest frequency on the training set to construct the vocabulary $\mathcal{V}$. For all of the neural generation models, i.e., Att2Seq, NRT and NETE, we set the maximum length of generated text to 15, which is reasonable as the average length of explanation sentences is around 13 (as shown in Table 2). Another reason of limiting the sentence length is that presenting too much information may overwhelm the users [16, 39]. We reuse the other hyper-parameters of the baselines as reported in the original papers.

For our model NETE, the learning rate is set to 0.0001 and the batch size 128. We set d (the dimension of user/item/sentiment/word representations) as 200, n (the dimension of RNN hidden states) as 256, and L (the number of MLP layers for rating prediction) as 4. For model regularization, the dropout ratio of RNN is set to 0.2 and the regularization parameter is fixed to 0.0001.

6 RESULTS AND ANALYSIS

In this section, we first present quantitative evaluation on the generated explanations, followed by a qualitative analysis on explanation case studies. Finally, we evaluate the recommendation performance.

6.1 Quantitative Analysis on Explanations

Evaluation results of the generated explanations on three datasets are shown in Table 3. We first analyze the degree of personalization for all of the neural generation methods in terms of the four metrics, i.e., USR, FMR, FCR, and DIV.

The results show that our NETE model and its variant NETE-PMI generally perform better on all metrics, especially on USR, which measures the uniqueness of the generated sentences. When we compare NRT, NETE-GM and NETE-MUL against Att2Seq, NETE-GRU and NETE as two groups of methods, we find that the former group generates less than 1% unique sentences on testing sets, while the latter produces distinct sentences with much higher USR, because their text generation tasks are trained individually. Moreover, NRT's results on ROUGE and BLEU are as good as Att2Seq's, which evidences that ROUGE and BLEU could not properly evaluate the uniqueness of sentences. Our NETE model increases the uniqueness of generated explanations not only by changing the training paradigm into single-task learning, but also by generating template-controlled explanations via the proposed GFRU component. More specifically, our NETE approach generates 55% unique sentences on average on the three datasets, which shows the capability of our model in generating diverse explanations.

In terms of the other three metrics, all the models show a similar trend with USR. Because of the infusion of features and the GFRU design in our model, the feature coverage ratio (FCR) and feature diversity (DIV) in generated sentences are largely improved. In addition, in terms of feature matching ratio (FMR), approximately

Table 3: Performance comparison of all neural generation methods in terms of Personalization, BLEU (%) and ROUGE (%) on three datasets. For feature diversity (DIV), a lower value indicates a better performance, while for the other metrics, the larger, the better. The best performing values are boldfaced. Improvements are made by NETE over the best baseline (and * respectively indicate the statistical significance for $p < 0.01$ and $p < 0.05$ via Student's t-test). Note that BLEU, ROUGE and improvement scores in the table are percentage values (i.e., 14.2 means 14.2%), while USR, FMR, FCR and DIV scores are absolute values (i.e., 0.18 means 0.18).**

	Personalization				BLEU (%)		ROUGE-1 (%)			ROUGE-2 (%)		
	USR	FMR	FCR	DIV	BLEU-1	BLEU-4	Precision	Recall	F1	Precision	Recall	F1
	TA-HK dataset											
NRT	0.00	-	0.00	13.61	14.26	0.80	17.57	16.52	16.56	2.45	2.64	2.48
Att2Seq	0.18	-	0.17	3.93	14.76	1.01	19.26	14.45	15.83	2.43	1.96	2.06
NETE-GM	0.00	-	0.00	14.40	14.01	0.83	17.55	16.19	16.42	2.50	2.60	2.50
NETE-GRU	0.27	-	0.15	3.00	13.84	0.92	18.55	13.64	15.02	2.23	1.76	1.86
NETE-MUL	0.02	0.66	0.07	3.92	22.09	3.33	32.59	23.96	26.30	8.87	6.51	7.00
NETE-PMI	**0.79**	0.38	**0.30**	2.92	14.55	0.82	17.84	13.96	14.90	2.01	1.70	1.74
NETE	0.57**	**0.78**	0.27**	**2.22****	**22.39****	**3.66****	**35.68****	**24.86****	**27.71****	**10.20****	**6.98****	**7.66****
Improvement (%)	+210.7	-	+57.1	+77.1	+51.7	+261.3	+85.2	+50.5	+67.3	+317.0	+164.0	+209.1
	YELP19 dataset											
NRT	0.00	-	0.00	3.72	3.96	0.19	19.25	8.03	10.95	1.30	0.51	0.69
Att2Seq	0.14	-	0.12	2.19	10.41	0.59	18.20	11.38	13.21	1.81	1.16	1.31
NETE-GM	0.00	-	0.00	3.79	3.27	0.16	19.32	7.67	10.68	1.25	0.46	0.64
NETE-GRU	0.28	-	0.14	1.98	11.12	0.64	16.85	11.60	13.00	1.68	1.19	1.30
NETE-MUL	0.02	0.68	0.05	2.05	17.37	2.13	32.44	20.42	23.73	8.13	4.67	5.51
NETE-PMI	**0.64**	0.58	**0.28**	1.65	10.62	0.56	15.37	10.80	12.01	1.49	1.05	1.15
NETE	0.52**	**0.80**	0.27**	**1.48****	**19.31****	**2.69****	**33.98****	**22.51****	**25.56****	**8.93****	**5.54****	**6.33****
Improvement (%)	+257.6	-	+116.3	+48.0	+85.5	+355.3	+76.5	+97.8	+93.4	+393.1	+377.6	+384.0
	AZ-MT dataset											
NRT	0.00	-	0.01	5.46	14.02	0.57	23.57	14.24	16.87	2.53	1.70	1.92
Att2Seq	0.34	-	0.18	2.81	12.78	1.01	20.53	13.49	15.42	2.77	1.87	2.09
NETE-GM	0.00	-	0.01	4.12	12.31	0.50	22.77	13.43	16.18	2.40	1.51	1.76
NETE-GRU	0.38	-	0.11	2.34	12.10	0.95	20.16	12.93	14.93	2.63	1.75	1.97
NETE-MUL	0.05	0.61	0.03	2.63	17.20	1.94	33.79	20.01	24.17	7.50	4.32	5.16
NETE-PMI	**0.72**	0.50	**0.19**	3.06	13.02	0.82	20.93	12.76	14.99	2.36	1.63	1.81
NETE	0.57**	**0.71**	**0.19***	**1.93****	**18.76****	**2.46****	**33.87****	**21.43****	**24.81****	**7.58****	**4.77****	**5.46****
Improvement (%)	+69.1	-	+5.6	+45.2	+33.8	+143.6	+43.7	+50.5	+47.1	+174.3	+154.9	+161.2

75% of the generated sentences from our model contain the given features, which implies the good controllability of our model to comment about the given features.

Notably, we observe that NETE-PMI performs better than NETE in terms of USR and FCR. It might be because that NETE-PMI's input features are predicted ones that may not exactly match those in the testing set, so the user-item-feature combination may not be commonly seen in the training data. Therefore, the generated explanations and their contained features may be more diverse, resulting in higher USR and FCR. As for the other two metrics, i.e., FMR and DIV, NETE-PMI's performance is also competitive to the baselines. This variant shows our model's capability of dealing with unseen features, which is quite common in real-world scenarios.

Finally, we analyze results on BLEU and ROUGE. As we can see, our NETE model consistently outperforms all the baselines/variants on three datasets. We attribute this to the effectiveness of our GFRU module in generating template-controlled explanations that are more relevant to the ground-truth. By comparing NETE with the best values of NRT and Att2Seq, we see that our model improves the baselines' performance by a large margin, notably with over 100% improvements regarding BLUE-4 and ROUGE-2, which measure the similarity between generated texts and ground-truth by a sequence of n words, i.e., n-gram. This shows that our model is able to produce high-quality contents that are much closer to the ground-truth. Furthermore, we see that NRT, Att2Seq, NETE-GM and NETE-GRU generally obtain the same performance on three datasets, since they all adopt GRU for text generation. In comparison, NETE and NETE-MUL, which employ our proposed GFRU component, achieve significant performance improvements, because they can make use of the given features for explanation generation. On the other hand, NETE-PMI, which takes the features predicted by the PMI method as input, performs similar with the GRU-based methods, i.e., NRT, Att2Seq, NETE-GM and NETE-GRU, because some of the predicted features may not match the ones in the testing set, and thus the generated contents may diverge from the ground-truth.

Besides automatic evaluation, we also conduct a small-scale user survey [24], where explanations produced by our method are perceived useful by participants to explaining the recommendations.

Table 4: Example explanations generated by our NETE model on the TA-HK dataset. The first line of each group shows the ground-truth rating and explanation, while other lines show the predicted ratings, the given features, and the generated explanations, where rating < 3 denotes negative sentiment and ≥ 3 for positive sentiment. We highlight the mentioned feature in the generated text.

Rating	Feature	Explanation
4		***The rooms are spacious and the bathroom has a large tub.***
3.90	bathroom	The **bathroom** was large and had a separate shower.
	tub	The bathroom had a separate shower and **tub**.
	rooms	The **rooms** are large and comfortable.
4		***The rooms are brilliant and ideal for business travellers.***
4.13	rooms	The **rooms** are very spacious and the **rooms** are very comfortable.
2		***The broken furniture and dirty surfaces are a dead giveaway.***
2.96	furniture	The **furniture** is worn.
4		***Ideal for plane spotters and very close to the airport.***
2.76	airport	It is not close to the **airport**.

6.2 Qualitative Case Study on Explanations

Some explanations generated by different methods have already been shown in Table 1. In this subsection, we present four groups of generated explanations on the TA-HK dataset in Table 4 to show the good controllability of our NETE model in terms of controlling the explanation to talk about certain features. Results on the other two datasets show similar patterns.

As we see from the first group, when feeding our model with different features, i.e., *bathroom*, *tub* and *rooms*, the generated explanations are not only different but also highly related to the given feature, which shows that our model can generate targeted explanations for specific features. By comparing the last generated sample in the first group with that in the second group (both about feature *rooms* but the user-item pairs are different), the model generates explanations that describe the same feature in different expressions, which shows that our model is able to produce different explanations personalized for different user-item pairs. The last two groups with predicted ratings lower than 3, which corresponds to negative sentiment, show that our model is capable of taking into account the sentiment of the predicted ratings for explanation generation. These observations manifest the controllability of our model in terms of generating explanations corresponding to the given user, item, feature and sentiment.

Moreover, there exist common expressions in the generated sentences, e.g., __ *was large/comfortable*, which constitute templates learned from data instead of manually defined ones. This also shows that our model generates template-controlled explanations that can automatically adapt to the input features. Overall, the intuitive linguistic quality of our generated explanations is satisfactory.

6.3 Recommendation Performance

The recommendation performance of our model and baselines is shown in Table 5. As it can be seen, the accuracy of all the methods on rating prediction task in terms of both RMSE and MAE is close, while the performance gap between them on the personalized ranking (i.e., top-N recommendation) task with regard to NDCG and HR widens, because the former task only evaluates a small proportion of unobserved items, which may cause selection bias in the data [35]. In the following, we focus on the results of the ranking task.

The observations on three datasets are consistent. Our NETE model performs significantly better than all the baselines and its variants. In particular, NRT and two variants of NETE, i.e., NETE-GM and NETE-MUL, generally achieve the same performance, because they share similar architectures, i.e., training the rating prediction and explanation generation tasks in a multi-task learning framework. In comparison, NETE trains the two tasks separately (i.e., single-task learning) for the sake of generating diverse explanations as discussed before. It actually shows the advantage of single-task learning, and that explanation generation may disturb the recommendation performance in multi-task learning. We also see that review-based methods (NETE, NRT and DeepCoNN) generally perform better than rating-only methods (MF and SVD++), which shows the advantage of incorporating rich context information to improve the recommendation performance.

7 CONCLUSIONS AND FUTURE WORK

In this work, we aim to improve both the expressiveness and the quality of recommendation explanations. To achieve this goal, we proposed NETE – a neural template explanation generation framework that bridges the benefits of template-based approaches and generation approaches. We not only evaluate the generated explanations based on traditional text quality measures such as BLEU and ROUGE, but also on innovative metrics that evaluate the uniqueness, matched features, feature coverage, and feature diversity of the explanations. Experimental results show that our approach is highly controllable to generate explanations about the given user, item, sentiment, and features. In the future, we will further consider adjective words that modify the features to increase the expressiveness of the generated explanations. Moreover, since it would be helpful for an explanation to discuss multiple features, we will extend our framework to generate multi-feature explanations.

ACKNOWLEDGMENTS

This work was partially supported by HKBU IRCMS Project (IRCMS/19-20/D05) and NSF IIS-1910154. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. 2017. Gated multimodal units for information fusion. In *ICLR Workshop*.
[2] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*. 152–161.
[3] Rose Catherine and William Cohen. 2017. Transnets: Learning to Transform for Recommendation. In *RecSys*. ACM, 288–296.
[4] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW*. ACM, 1583–1592.

Table 5: Performance comparison of all methods in terms of RMSE, MAE, NDCG@5 (‰) and HR@5 (‰). The best performing values are boldfaced. Improvements are made by NETE over the best baseline (* indicates the statistical significance for $p < 0.05$ via Student's t-test).

	Rating Prediction						Personalized Ranking					
	TA-HK		YELP19		AZ-MT		TA-HK		YELP19		AZ-MT	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5	HR@5
	Baselines											
MF	0.798	0.613	1.011	0.782	0.963	0.719	0.361	0.559	0.116	0.140	0.449	0.416
SVD++	0.798	0.610	1.011	0.785	0.965	0.718	0.362	0.553	0.116	0.138	0.443	0.350
DeepCoNN	0.796	0.607	1.011	0.789	0.959	0.721	0.630	0.963	0.225	0.216	1.044	1.096
NRT	0.792	**0.605**	**1.007**	0.783	0.957	0.718	0.687	1.074	0.218	0.218	1.305	1.178
	Ours											
NETE-GM	0.793	0.606	1.008	0.785	0.957	**0.713**	0.719	1.119	0.281	0.288	1.616	1.452
NETE-MUL	**0.790**	0.608	1.008	**0.781**	**0.956**	0.717	0.594	0.915	0.234	0.246	1.587	1.507
NETE	0.792	0.608	1.010	0.789	0.961	0.727	**1.039***	**1.509***	**0.484***	**0.515***	**1.671***	**1.578***
Improvement (%)	-	-	-	-	-	-	+51.2	+40.5	+115.1	+136.2	+28.0	+34.0

[5] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate Natural Language Explanations for Recommendation. In *SIGIR Workshop EARS*.

[6] Li Chen and Feng Wang. 2017. Explaining recommendations based on feature sentiments in product reviews. In *IUI*. ACM, 17–28.

[7] Xu Chen, Han Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized Fashion Recommendation with Visual Explanations based on Multimodal Attention Network: Towards Visually Explainable Recommendation. In *SIGIR*. ACM, 765–774.

[8] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic Explainable Recommendation based on Neural Attentive Models. In *AAAI*.

[9] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-Attentive Multi-Task Learning for Explainable Recommendation. In *IJCAI*.

[10] 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. In *EMNLP*. 1724–1734.

[11] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to Generate Product Reviews from Attributes. In *EACL*, Vol. 1. 623–632.

[12] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable Recommendation Through Attentive Multi-View Learning. In *AAAI*.

[13] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should I explain? A comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies* 72, 4 (2014), 367–382.

[14] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. ACM, 173–182.

[16] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *CSCW*. ACM, 241–250.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] Min Hou, Le Wu, Enhong Chen, Zhi Li, Vincent W Zheng, and Qi Liu. 2019. Explainable Fashion Recommendation: A Semantic Attribute Region Guided Approach. In *IJCAI*.

[19] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 8 (2009), 30–37.

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

[23] Johannes Kunkel, Tim Donkers, Lisa Michael, Catalin-Mihai Barbu, and Jürgen Ziegler. 2019. Let Me Explain: Impact of Personal and Impersonal Explanations on Trust in Recommender Systems. In *CHI*. ACM, 487.

[24] Lei Li, Li Chen, and Yongfeng Zhang. 2020. Towards Controllable Explanation Generation for Recommender Systems via Neural Template. In *WWW Demo*.

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

[26] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. 2020. Directional and Explainable Serendipity Recommendation. In *WWW*. 122–132.

[27] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[28] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*. 1412–1421.

[29] 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.

[30] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*. 3349–3358.

[31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*. 311–318.

[32] Georgina Peake and Jun Wang. 2018. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *KDD*. ACM.

[33] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*. ACM, 175–186.

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

[35] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *RecSys*. 213–220.

[36] Nava Tintarev and Judith Masthoff. 2015. Explaining Recommendations: Design and Evaluation. In *Recommender Systems Handbook* (2 ed.), Bracha Shapira (Ed.). Springer, Chapter 10, 353–382.

[37] Quoc-Tuan Truong and Hady Lauw. 2019. Multimodal Review Generation for Recommender Systems. In *WWW*. ACM, 1864–1874.

[38] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. In *SIGIR*. ACM.

[39] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A Reinforcement Learning Framework for Explainable Recommendation. In *ICDM*.

[40] Zhongqing Wang and Yue Zhang. 2017. Opinion Recommendation using Neural Memory Model. In *EMNLP*. 1627–1638.

[41] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. In *EMNLP*. 3174–3187.

[42] Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *EMNLP*. 2190–2199.

[43] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[44] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[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 *SIGIR*. ACM, 83–92.

[46] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review? Boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*. ACM, 1027–1030.

[47] Lujun Zhao, Kaisong Song, Changlong Sun, Qi Zhang, Xuanjing Huang, and Xiaozhong Liu. 2019. Review Response Generation in E-Commerce Platforms with External Product Information. In *WWW*. ACM, 2425–2435.

[48] 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.