A Reinforcement Learning Framework for Explainable Recommendation

Xiting Wang
Microsoft Research Asia
xitwan@microsoft.com

Yiru Chen
Peking University
chen1ru@pku.edu.cn

Jie Yang
Tsinghua University
yangj16@mails.tsinghua.edu.cn

Le Wu
Hefei University of Technology
lewu@hfut.edu.cn

Zhengtao Wu
University of Science and Technology of China
wzt@mail.ustc.edu.cn

Xing Xie
Microsoft Research Asia
xing.xie@microsoft.com

***Abstract*—Explainable recommendation, which provides explanations about why an item is recommended, has attracted increasing attention due to its ability in helping users make better decisions and increasing users' trust in the system. Existing explainable recommendation methods either ignore the working mechanism of the recommendation model or are designed for a specific recommendation model. Moreover, it is difficult for existing methods to ensure the presentation quality of the explanations (e.g., consistency).**

To solve these problems, we design a reinforcement learning framework for explainable recommendation. Our framework can explain any recommendation model (model-agnostic) and can flexibly control the explanation quality based on the application scenario. To demonstrate the effectiveness of our framework, we show how it can be used for generating sentence-level explanations. Specifically, we instantiate the explanation generator in the framework with a personalized-attention-based neural network. Offline experiments demonstrate that our method can well explain both collaborative filtering methods and deep-learning-based models. Evaluation with human subjects shows that the explanations generated by our method are significantly more useful than the explanations generated by the baselines.

***Keywords*-Explainable recommendation, reinforcement learning, personalized explanation, attention networks**

I. Introduction

Billions of online users leverage recommender systems to sift through massive contents and make decisions related to their personal lives. As recommender systems impact people's lives in increasingly profound ways, there is a growing need to ensure that the users understand and trust the system [1]. Explanations about why the items are recommended, which serve as a bridge between recommender systems and users, have been proved to play an important role in recommender systems. Good explanations may increase users' trust in the system [2], help users make better decisions [3], and persuade users to try or buy an item [4].

Current explainable recommendation methods can be divided into two groups. *Post-hoc* methods [3], [5], [6] generate explanations after the items have been recommended. They usually choose explanations from a pre-defined set of candidates, such as "people also bought" [5] and "7 of your friends like this" [6]. While the explanations are usually persuasive and highly readable, the working mechanism of the recommendation model is often ignored and the diversity of the explanations is often limited by the number of predefined candidates. *Embedded methods* [7]–[13] incorporate the explanation process into the construction of a recommendation model. The explanations are often pieces of text [13] or images [14] selected from the side information of the items. Usually, the pieces that contribute most to the recommendation accuracy are selected. While the produced explanations are diversified and closely tied to the recommendation model, it is difficult to ensure the readability and consistency of these explanations. For example, it is difficult to ensure that the sentiment in a selected piece of text is consistent with the rating. Also, we need to design different explanation methods for different recommendation models.

Based on the previous discussion, we summarize three desirable properties for an explainable recommendation method.

- **Model-agnostic**. The method can be used to explain any recommendation model. This allows us to explain complex deep-learning-based model and provides flexibility to explain future recommendation models.
- **Model explainability**. The explanations should be tied closely to the recommendation model and reveal its working mechanism. This increases the explainability of the recommendation model [15].
- **Explanation quality control**. Ensuring the quality of the explanations (e.g., their readability, consistency, and diversity) is very important in real-world applications. It is desirable that the method can flexibly control the explanation quality based on the application scenario.

Although post-hoc methods are model-agnostic, they do not consider model explainability and have problems in generating diversified results (quality control). While the embedded methods have good model explainability, they are not model-agnostic and are not good at explanation quality control.

We solve the aforementioned problems by designing a reinforcement learning framework for explainable recommendation. Our framework is model-agnostic, have good model explainability, and can flexibly control the explanation

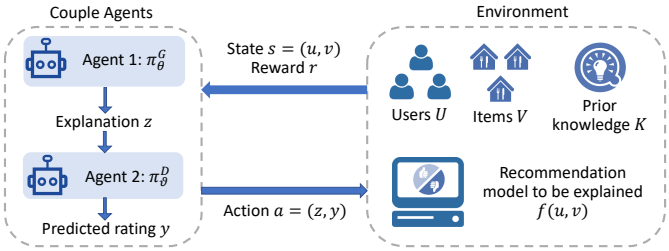

Figure 1: A reinforcement learning framework for generating recommendation explanations.

quality. Fig. 1 presents the overview of the framework. In our design, the recommendation model to be explained is a part of the environment. It is separated from the agents, which are responsible for 1) generating explanations and 2) predicting the output ratings of the recommendation model based on the explanations. The agents treat the recommendation model as a black box (model-agnostic) and interact with the environment to get information about how good the explanations are. The environment rewards the agents if they can correctly predict the output ratings of the recommendation model (model-explainability). Based on some prior knowledge about desirable explanations (e.g., desirable length of an explanation), the environment also rewards the agents if the explanations have good presentation quality (explanation quality control). The agents learn to generate explanations with good explainability and presentation quality by optimizing the expected reward of their actions.

The major contributions of this paper are as follows:

First, we **design a reinforcement learning framework for explainable recommendation**. The framework is model-agnostic, has good explainability, and can flexibly control the presentation quality of the explanations. We mathematically define the framework and its optimization goal, and show that it can be optimized by using doubly statistic gradient. We mathematically define the framework and its optimization goal. We further show that the defined goal can be optimized by using doubly stochastic policy gradient.

Second, we **instantiate the agents with personalized-attention-based neural networks**. These agents can be used to generate personalized sentence-level explanations. The key idea is to specify a distribution over all possible explanations and model the probability based on users' personalized attention on each sentence. Recurrent units are incorporated to the networks to model dependent selection of sentences and further improve explanation quality.

Finally, we **evaluate the effectiveness of our method by using both offline experiments and evaluation with human subjects**. We demonstrate that our method can effectively explain different recommender systems, including collaborative filtering methods such as NMF [16] and deep-learning-based models such as CDL [17]. Evaluation with human subjects show that the explanations generated by our method is considered significantly more useful than the baselines. We also conduct qualitative analysis to intuitively explain why our method performs the best.

The rest of the paper proceeds as follows. We first introduce the reinforcement learning framework and our model for generating personalized sentence-level explanation in Section II. Sections III and IV show the results of the offline experiments and evaluation results with human subjects, respectively. The related works are summarized in Section V. Section VI summarizes this work and concludes with a discussion on limitations and future research directions.

II. Reinforcement Learning for Explainable Recommendation

In this section, we first introduce the problem definition and our reinforcement learning framework for explainable recommendation. Next, we use sentence-level explanations as a guiding example to instantiate two crucial components in the framework: couple agents and reward function. Finally, we end by showing that the reward can be approximately optimized by using doubly stochastic policy gradient.

A. Problem Definition

We define our problem as follows.

Input: The input data of our model is the user set U, the item set V, and a recommendation model f to be explained. In this paper, we assume that f is targeted for explicit feedback, i.e., given a user $\boldsymbol{u} \in U$ and an item $\boldsymbol{v} \in V$, $f(\boldsymbol{u}, \boldsymbol{v})$ predicts a rating that captures how much $\boldsymbol{u}$ likes $\boldsymbol{v}$. It is also easy for our framework to consider implicit feedback (e.g., clicks). A user $\boldsymbol{u}$ can be represented by the user ID and/or some side information about the user (e.g., age and gender). Each item $\boldsymbol{v} \in V$ consists of an item ID i and a series of interpretable components: $\boldsymbol{v} = (i, \boldsymbol{l}_1, \boldsymbol{l}_2, ..., \boldsymbol{l}_m)$. Here $\boldsymbol{l}_j$ is an interpretable component that can be presented to a user for the purpose of explanation. If the items are restaurants, an interpretable component can be one sentence from a review of the restaurant or a key feature of the restaurant (e.g., five-star rating). An interpretable component can also be part of the textual description of an item. Taking the book *Harry Potter* as an example, an interpretable component for this book can be part of the book description on Wikipedia, e.g., "As of February 2018, the books have sold more than 500 million copies worldwide, making them the best-selling book series in history."

To generate explanations, it is important that the items can be represented by interpretable components. However, it is not necessary that f uses the interpretable components for training or testing. f can be a collaborative filtering method that only uses user IDs, item IDs, and rating scores as inputs.

Output: Given a user $\boldsymbol{u}$ and an item $\boldsymbol{v}$, our method extracts a subset of interpretable components from $\boldsymbol{v}$ as a personalized explanation for $\boldsymbol{u}$. Mathematically, the explanation $\boldsymbol{z}$ is defined as $(z_1, z_2, ...z_m)$, where $z_j = 1$ means that the jth

interpretable component is selected as part of the explanation and $z_j = 0$ means that the jth component is not selected. A good explanation should be easy to read (e.g., concise), consistent with the rating (consistency), and be sufficient for predicting $\boldsymbol{u}$'s preference on $\boldsymbol{v}$ (explainability).

B. Framework

Our framework for explainable recommendation is shown in Fig. 1. To describe the framework, we follow the common terminologies in reinforcement learning [18].

Environment: In our design, the environment consists of the user pool U, the item pool V, the recommender system to be explained f, and some prior knowledge K about the explanations. An example knowledge is the ideal length of the explanation (e.g., number of words within the explanation). The knowledge also includes the relative importance among different quality measures of the explanations (e.g., readability and consistency). Because of the advantages of reinforcement learning, the form of K can be flexibly designed based on the specific application scenario.

State: The state s is defined as the feature representation of the users and items: $\boldsymbol{s} = (\boldsymbol{u}, \boldsymbol{v})$.

Agent: We design couple agents to 1) generate the explanation and 2) ensure that the explanation is sufficient for predicting user preference (i.e., ensure explainability). The two agents are the generator $\pi^G_{\boldsymbol{\theta}}$ and the discriminator $\pi^D_{\boldsymbol{\vartheta}}$. Given a state $\boldsymbol{s} = (\boldsymbol{u}, \boldsymbol{v})$, $\pi^G_{\boldsymbol{\theta}}$ is responsible for generating the explanation $\boldsymbol{z}$, which explains to user $\boldsymbol{u}$ why v is or is not recommended. More specifically, $\pi^G_{\boldsymbol{\theta}}$ specifies a distribution over all possible explanations:

$$\pi^G_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{v}) = p(\boldsymbol{z}|\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}$ refers to the parameters of π^G. $\boldsymbol{z}$ can be generated by sampling from $p(\boldsymbol{z}|\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta})$. The other agent $\pi^D_{\boldsymbol{\vartheta}}$ takes $\boldsymbol{z}$ as the input to predict whether $\boldsymbol{u}$ will like $\boldsymbol{v}$:

$$y = \pi^D_{\boldsymbol{\vartheta}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}). \tag{2}$$

Here, y is the predicted rating and $\boldsymbol{\vartheta}$ is the parameters of π^D. This agent only leverages the information in the selected ($z_j = 1$) interpretable components for rating prediction. We assume the explainability of $\boldsymbol{z}$ is good if y is similar to $f(\boldsymbol{u}, \boldsymbol{v})$. This idea is borrowed from the *wrapper* methods for feature selection, which evaluate subsets of features based on their usefulness to a given predictor [19].

Action: For each state $\boldsymbol{s}$, the agents give action $\boldsymbol{a}$, which includes the explanation and predicting the predicted rating: $\boldsymbol{a} = (\boldsymbol{z}, y)$.

Reward: For state $\boldsymbol{s} = (\boldsymbol{u}, \boldsymbol{v})$ and action $\boldsymbol{a} = (\boldsymbol{z}, y)$, the environment computes the reward r based on the recommendation model to be explained f:

$$r = \mathcal{L}(f(\boldsymbol{u}, \boldsymbol{v}), y) + \Omega(\boldsymbol{z}). \tag{3}$$

Here $\mathcal{L}(f(\boldsymbol{u}, \boldsymbol{v}), y)$ is a measure of how similar y is to $f(\boldsymbol{u}, \boldsymbol{v})$. It represents the explainability of $\boldsymbol{z}$. $\Omega(\boldsymbol{z})$ measures the presentation quality (e.g., readability and consistency) of $\boldsymbol{z}$. The specific form of $\Omega(\boldsymbol{z})$ depends on the prior knowledge K. A detailed discussion of $\mathcal{L}$ and Ω is given in Sec. II-D.

Optimization goal: Following the REINFORCE algorithm [18], we optimize the expected reward:

$$\arg\max_{\boldsymbol{\theta}, \boldsymbol{\vartheta}} \sum_{\boldsymbol{u}, \boldsymbol{v}} E_{\boldsymbol{z} \sim p(\cdot|\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\theta})}[\mathcal{L}(f(\boldsymbol{u}, \boldsymbol{v}), \pi^D_{\boldsymbol{\vartheta}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z})) + \Omega(\boldsymbol{z})]. \tag{4}$$

For the rest of the paper, we instantiate the framework with a guiding example. In this example, we consider sentences as the interpretable components, i.e., each interpretable component $\boldsymbol{l}_j$ of $\boldsymbol{v}$ is a natural language sentence (e.g., a sentence from the review comments of $\boldsymbol{v}$). We then demonstrate how recommender systems can be explained using sentence-level explanations.

C. Couple Agents

Fig. 2 shows the neural network architecture of the couple agents. In this figure, the inputs (e.g., $\boldsymbol{l}_m$) or outputs (e.g., y) of an agent are marked with a circular border. The parameters of the agents (e.g., $\boldsymbol{\alpha}_{\boldsymbol{u}}$) or features of the neural networks (e.g., $\boldsymbol{\gamma}_m$) are marked with a rectangular border.

Agent π^G_θ: As described in Eq. (1), π^G_θ specifies a distribution over all possible explanations. Intuitively, the probability that $\boldsymbol{l}_j$ is selected should be positively correlated with the attention that $\boldsymbol{u}$ has on $\boldsymbol{l}_j$. Based on this idea, we design the neural network architecture as shown in Fig. 2 (left).

(1) Lookup. First, a lookup layer transforms $\boldsymbol{u}$ into a user latent vector $\boldsymbol{\alpha}_{\boldsymbol{u}} \in \mathbb{R}^{d_a}$ and transforms the item ID i into an item ID embedding $\boldsymbol{\beta}_i \in \mathbb{R}^{d_a}$. Here we assume that $\boldsymbol{u}$ is a one-hot representation of the user ID. If side information about the user is provided, we can easily incorporate the information by merging its embedding vector with $\boldsymbol{\alpha}_{\boldsymbol{u}}$.

(2) CNN. We then compute an embedding of each sentence by using a text processor. In this paper, we follow the state-of-the-art method, NARRE [13], to use the CNN text processor. The input of the CNN text processor is $\boldsymbol{l}_j \in \mathbb{R}^{d_s \times d_w}$, which is a word embedding matrix where the kth row contains the embedding vector of the kth word in the sentence. Here d_s is the maximum number of words in a sentence and d_w is the dimensionality of the word embedding vectors. In our implementation, we use word embeddings pre-trained on the Wikipedia corpus. If the number of words in the sentence is smaller than d_s, we padded $\boldsymbol{l}_j$ with zeros. Given each $\boldsymbol{l}_j$, the CNN text processor transforms it into a sentence embedding $\boldsymbol{\gamma}_j \in \mathbb{R}^{d_q}$. Suppose the CNN layer has d_q neurons, each with a filter $Q_t \in \mathbb{R}^{d_k \times d_w}$ and a bias $c_t \in \mathbb{R}$. The embedding $\boldsymbol{\gamma}_j$ of sentence $\boldsymbol{l}_j$ is computed by

$$\begin{aligned} \boldsymbol{o}_{jt} &= \sigma_R(\boldsymbol{l}_j * Q_t + c_t), \ \forall t \in [1, d_q], \\ \tilde{o}_{jt} &= \max(o_{jt1}, o_{jt2}, ..., o_{jt(d_s - d_k + 1)}), \\ \boldsymbol{\gamma}_j &= [\tilde{o}_{j1}, \tilde{o}_{j2}, ..., \tilde{o}_{jd_q}]. \end{aligned} \tag{5}$$

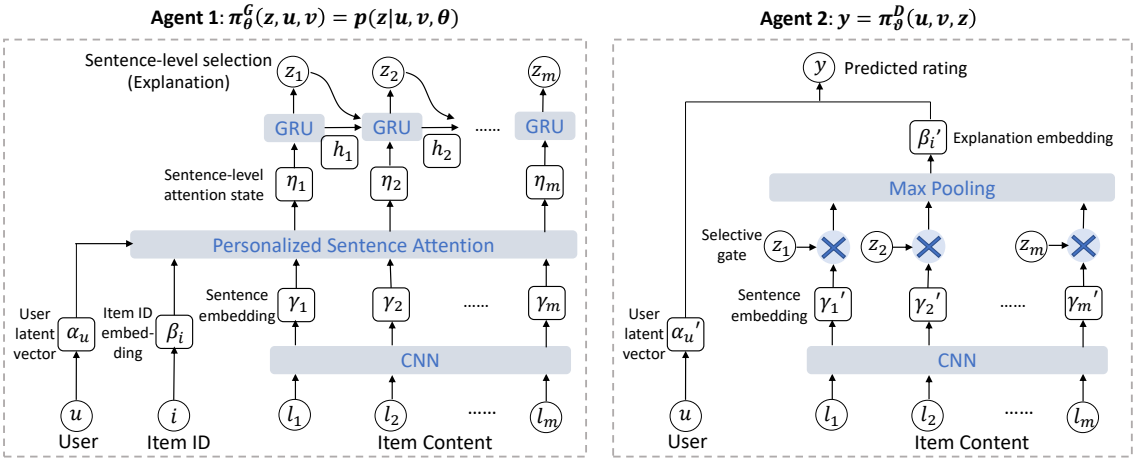

Figure 2: Couple agents for generating (π^G_θ) and discriminating (π^D_ϑ) the explanations. Nodes with a circular border represent the inputs or outputs of an agent. Nodes with a rectangular border are parameters or features of the neural networks.

Here $*$ denotes the convolution operator, $\sigma_R(\cdot)$ is ReLU, a non-linear activation function with $\sigma_R(\boldsymbol{x}) = \max(0, \boldsymbol{x})$, and $\boldsymbol{o}_{jt} = [o_{jt1}, o_{jt2}, ..., o_{jt(d_s-d_k+1)}] \in \mathbb{R}^{d_s-d_w+1}$.

(3) Attention-based selection. Finally, we calculate the probability of selecting $\boldsymbol{l}_j$ based on the attention $\boldsymbol{u}$ has on $\boldsymbol{l}_j$. We start with a simple case in which the probability of selecting $\boldsymbol{l}_j$ is considered conditionally independent from other selections given $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{\theta}$. In this case, we have

$$p(\boldsymbol{z}|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) = \textstyle\prod_{j=1}^{m} p(z_j|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}),$$
$$p(z_j|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) = \sigma_S(\boldsymbol{\eta}_j W_z + b_z), \quad (6)$$

where $\sigma_S(x) = e^x/(e^x+1)$ is the sigmoid function, $W_z \in \mathbb{R}^{d_a \times 1}$ and $b_z \in \mathbb{R}$ are trainable parameters, and $\boldsymbol{\eta}_j$ is the personalized senetence-level attention state computed by jointly considering $\boldsymbol{u}$, i, and $\boldsymbol{l}_j$:

$$\boldsymbol{\eta}_j = \sigma_R(\boldsymbol{\alpha_u} + \boldsymbol{\beta}_i + \boldsymbol{\gamma}_j W_a + \boldsymbol{b}_a). \quad (7)$$

Here $W_a \in \mathbb{R}^{d_q \times d_a}$ and $\boldsymbol{b}_a \in \mathbb{R}^{d_a}$ are trainable parameters. Although independent selection of sentences is sufficient in some applications, it is not able to ensure that the explanation is coherent (e.g., selecting adjacent sentences to provide more context) and diversified. To solve the problem, we introduce a hidden state $\boldsymbol{h}_j \in \mathbb{R}^{d_h}$ to model the dependent selection of sentences. Specifically, we replace Eq. (6) with:

$$p(\boldsymbol{z}|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) = \textstyle\prod_{j=1}^{m} p(z_j|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta},\boldsymbol{z}_{1:(j-1)}),$$
$$p(z_j|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta},\boldsymbol{z}_{1:(j-1)}) = \sigma_S(\boldsymbol{\eta}_j W_z + \boldsymbol{h}_{j-1} W_h + b_z),$$
$$\boldsymbol{h}_j = g(\boldsymbol{h}_{j-1}, [\boldsymbol{\eta}_j; z_{j-1}]). \quad (8)$$

Here $\boldsymbol{z}_{1:(j-1)} = [z_1, z_2, ...z_{j-1}] \in \mathbb{R}^{j-1}$, $W_h \in \mathbb{R}^{d_h \times 1}$ is a trainable parameter, and $g(\cdot)$ is a recurrent unit, which can be the vanilla RNN, LSTM, or GRU. Here we use GRU in our implementation as it is computationally efficient and has comparable performance with LSTM [20].

Agent $\boldsymbol{\pi}^D_\vartheta$: Given a user $\boldsymbol{u}$, an item $\boldsymbol{v}$, and the explanation $\boldsymbol{z}$, $\boldsymbol{\pi}^D_\vartheta$ predicts how much the user likes the item by only considering $\boldsymbol{u}$ and the information in the selected interpretable components of $\boldsymbol{v}$. We say that the explainability of $\boldsymbol{z}$ is good if it is sufficient for rating prediction. As shown in the right part of Fig. 2, $\boldsymbol{z}$ is used as a gate in the agent and masks information that is not selected in the explanation. Specifically, y is calculated by:

$$y = \boldsymbol{\alpha}'_u \boldsymbol{\beta}'^T_i, \quad (9)$$

where $\boldsymbol{\alpha}'_u \in \mathbb{R}^{d_p}$ is the user latent vector, $\boldsymbol{\beta}'_i = [\beta'_{i1}, \beta'_{i2}, ..., \beta'_{id_p}] \in \mathbb{R}^{d_p}$ is the explanation embedding computed by

$$\beta'_{it} = max(\tilde{\gamma}_{1t}, \tilde{\gamma}_{2t}, ..., \tilde{\gamma}_{mt}), \forall t \in [1, d_p],$$
$$\tilde{\boldsymbol{\gamma}}_j = z_j \boldsymbol{\gamma}'_j, \forall j \in [1, m]. \quad (10)$$

Here $\tilde{\boldsymbol{\gamma}}_j \in \mathbb{R}^{d_q}$, $\tilde{\gamma}_{jt}$ is the tth element of $\tilde{\boldsymbol{\gamma}}_j$, $\boldsymbol{\gamma}'_j \in \mathbb{R}^{d_q}$ is a sentence embedding calculated by using equations similar to that of Eq. (5). The only difference from Eq. (5) is that we use different filters Q'_t and a different bias c'_t in $\boldsymbol{\pi}^D_\vartheta$.

D. Reward

We now introduce our design of the reward r. As described in Eq. (3), the reward consists of two parts. The first part is the explainability $\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}), y)$. The second part is the presentation quality $\Omega(\boldsymbol{z})$, which is a weighted combination of readability $\Omega_r(\boldsymbol{z})$ and consistency $\Omega_c(\boldsymbol{z})$: $\Omega(\boldsymbol{z}) = \lambda_r \Omega_r(\boldsymbol{z}) + \lambda_c \Omega_c(\boldsymbol{z})$. Here $\lambda_r > 0$ and $\lambda_c > 0$ balance the importance of different parts of the reward.

Explainability. As discussed before, we assume the explainability of $\boldsymbol{z}$ is good if it is sufficient for rating prediction. We borrow this idea from the *wrapper* methods for feature selection [19], which evaluate subsets of features based on their usefulness to a given predictor. To measure the quality for rating prediction, we follow NARRE [13] and leverage the commonly used objective function squared loss:

$$\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}), y) = -(y - f(\boldsymbol{u},\boldsymbol{v}))^2. \quad (11)$$

Here $\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}),y)$ is the negative squared loss. Larger $\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}),y)$ indicates better explainability.

Readability. Studies on explainable recommendation suggest that a user will be overwhelmed if s/he is presented with too much information within an explanation [21], [22]. Accordingly, we assume that an explanation is difficult to read if it contains too much information. Suppose z^* is the ideal number of sentences in the explanation, the readability of $\boldsymbol{z}$ is defined as:

$$\Omega_r(\boldsymbol{z}) = -|z^* - \sum_{j=1}^{m} z_j|. \tag{12}$$

Larger $\Omega_r(\boldsymbol{z})$ indicates better readability. If z^*=0, $\Omega_r(\boldsymbol{z})$ becomes a commonly used regularization term. If $z^*>0$, we also penalize the explanation for containing no information.

The readability can be flexibly designed based on the application. For example, sometimes we may hope that the selected sentences are adjacent to each other so that the explanation is more coherent and the users can better understand the overall context. Then, we can design $\Omega_r(\boldsymbol{z})$ as:

$$\Omega_r(\boldsymbol{z}) = -|z^* - \sum_{j=1}^{m} z_j| - \lambda_b \sum_{j=2}^{m} |z_j - z_{j-1}|, \tag{13}$$

where $\lambda_b > 0$ is used to balance the two terms on the right.

Consistency. If the interpretable components are sentences from the review comments, the generated explanations usually express positive or negative sentiment. In such a case, it is desirable that the sentiment of the explanation is consistent with the rating, i.e., higher (lower) ratings correspond to more positive (negative) sentiments. To this end, we design a consistency measure based on the Pearson correlation [23]:

$$\Omega_c(\boldsymbol{z}) = \left(\frac{\sum_{j=1}^{m} z_j \varphi(\boldsymbol{l}_j)}{\sum_{j=1}^{m} z_j} - \bar{\varphi}\right)(f(\boldsymbol{u},\boldsymbol{v}) - \bar{f}), \tag{14}$$

where $\varphi(\boldsymbol{l}_j)$ returns the sentiment score (in the range of [-1, 1]) of $\boldsymbol{l}_j$, $\bar{\varphi} \in \mathbb{R}$ is the average sentiment of all interpretable components in the training data, and $\bar{f} \in \mathbb{R}$ is the average rating of all the training samples. In our implementation, we calculate the sentiments by using a word-embedding-based sentiment classification method [24].

E. Doubly Stochastic Policy Gradient

Now we demonstrate how the expected reward defined in Eq. (4) can be optimized. Following the idea of the REINFORCE algorithm [18], we derive a sampled approximation to the gradient of the expected reward. For simplicity, we denote $\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}),\pi^D_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z})) + \Omega(\boldsymbol{z})$ as $\psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z})$. Consider a training pair $(\boldsymbol{u},\boldsymbol{v})$, by using policy gradient, we have:

$$\begin{aligned}
& \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{z}\sim p(\cdot|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})} \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}) \\
\approx\ & \nabla_{\boldsymbol{\theta}} \textstyle\sum_{\boldsymbol{z}'} p(\boldsymbol{z}'|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}') \\
=\ & \textstyle\sum_{\boldsymbol{z}'} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{z}'|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}') \\
=\ & \textstyle\sum_{\boldsymbol{z}'} p(\boldsymbol{z}'|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}'|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}') \\
\approx\ & E_{\boldsymbol{z}\sim p(\cdot|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta}) \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}).
\end{aligned} \tag{15}$$

The above equation tells us that $\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{z}\sim p(\cdot|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})} \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z})$ is an expected summation over $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})$ weighted by the reward $\psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z})$. Intuitively speaking, suppose we have multiple samples of $\boldsymbol{z}$, the model will update $\boldsymbol{\theta}$ so that sample $\boldsymbol{z}'$ with larger reward $\psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}')$ will have larger probability to be sampled in the future (increased $\log p(\boldsymbol{z}'|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})$).

Similarly, we can update $\boldsymbol{\vartheta}$ by:

$$\begin{aligned}
& \nabla_{\boldsymbol{\vartheta}} E_{\boldsymbol{z}\sim p(\cdot|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})} \psi_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z}) \\
=\ & E_{\boldsymbol{z}\sim p(\cdot|\boldsymbol{u},\boldsymbol{v},\boldsymbol{\theta})} \nabla_{\boldsymbol{\vartheta}} \mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}), \pi^D_{\boldsymbol{\vartheta}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{z})).
\end{aligned} \tag{16}$$

Based on Eq. (15) and Eq. (16), we can now optimize Eq. (4) by using doubly stochastic gradient descend [25]. To alleviate the issue of overfitting, we follow NARRE [13] and use dropout [26] in the neural networks of the agents.

III. Offline Experiments

In this section, we conduct two experiments to evaluate our method in an offline setting. In the first experiment, we demonstrate that our method outperforms the baselines in explaining different recommender systems. In the second experiment, we show the influence of the parameters of our proposed model. The results show that our method performs the best at varied explanation lengths.

A. Experimental settings

Datasets. We use two public datasets from different domains to evaluate our model. The first dataset, **Amazon_Toys_and_Games**, is from Amazon 5-core[1] This dataset contains more than 150,000 reviews and ratings of toys and games sold on Amazon. The second dataset, **Yelp_2018_LasVegas**, is from Yelp Challenge 2018[2]. Since the raw data is large and sparse, we preprocess the dataset, keeping only English reviews written for restaurants located jiein Las Vegas, and ensuring that users and items have at least 10 reviews. Even after the preprocessing, the dataset is still large, with over 500,000 reviews from more than 20,000 users. Table I summarizes the statistics of the two datasets.

The ratings of both datasets range from 1 to 5. For each item (a restaurant, a toy, or a game), the interpretable components are considered as sentences that come from the review comments of this item. We remove a sentence if it is longer than 90% of the sentences. Then, we set m (the

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

[2]https://www.yelp.com/dataset/challenge

Table I. Statistics of the datasets.

	Amazon_Toys_and_Games	Yelp_2018_LasVegas
#users	19,412	23,196
#items	11,924	13,433
#reviews and ratings	167,597	568,454

number of interpretable components of an item) so that 90% of the items can keep all interpretable components. If $m_{\boldsymbol{v}}$, which is the number of interpretable components of item $\boldsymbol{v}$, is smaller than m, we set $\boldsymbol{l}_j$ of $\boldsymbol{v}$ to $\mathbf{0}$ for all $j > m_{\boldsymbol{v}}$. We randomly split each dataset into a training set (80%), a validation set (10%), and a test set (10%).

Baselines. We compare our method with two baselines. Suppose we want to include z^* sentences in the explanation. The first baseline, **Random**, randomly selects z^* sentences from all candidates. The second baseline, **NARRE**, is the state-of-art explainable recommendation method [13]. Given a set of reviews, NARRE learns the usefulness of each review by using a Neural Attentional Regression model. By considering sentences as reviews in their model, we get a usefulness score for each sentence. The top z^* sentences with the highest usefulness score are included in the explanation. Note that the usefulness scores given by NARRE are not personalized, i.e., NARRE generates the same explanations for different users.

Recently, researchers have proposed methods to generate feature/phrase-level explanations [8], [9]. However, features/phrases reduce the integrity of the reviews [27], and the performance of these methods often rely on the manual preprocessing needed in phrase-level sentiment analysis [13]. Moreover, our method is not conflicted with such explanations, since the framework can also be adopted to generate feature/phrase-level explanations. Thus, we did not compare with feature/phrase-level explanations in this paper.

Evaluation criteria. To evaluate the performance of our method, we design two criteria. The first criterion, M_c, measures whether the explanations are consistent with the ratings. It is defined as the Pearson correlation between ratings and the explanation sentiments, which can be regarded as a normalized version of Eq. (14). Specifically,

$$M_c = \frac{\sum\limits_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{T}} (\phi(\boldsymbol{u},\boldsymbol{v}) - \bar{\phi})(f(\boldsymbol{u},\boldsymbol{v}) - \bar{f})}{\sqrt{\sum\limits_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{T}} (\phi(\boldsymbol{u},\boldsymbol{v}) - \bar{\phi})^2} \sqrt{\sum\limits_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{T}} (f(\boldsymbol{u},\boldsymbol{v}) - \bar{f})^2}}. \quad (17)$$

Here $\mathcal{T}$ refers to the test set, $\phi(\boldsymbol{u},\boldsymbol{v}) = \frac{\sum_{j=1}^{m} z_j \varphi(\boldsymbol{l}_j)}{\sum_{j=1}^{m} z_j}$ is the average sentiment of sentences in the explanation, and $\bar{\phi} = \sum\limits_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{T}} \phi(\boldsymbol{u},\boldsymbol{v})$ is the average sentiment of all explanations. A larger M_c indicates better consistency. The second criterion, M_e, measures the explainability by summing up $\mathcal{L}(f(\boldsymbol{u},\boldsymbol{v}), y)$ defined in Eq. (11):

$$M_e = -\sum_{(\boldsymbol{u},\boldsymbol{v})\in\mathcal{T}} (y - f(\boldsymbol{u},\boldsymbol{v}))^2. \quad (18)$$

To calculate M_e for the baselines, we replace π^G in our framework with the explanation selection methods of the baselines and update π^D by using Eq. (16). Then we evaluate M_e based on the output y of π^D.

We do not measure **readability** in an offline setting because the baselines cannot dynamically determine the number of sentences in an explanation. Thus, they are designed so that Ω_r in Eq. (12) is always 0. For fair comparison, we ensure that our Ω_r is also 0 by randomly adding sentences to and removing sentences from the explanations whenever necessary.

Parameter settings. The parameters of NARRE are carefully tuned. We try different learning rates ([0.01, 0.02, 0.05]), different batch sizes ([10, 20, 50, 100]), and different numbers of latent factors ([16, 32, 64]). The parameters with the best performance in the validation set are used for testing. For CNN text processors used in NARRE and our method, we reuse the settings reported in the paper of NARRE. Specifically, we set the number of neurons in the CNN text processor (d_q) to 100, the dropout ratio to 0.5, and the window size of the CNN filter (d_k) to 3. For the parameters used in the attention-based selection, we settle for $d_h = 16$ and $d_a = 32$. To balance explainability, readability, and consistency, we set λ_r to 1 and λ_c to 0.5.

B. Explaining different recommender systems

In this experiment, we train four different recommendation models and evaluate how well our method can explain these models (f). The four models are:

- **NMF** [16]. NMF uses Non-negative Matrix Factorization for rating prediction. It only leverages user IDs, item IDs and rating scores as the input.
- **PMF** [28]. The Probabilistic Matrix Factorization model leverages the Gaussian distribution to model latent user features and latent item features.
- **SVD++** [29]. This model merges Singular Value Decomposition with neighborhood models. Compared with other collaborative filtering methods, this model incorporates an additional set of factors that model item-item relations.
- **CDL** [17]. The collaborative deep learning model is a state-of-the-art deep model for recommendation. It jointly performs collaborative filtering and deep representation learning. A Bayesian stacked denoising autoencoder (SDAE) is introduced for encoding auxiliary information.

We have simple collaborative filtering methods that leverages only user IDs, item IDs and rating scores for prediction (e.g., NMF). The above models are selected to cover different types and different complexities. We have

Table II. Explaining different recommendation models trained on the **Amazon_Toys_and_Games** dataset. Here NMF, PMF, SVD++, and CDL are recommendation models to be explained. Larger M_c and M_e indicate better consistency and explainability, respectively.

	M_c					M_e				
	NMF	PMF	SVD++	CDL	GT	NMF	PMF	SVD++	CDL	GT
Random	0.006	0.007	0.035	0.010	0.030	-1.329	-1.046	-0.150	-1.080	-0.981
NARRE	0.012	0.022	0.038	0.043	0.048	-1.271	-1.032	-0.142	-0.967	-0.927
Ours	**0.025**	**0.028**	**0.048**	**0.079**	**0.155**	**-1.234**	**-0.956**	**-0.130**	**-0.956**	**-0.903**

Table III. Explaining different recommendation models trained on the **Yelp_2018_LasVegas** dataset. Here NMF, PMF, SVD++, CDL, and GT are recommendation models to be explained. Larger M_c and M_e indicate better consistency and explainability, respectively.

	M_c					M_e				
	NMF	PMF	SVD++	CDL	GT	NMF	PMF	SVD++	CDL	GT
Random	-0.030	-0.030	-0.031	0.012	0.007	-0.478	-0.287	-0.266	-0.517	-1.488
NARRE	-0.015	-0.000	0.018	0.031	0.038	-0.448	-0.266	-0.239	-0.482	-1.424
Ours	**0.018**	**0.037**	**0.041**	**0.227**	**0.168**	**-0.421**	**-0.258**	**-0.232**	**-0.460**	**-1.380**

simple collaborative filtering methods that leverage only user IDs, item IDs and rating scores for prediction (e.g., NMF). We also have complex, hybrid methods that take advantage of deep neural networks (CDL). Except for these four models, we also evaluate whether our method can well explain the ground truth rating scores (**GT**).

Tables II and III show the experiment results for the **Amazon_Toys_and_Games** dataset and the **Yelp_2018_LasVegas** dataset, respectively. We make a few observations based on the results. First, NARRE consistently outperforms Random. This demonstrates the effectiveness of NARRE's attention-based usefulness calculation method. Second, our method consistently performs better than NARRE in terms of both M_c and M_e. This is because the explanations generated by NARRE are not personalized. Thus, the explainability (M_e) of the model is limited. Also, NARRE cannot optimize sentiment consistency (M_c). Because we use reinforcement learning, we can flexibly incorporate sentiment consistency as part of the reward and optimize it during training. By comparing the M_e values for different f, we observe that some recommendation models are easy to explain (e.g., SVD++) and some recommendation models are more difficult to explain (e.g., CDL). Whether a model can be easily explained may depend on the dataset used and the training parameters.

C. Effect of varied explanation lengths

In this experiment, we examine how the explanation length $z*$ impacts M_c and M_e. Here f is set to **GT**. The results are shown in Tables IV and V. Based on the analysis of the results, we make the following conclusions.

First, for any $z^* \in [1, 5]$, our method consistently outperforms NARRE, and NARRE consistently performs better than Random. This demonstrates again that our personalized explanations have better explainability (M_e) compared with NARRE's non-personalized explanations. Also, because of the advantages of reinforcement learning, we can flexibly design the reward function. Thus, we are able to explicitly optimize sentiment consistency during training and perform better than NARRE in terms of M_c.

Second, both M_c and M_e tend to increase with increasing z^*. This pattern is clearer for our method and NARRE than for Random. It shows that by using our method or NARRE, the amount of useful information contained in the explanations increases with the explanation length (increasing m_e). Also, the useful information is usually consistent with the rating scores (increasing m_c).

IV. Evaluation with Human Subjects

In this section, we evaluate whether the generated explanations can help real-world users make better decisions. To this end, we perform both quantitative and qualitative analysis. The quantitative results show that our method outperforms the baselines and the qualitative analysis helps us better understand why our method is better.

A. Quantitative analysis

To evaluate the usefulness of the explanations, we recruit four Yelp users who have written at least 20 Yelp reviews and ask them to label the explanations. Before the experiment, we collect their reviews and the reviews about the restaurants they mention. These newly collected reviews are merged with other reviews in the Yelp dataset. We randomly sample 100 restaurants for testing and make sure that none of the participants go to any of the 100 restaurants. For each restaurant, we generate explanations by using our method, NARRE, and Random. The three generated explanations are then presented to the participants in random order. Next, we ask the participants to choose the explanations that are most useful in helping them decide whether they will go to the restaurants. If they find that two (or three) explanations for a restaurant are equally useful, we ask them to choose both

Table IV. Comparison of M_c and M_e at different explanation lengths (the **Amazon_Toys_and_Games** dataset).

	M_c					M_e				
	$z^*=1$	$z^*=2$	$z^*=3$	$z^*=4$	$z^*=5$	$z^*=1$	$z^*=2$	$z^*=3$	$z^*=4$	$z^*=5$
Random	0.030	0.013	0.029	0.037	0.037	-0.981	-0.991	-0.973	-0.962	-0.995
NARRE	0.048	0.064	0.089	0.110	0.133	-0.927	-0.919	-0.910	-0.911	-0.906
Ours	**0.155**	**0.142**	**0.140**	**0.160**	**0.161**	**-0.903**	**-0.901**	**-0.898**	**-0.898**	**-0.894**

Table V. Comparison of M_c and M_e at different explanation lengths (the **Yelp_2018_LasVegas** dataset).

	M_c					M_e				
	$z^*=1$	$z^*=2$	$z^*=3$	$z^*=4$	$z^*=5$	$z^*=1$	$z^*=2$	$z^*=3$	$z^*=4$	$z^*=5$
Random	0.007	0.011	0.012	0.032	0.030	-1.488	-1.405	-1.403	-1.400	-1.406
NARRE	0.038	0.035	0.044	0.057	0.054	-1.424	-1.390	-1.377	-1.378	-1.372
Ours	**0.168**	**0.172**	**0.183**	**0.188**	**0.160**	**-1.380**	**-1.377**	**-1.370**	**-1.366**	**-1.353**

(or all) explanations. They can ignore a restaurant if none of the explanations are useful. On average, a participant ignores about 5.5% of the restaurants.

Table 3 shows how frequently one method is considered the most useful. We normalized the results so that the frequencies of all methods add up to 1. The labeling results of each participant (P1 to P4) and the overall summarization are displayed. The error bars in the figure denote the standard deviation values. We can see that all participants consider our method the most useful. Also, compared with Random, all participants consider NARRE more or equally useful. The overall summarization also shows that our method (42%±2%) is statistically better than NARRE (34%±5%), and NARRE is statistically better than Random (24%±4%). Note that the standard deviation of NARRE (5%) is much larger than that of our method (2%). This suggests that our method is robust and works well for different users.

B. Qualitative analysis

To better understand why our method is better than NARRE, we more carefully analyze two participants, P3 and P4. As shown in Fig. 3, among the four participants, P3 likes explanations generated by NARRE the most (40%)

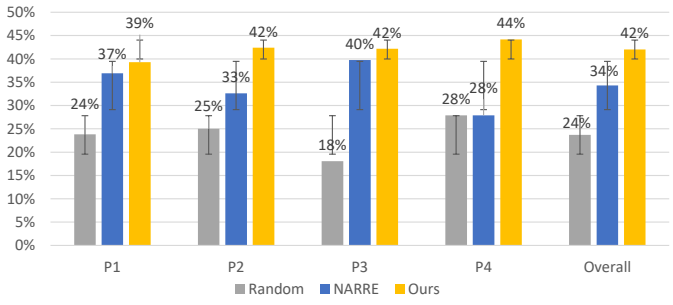

Figure 3: Frequency of a method (Random, NARRE, or ours) being considered the most useful. We show the results of individual participants (P1 to P4) as well as the overall summarization.

and P4 likes NARRE the least (28%). After analyzing their review comments in Yelp, we find that P3 likes **food**, especially meat, while P4 pays special attention to the **service** and environment of the restaurants. The most frequently mentioned keywords in the review comments of P3 are "**chicken**, **buffet**, portions, **sushi**, **beef**" while the most frequently mentioned keywords in the review comments of P4 are "**service, pizza**, **server**, **table**, **clean**." After analyzing the generated explanations, we find that many explanations generated by NARRE contain information about food and few of them are about service and environment. As a result, P3 accepts explanations generated by NARRE better than P4. Because the explanations generated by NARRE are non-personalized, they work best for people with common tastes. Different from NARRE, our method generates personalized explanations and works well for both P3 and P4. In Table VI, we use three restaurants (items) as examples and display the generated explanations. As shown in the table, we are able to generate **food** related explanations for P3 and **service** related explanations for P4.

V. Related Work

We divide existing explainable recommendation methods into two groups: post-hoc and embedded.

In post-hoc methods [3], [5], [6], [30], the explanations are generated after the items are recommended. Typical post-hoc explanations are item-based (e.g., "customers who bought this item also bought..." [5]), tag-based (e.g., "you have enjoyed other movies tagged with quirky" [30]), or social-based (e.g., "Amit Sharma and 5 of your friends like this" [6]). These methods usually select explanations from a set of pre-defined candidates without considering the working mechanism of the recommendation model. While the explanations are usually persuasive and readable, the diversity of the explanations is usually limited by the number of manually-defined explanation candidates. Moreover, these models do not consider model explainability. Compared with

Table VI. Example explanations generated by NARRE and our methods. Our method accurately captures the interests of P3 and P4. It generates **food** related explanations for P3 and **service** related explanations for P4. NARRE only focus on one aspect since it is not personalized.

	NARRE	Ours - P3	Ours - P4
Item 1	By the way, try to park at the side of gold coast farthest from the rio if you want to have a shorter walk, which is healthier than it sounds due to less secondhand smoke exposure.	The **chicken**'s feet was tasty, so were the **har gow**.	In the past we had **trouble communicating with the staff** because they usually speak in their own language, this last time though it seems they have hired more **English speaking staff** and it was **considerably easier to order**.
Item 2	If you needa **fajita**, your search should end here.	They came with red & green **peppers** and **onions**. First, I thought the **salsa** was delicious, and i appreciated it was actually spicy versus the mild you typically receive.	Overall, the **service** throughout our meal was swift & friendly.
Item 3	Unfortunately, after living in the city for a few years and trying a lot of wonderful **food** that this city has to offer, we returned for a visit and I was less than impressed.	It was the perfect **burger**, **cheesy** with just the right amount of dressing and **chips**!	At least **put the stuff in a fancy container**?

post-hoc methods, our method has better model explainability and can generate more diversified results. Given a list of pre-defined candidates, our framework can also be used to automatically rank these candidates.

Embedded methods [7]–[10], [12], [13] integrate the explanation generation process into the construction of a recommendation model. Pioneer embedded methods [7]–[9], [11] design explainable recommendation methods based on collaborative filtering. For example, Zhang et al. [8] improve the explainability of collaborative filtering by adding an additional set of latent vectors learned from explicit factors. The explicit factors are aspects of a product extracted from reviews. Diao et al. [9] proposed a probabilistic model based on topic modeling and collaborative filtering. The model can jointly predict user preferences and learn the important aspects in the review as well as the sentiments of these aspects. Except for collaborative filtering, researchers have also proposed explanation strategies for deep-learning-based methods [12]–[14]. Most of these works leverage the attention mechanisms to 1) learn a better embedding for the interpretable components and 2) automatically assign a weight to each interpretable component. Instead of refining existing recommendation models, researchers have also built inherently more explainable models by using intuitive data structures such as graphs [10], decision trees [31], and knowledge bases [32], [33]. Recently, researchers also proposed methods that automatically generate natural language sentences word-by-word [20], [34].

The embedded methods usually have good model explainability and sometimes help improve model accuracy. However, different explanation strategies need to be designed for different types of recommendation models. Moreover, it is difficult to control the presentation quality (e.g., consistency) of the generated explanations. Compared with embedded methods, our framework is model-agnostic and can flexibly control the explanation quality.

VI. Conclusions and Future Work

In this paper, we present a reinforcement learning framework for explainable recommendation. Our framework is model-agnostic, has good explainability, and can flexibly control the presentation quality of the explanations based on the application scenario. To show the effectiveness of the framework, we use sentence-level explanations as a guiding example and show how the agents and reward functions can be designed. Designed as a personalized-attention-based neural network, our generator agent models users' personalized preference for each sentence. It also models dependent selection of the sentences for better quality control. Both offline experiments and evaluation with human subjects demonstrate the effectiveness of our framework.

Limitations. The evaluation results demonstrate the effectiveness of our model. However, our work does have several limitations. First, we optimize the model by using doubly stochastic policy gradient. Similar to the REINFORCE algorithm, it suffers from high variance and slow convergence [35]. We can alleviate the issue by using advanced sampling methods (e.g., Gumbel-Softmax [35]) or actor-critic reinforcement learning methods such as Deep Deterministic Policy Gradient [36]. Second, in this paper, the usefulness of the explanations is evaluated through an informal user study with only four human subjects. To better understand how the explanations impact the general public, we will present a user study conducted with more participants in an extended version of the paper.

Future work. An interesting future work is to use our method to debug failed recommender systems. Since the framework has good explainability, it may help us better understand the recommender systems and find the reason why they work or do not work. We are also interested in developing more offline evaluation measures for explainable recommendation and evaluate their correlations with measures that can only be evaluated with human subjects. Another interesting direction for future work is to generate other

types of explanations by using our framework. For example, we can generate visual explanations (e.g., images) and also word-level explanations (e.g., fragments of sentences). Moreover, we plan to evaluate how our model performs when used to explain advanced CNN-based or RNN-based recommendation models (e.g., DeepCoNN [37]). Providing a library that facilitates implementation of models in our framework is also on the schedule.

REFERENCES

[1] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl, "Is seeing believing?: How recommender system interfaces affect users' opinions," in *SIGCHI*, 2003, pp. 585–592.

[2] N. Tintarev and J. Masthoff, "Effective explanations of recommendations: User-centered design," in *ACM Conference on Recommender Systems*, 2007, pp. 153–156.

[3] M. Bilgic and R. J. Mooney, "Explaining recommendations: Satisfaction vs. promotion," in *Beyond Personalization Workshop, IUI*, vol. 5, 2005, p. 153.

[4] G. Carenini and J. D. Moore, "An empirical study of the influence of user tailoring on evaluative argument effectiveness," in *IJCAI*, 2001, pp. 1307–1312.

[5] N. Tintarev and J. Masthoff, *Designing and evaluating explanations for recommender systems*, 2011, pp. 479–510.

[6] A. Sharma and D. Cosley, "Do social explanations work?: Studying and modeling the effects of social explanations in recommender systems," in *WWW*, 2013, pp. 1133–1144.

[7] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *ACM Conference on Recommender Systems*, 2013, pp. 165–172.

[8] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *SIGIR*, 2014, pp. 83–92.

[9] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars)," in *SIGKDD*, 2014, pp. 193–202.

[10] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *CIKM*, 2015, pp. 1661–1670.

[11] Y. Wu and M. Ester, "Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering," in *WSDM*, 2015, pp. 199–208.

[12] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *SIGIR*, 2017, pp. 335–344.

[13] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *WWW*, 2018, pp. 1583–1592.

[14] X. Chen, Y. Zhang, H. Xu, Y. Cao, Z. Qin, and H. Zha, "Visually explainable recommendation," *arXiv preprint arXiv:1801.10288*, 2018.

[15] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *arXiv preprint arXiv:1804.11192*, 2018.

[16] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*, 2001, pp. 556–562.

[17] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *SIGKDD*, 2015, pp. 1235–1244.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*, 1992, vol. 173, pp. 5–32.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[20] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," in *SIGIR*, 2017, pp. 345–354.

[21] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the ACM conference on Computer supported cooperative work*, 2000, pp. 241–250.

[22] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*, 2007, pp. 291–324.

[23] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, 2009, pp. 1–4.

[24] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2014, pp. 1555–1565.

[25] M. B. Christopher, *PATTERN RECOGNITION AND MACHINE LEARNING.*, 2016.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *SIGKDD*, 2016, pp. 1135–1144.

[28] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS*, 2008, pp. 1257–1264.

[29] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*, 2008, pp. 426–434.

[30] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: Explaining recommendations using tags," in *IUI*, 2009, pp. 47–56.

[31] X. Wang, X. He, F. Feng, L. Nie, and T.-S. Chua, "Tem: Tree-enhanced embedding model for explainable recommendation," in *WWW*, 2018, pp. 1543–1552.

[32] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripple network: Propagating user preferences on the knowledge graph for recommender systems," *arXiv preprint arXiv:1803.03467*, 2018.

[33] Y. Zhang, Q. Ai, X. Chen, and P. Wang, "Learning over knowledge-base embeddings for recommendation," *arXiv preprint arXiv:1803.06540*, 2018.

[34] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations," in *IUI Companion*, 2018, p. 57.

[35] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[36] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[37] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *WSDM*, 2017, pp. 425–434.