Incorporating Review-missing Interactions for Generative Explainable Recommendation

Xi Li, Xiaohe Bo, Chen Ma, Xu Chen*
Gaoling School of Artificial Intelligence, Renmin University of China
{lixi2021, xiaohe, machen2001, xu.chen}@ruc.edu.cn

Abstract

Explainable recommendation has attracted much attention from the academic and industry communities. Traditional models usually leverage user reviews as ground truths for model training, and the interactions without reviews are totally ignored. However, in practice, a large amount of users may not leave reviews after purchasing items. In this paper, we argue that the interactions without reviews may also contain comprehensive user preferences, and incorporating them to build explainable recommender model may further improve the explanation quality. To follow such intuition, we first leverage generative models to predict the missing reviews, and then train the recommender model based on all the predicted and original reviews. In specific, since the reviews are discrete tokens, we regard the review generation process as a reinforcement learning problem, where each token is an action at one step. We hope that the generated reviews are indistinguishable with the real ones. Thus, we introduce an discriminator as a reward model to evaluate the quality of the generated reviews. At last, to smooth the review generation process, we introduce a self-paced learning strategy to first generate shorter reviews and then predict the longer ones. We conduct extensive experiments on three publicly available datasets to demonstrate the effectiveness of our model.

1 Introduction

Explainable recommendation has recently attracted much attention from the academic and industry communities. It basically aims to solve the problem of "why an item should be recommended to a user". In early years, explainable recommender models are mostly based on features (Zhang et al., 2014; Seo et al., 2017), where the explanations are generated by predicting user favorite features, and filling them into fixed templates. However, the explanation capability of such methods can be limited by the feature set, which is usually small in practice, and the fixed template may be less friendly and vivid as an interface to communicate with users. To alleviate these problems, people propose to directly generate explanations in the form of natural languages (called generative explainable recommendation) (Li et al., 2020, 2023, 2017a), which are much more flexible and user accessible. Usually, generative explainable recommendation regards user reviews as explanation ground truths, and converts the explanation generation task to the review prediction problem. For example, NRT (Li et al., 2017b) regards user reviews as word sequences, and predicts them based on gated recurrent neural networks. PETER (Li et al., 2021) incorporates IDs of users and items into the transformer to generate personalized explanations. PEPLER (Li et al., 2023) regards user and item IDs as prompts and leverages GPT-2 to enhance the explanation generation process.

While the above generative explainable recommender models have achieved remarkable successes, they all require that all the user-item interactions should have reviews, and for the review-missing interactions, they usually ignore them in the model optimization process. We believe such requirement may severely limit the performance of existing models in practice to provide reasonable explanations. To be more specific, in real-world scenarios, a large amount of users may only leave a small number of reviews, even though they have interacted with a lot of items in the past.

As can be seen in Figure 1, we count the numbers of interactions with only ratings and the ones with both ratings and reviews. We can see that most interactions are without reviews in Figure 1(a). To be more specific, in Figure 1(b), the majority of users have a review missing rate of 75%-100%. Then, the interactions with only ratings are important to reveal more comprehensive user preferences

* Corresponding authors.

Proceedings of the 31st International Conference on Computational Linguistics, pages 7870–7880
January 19–24, 2025.

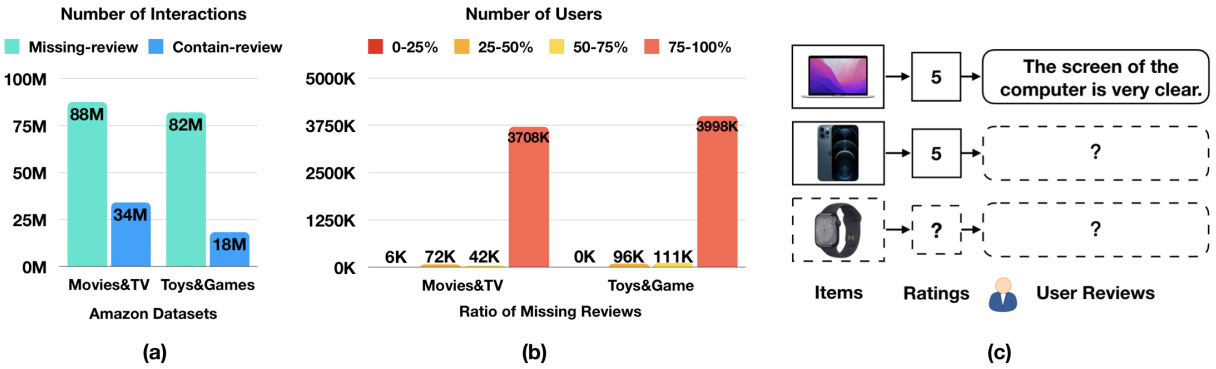

Figure 1: Figures that present the review-missing situations on the two real-world datasets. (a) shows the number of interactions that miss reviews or contain reviews. (b) presents the extent to which reviews are missing for users. (c) illustrates an example showing that leveraging interaction records without reviews can improve model performance, where the question mark represents unknown ratings or reviews for users.

and also can provide supports for other interactions to generate reasonable explanations. For instance, in Figure 1(c), the user has two interactions: (A) she purchases a laptop, leaving a positive score and a review on the screen, and (B) she scores an iPhone with 5-star, but does not post reviews. Suppose we need to predict the review of the user for an Apple watch, then according to (B) and the collaborative nature, we know that the user may give a positive score to the Apple watch due to its brand. According to (A), we know that when the user gives a positive score on a digital item, then she may comment on the screen. By combining the above two conditions, the model can predict that the user may post a review including "screen" on the Apple watch. If we do not have the information of (B), the prediction can be less accurate. As a result, the interactions with only rating information are important for the review generation, which has been ignored in existing generative explainable recommender models.

In this paper, we propose to incorporate interactions without reviews to improve explainable recommender models(called **IWRER** for short). In specific, we design a general adversarial framework, where the review prediction task is converted to a reinforcement learning problem, and we predict the reviews for the rating-only interactions to retrain explainable recommender models. To make the review prediction process more smoothly, we introduce a self-paced learning technique, where shorter reviews are optimized before the longer ones. The main contributions of this paper can be summarized as follows:

- We propose the idea of incorporating interactions without reviews to conduct explainable recommendation, which is the first work in the domain of generative explainable recommendation, to the best of our knowledge.
- We design an adversarial framework to implement the above idea, and introduce self-paced learning to improve the training process.
- We conduct extensive experiments along with qualitative and quantitative analysis to demonstrate the effectiveness of our framework.

2 Related Work

Explainable recommendation systems enhance user trust and satisfaction by providing insightful explanations. There are many forms of recommendation explanations (Zhang et al., 2020). For instance, (Peake and Wang, 2018; Balog et al., 2019) are rule-mining methods, (Chen et al., 2019) provides visual explanations, (Sharma and Cosley, 2013) generates explanations based on social networks, and (Chen et al., 2021) generate natural languages for explanations. For the last way, early models such as NRT (Li et al., 2017b) use multi-layer perception to predict ratings and generate explanations with GRU. To ensure the controllability and expressiveness of generated sentences, NETE (Li et al., 2020) incorporates features and sentiment analysis into the process of text generation. To generate more personalized explanations, PETER (Li et al., 2021) utilizes an attention masking matrix and transformer blocks, incorporating user and item IDs into the transformer. PEPLER (Li et al., 2023) leverages user and item IDs as prompts to incor-

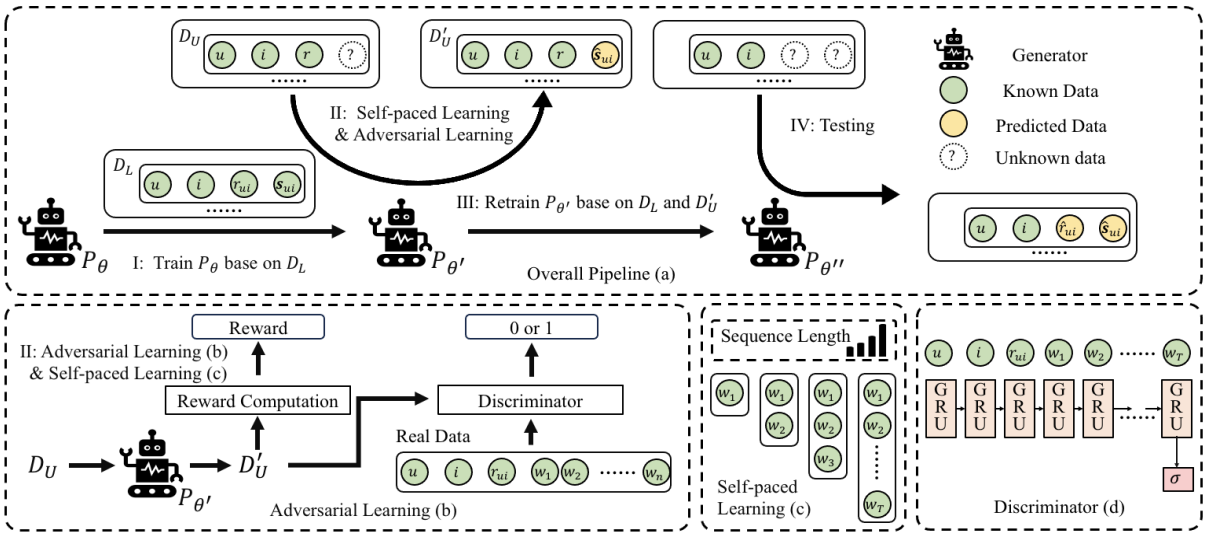

Figure 2: Our model architecture. (a) Overall pipeline. (b) Adversarial learning for review generation. (c) Self-paced learning. (d) Discriminator. I to IV are four steps representing the pre-training, review generation, model retraining, and testing stage.

porate the knowledge of pre-trained models into explainable recommender models.

However, all of the above models filtered the data in which reviews are missing, which also reflects user preferences. Therefore, we propose our framework IWRER, which can incorporate these interactions to capture user preferences more accurately and comprehensively, so that the model performance can be improved.

3 Problem Definition

Let us consider a user set $\mathcal{U}$ and an item set $\mathcal{I}$. The interactions between these users and items are gathered into a set $\mathcal{O} = \{(u, i, r_{ui}, \boldsymbol{s}_{ui})|u \in \mathcal{U}, i \in \mathcal{I}\}$, where each element represents that user u has interacted with item i, and the rating and review are r_{ui} and $\boldsymbol{s}_{ui}$, respectively. The review $\boldsymbol{s}_{ui}$ is a sequence of word tokens, that is, $\boldsymbol{s}_{ui} = \{w_{ui}^1, w_{ui}^2, ..., w_{ui}^{l_{ui}}\}$, where l_{ui} is the length of the review. The task is to learn a model, so that we can accurately predict the rating and review for a given user-item pair.

Traditional models usually assume that $\boldsymbol{s}_{ui}$ is visible in all the samples (Li et al., 2020, 2023, 2017a). However, as mentioned before, such assumption is unreasonable and impractical. Thus, in our problem, we allow $\mathcal{O}$ to have review-missing interactions, that is, we have two sets $D_L = \{(u, i, r_{ui}, \boldsymbol{s}_{ui})\}$ and $D_U = \{(u, i, r_{ui})\}$, where in D_L, the interactions are all accompanied with reviews, but in D_U, we only have ratings for the interactions. Our task is to design a unified framework to leverage both D_L and D_U to better learn an explainable recommender model for rating prediction and explanation generation, which is basically a semi-supervised learning problem.

4 Methodology

The overall pipeline of our framework is presented in Figure 2. We have four steps. The first step is pre-training an explainable recommender model based on D_L. Then, we generate reviews for D_U based on reinforcement learning in an adversarial manner. To learn the generation model more smoothly, we introduce a self-paced learning technique to schedule the training samples in an easy to hard manner. In the next, the original and generated reviews are both leveraged to retrain the explainable recommender model. At last, we use the learned model to make predictions in the test set. In the subsequent section, we present a detailed explanation of our framework.

4.1 Review Generation as a Reinforcement Learning Problem

Generating reviews for D_U based on D_L is basically a data augmentation problem. However, different from traditional settings, where the key task is to predict binary or multi-category labels, our problem needs to generate user reviews, which are sequential word tokens. Such differences bring significant challenges for us to generate high quality user reviews. To begin with, since the words are

represented by discrete IDs, it is hard to directly propagate gradients in a fully differentiable manner. Then, the prediction error of the current words may influence the accuracy of the subsequent inference, and the accumulated error can be considerably amplified as the sequence length becomes larger.

To solve the first problem, we regard the review generation process as a reinforcement learning problem, where the model for generating reviews is updated based on the reward for measuring the review quality. However, how to comprehensively and accurately measure the review quality is not easy, since there can be too many perspectives to evaluate the reviews, and it is labor-intensive for designing the evaluation rules. To alleviate this problem, we take inspiration from adversarial learning to design an automatic evaluation strategy. In specific, we suppose the real user reviews have the highest quality, and if the model generated reviews are more similar to the real ones, then they should also have higher quality. We deploy a discriminator to measure the review quality, which is expected to output 1 for the real reviews and 0 for the ones that are far from them. In addition, the review generation model is optimized to produce high quality reviews to fool the discriminator. We formulate the state, action, reward, and policy of the designed reinforcement learning problem at step t as follows, detailing each component for clarity: the **state** s_t is $(u, i, r_{ui}, w_1, w_2, ..., w_{t-1})$, where u and i denote user and item separately, and $\{w_1, w_2, ..., w_{t-1}\}$ is the subsequence that has already been generate; the **action** is the word to be generated at step t; the **reward** is computed based on the discriminator after the complete review has been generated; the **policy** $P_{\boldsymbol{\theta}}$ chooses a word w_t based on the current state s_t, where $\boldsymbol{\theta}$ is the parameter set of the policy.

Suppose the discriminator is $J_{\boldsymbol{\varphi}}(\cdot) \in (0, 1)$, then we use the following minimax objective to iteratively optimize $P_{\boldsymbol{\theta}}$ and $J_{\boldsymbol{\varphi}}$:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathrm{E}_{\boldsymbol{s}\sim P_{\text{data}}}[\log J_{\boldsymbol{\varphi}}(\boldsymbol{s})] + \mathrm{E}_{\boldsymbol{s}\sim P_{\text{gen}}}[\log(1 - J_{\boldsymbol{\varphi}}(\boldsymbol{s}))], \tag{1}$$

where $\boldsymbol{s} = (u, i, r_{ui}, w_1, w_2, ..., w_T)$ is a complete review. P_{data} and P_{gen} represent the distribution of the real user reviews and the ones generated by $P_{\boldsymbol{\theta}}$. In this objective, the generator $\boldsymbol{\theta}$ is optimized to produce reviews which can lead to larger $J_{\boldsymbol{\varphi}}$. The discriminator $\boldsymbol{\varphi}$ is learned to separate the reviews from the users and the generator $\boldsymbol{\theta}$. After convergence, the generator is expected to produce reviews that can not be identified by the discriminator. The empirically version of equation (1) can be written as follows:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \sum_{\boldsymbol{s}\sim O_{\text{data}}} [\log J_{\boldsymbol{\varphi}}(\boldsymbol{s})] + \sum_{\boldsymbol{s}\sim O_{\text{gen}}} [\log(1 - J_{\boldsymbol{\varphi}}(\boldsymbol{s}))], \tag{2}$$

where O_{data} and O_{gen} are the reviews in D_L and the ones generated from $P_{\boldsymbol{\theta}}$, respectively.

4.2 Review Generation via Self-paced Learning

As mentioned before, word prediction errors may be accumulated in the sequence generation process. To overcome this challenge, we introduce a strategy called self-paced learning, to smooth the training process of the generator (Wang et al., 2022). In specific, we first let the generator produce shorter reviews. After it can well handle these samples, we gradually increase the review length. Intuitively, when the reviews are shorter, the model only needs to handle smaller accumulation errors, which builds easy optimization tasks. Once the model can generate good enough short sequences, we increase the review length. At this time, the model can already generate accurate short sequences. Thus, longer reviews may not lead to too large accumulation of errors.

To achieve the above idea, an important challenge is how to control the sequence length of the generated reviews adaptively in the optimization process. Before solving this challenge, we first take a closer look at the optimization process of $\boldsymbol{\theta}$. To begin with, suppose $L = \sum_{\boldsymbol{s}\sim O_{\text{gen}}}[\log(1 - J_{\boldsymbol{\varphi}}(\boldsymbol{s}))]$, we have: where we use $\hat{P}$ to represent the joint distribution of all the words in a sequence. The length of the generated review is T. To control the sequence length, we improve $P_{\boldsymbol{\theta}}$ to a new policy as follows:

$$P_{\boldsymbol{\theta}}^{'} = \begin{cases} P_{\boldsymbol{\theta}}(w_t|u, i, w_{1,...,t-1}) & \text{If } t < g(\boldsymbol{\mu}) \\ \kappa & \text{If } t > g(\boldsymbol{\mu}) \text{ and } w_t = \text{EOS} \\ \dfrac{(1-\kappa)}{|\mathcal{V}|} & \text{If } t > g(\boldsymbol{\mu}) \text{ and } w_t \neq \text{EOS} \end{cases} \tag{3}$$

where $P_{\boldsymbol{\theta}}^{'}$ represents $P_{\boldsymbol{\theta}}^{'}(w_t|u, i, w_{1,...,t-1})$, $|\mathcal{V}|$ is the size of the word vocabulary. $g(\boldsymbol{\mu})$ is a self-paced regularizer. "EOS" is a special token indicating the complete review has been generated. When the generation step is smaller than $g(\boldsymbol{\mu})$, this policy is the same as the original one. After the sequence is longer than $g(\boldsymbol{\mu})$, the policy outputs "EOS" with a large probability κ, and the other words only have very small chances to be generated. By this operation, the length of the generated review is smaller

than $g(\boldsymbol{\mu})$ with a large probability. In the training process, we gradually enlarge $g(\boldsymbol{\mu})$ to increase the review length, such that $\boldsymbol{\theta}$ can be optimized in an easy to hard manner. To this end, we revise objective L as follows:

$$L = \sum_{\boldsymbol{s} \sim O_{\text{gen}}} [\log (1 - J_{\boldsymbol{\varphi}}(\boldsymbol{s}))] + g(\boldsymbol{\mu}), \tag{4}$$

where we let $g(\boldsymbol{\mu}) = \max\{a\boldsymbol{\mu}^2 + b, 10\}$, a and b are hyper-parameters that control the growth trend of $g(\boldsymbol{\mu})$.

By combining the above reinforcement learning and self-paced learning components, the training process of our framework is presented in Algorithm 1 in Appendix A.

4.3 Model Specification

For better understanding our framework, we introduce the implementation of the generator and discriminator of our framework in detail.

4.3.1 Implementation of Generator P_θ

The generator, denoted as $P_{\boldsymbol{\theta}}$, is responsible for simulating user-generated reviews on D_U based on the user-item interaction data $D_U \cup D_L$. In this paper, we use PETER (Li et al., 2021) and PEPLER (Li et al., 2023) as $P_{\boldsymbol{\theta}}$, because they are explainable recommendation models with the ability of natural language generation and rating prediction. As mentioned in section 3, after training on D_L, $P_{\boldsymbol{\theta}}$ can predict ratings and generate explanations given (u, i) in D_U.

4.3.2 Implementation of Discriminator J_φ

The discriminator $J_{\boldsymbol{\varphi}}$ plays a crucial role in judging whether the input data $(u, i, r_{ui}, w_1, w_2, ..., w_T)$ can form a reasonable sample. We implement it using GRU (Chung et al., 2014), which is adept at processing sequential data like text. The GRU-based architecture allows $J_{\boldsymbol{\varphi}}$ to capture and analyze the user behavior sequence, in which we leverage the the embedding of user u to initialize its first hidden state. The output of the GRU is then passed through the sigmoid activation function σ to obtain a probability score in (0, 1), indicating whether the sequence of user behavior constitutes a reasonable sample:

$$J_{\boldsymbol{\varphi}}(u, i, r_{ui}, w_1, w_2, ..., w_{l_{ui}}) = \sigma(\boldsymbol{b}^T \boldsymbol{h}_{l+1}) \tag{5}$$

where $\boldsymbol{h}_{l+1}$ denotes the last hidden state, $\boldsymbol{b}$ is a linear layer to project $\boldsymbol{h}_{l+1}$ into a scalar value.

5 Experiments

5.1 Experiments Setup

5.1.1 Datasets

We conduct our experiments based on three publicly available explainable recommendation datasets following (Li et al., 2020), which cover the area of Hotel, E-commerce and Restaurant.

	TA-HK	AZ-MT	YELP19
# user	9,765	7,506	27,147
# item	6,280	7,360	20,266
# records	105,570	226,362	463,737
# sparsity	99.83%	99.59%	99.92%
# domain	Hotel	E-commerce	Restaurant

Table 1: Statistics of the three real-world datasets.

TripAdvisor (TA-HK) is a travel dataset[1], where there are numerous ratings and reviews, reflecting consumer preferences for Hong Kong's hotels. **Amazon (Movies&TV)** is an e-commerce dataset[2], which provides extensive user reviews and ratings for a wide range of movies and TV shows. **Yelp Challenge 2019 (YELP19)** is a restaurant dataset[3] including diverse user reviews and ratings for restaurants across various businesses. The statistics of the above three real-world datasets are reported in Table 1.

5.1.2 Baselines

To assess the performance of our framework, we apply it to PETER and PEPLER, and compare it against the following methods: **Att2Seq** (Dong et al., 2017) is an encoder-decoder architecture that generates reviews by LSTM (Hochreiter and Schmidhuber, 1997); **NRT** (Li et al., 2017b) is an explainable recommendation model that predicts ratings by MLP (Rosenblatt, 1958) and generates reviews with GRU. **PETER** (Li et al., 2021) is a personalized explainable recommendation model that incorporates user and item information for explanation generation; **PEPLER** (Li et al., 2023) employs user and item IDs as prompts to enhance text generation. We apply our framework on PETER and PEPLER to verify the improvement, which are two explainable recommendation models. The resulting methods are referred to as IWRER(PETER) and IWRER(PEPLER), respec-

[1] https://www.cs.cmu.edu/~jiweil/html/hotel-review.html
[2] https://jmcauley.ucsd.edu/data/amazon/
[3] https://www.yelp.com/dataset

Metrics	BLEU (%)		ROUGE-1 (%)			ROUGE-2 (%)			RMSE (↓)
	B-1	B-4	F1	R	P	F1	R	P	
	TA-HK dataset								
MF	-	-	-	-	-	-	-	-	0.913
SVD++	-	-	-	-	-	-	-	-	0.881
Att2seq	14.871	0.940	15.271	14.805	17.867	1.896	1.914	2.179	-
NRT	14.619	0.854	15.529	14.616	18.655	1.909	1.899	2.207	0.815
PETER	14.904	0.822	16.176	16.806	17.047	1.963	2.147	1.998	0.849
IWRER(PETER)	**16.311**	**1.076**	**16.901**	**16.884**	**18.745**	**2.273**	**2.427**	**2.376**	**0.818**
PEPLER	15.593	0.993	15.962	15.461	19.012	2.110	2.107	2.493	0.823
IWRER(PEPLER)	**15.906**	**1.083**	**16.619**	**16.070**	**19.740**	**2.217**	**2.272**	**2.526**	**0.804**
	AZ-MT dataset								
MF	-	-	-	-	-	-	-	-	1.088
SVD++	-	-	-	-	-	-	-	-	1.008
Att2seq	12.558	0.956	15.214	13.621	20.118	2.073	1.963	2.647	-
NRT	10.887	0.739	15.091	12.795	21.221	1.903	1.718	2.566	0.978
PETER	13.335	1.011	15.543	14.281	19.547	2.111	2.022	2.617	0.967
IWRER(PETER)	**13.517**	**1.040**	**15.870**	**14.534**	**20.080**	**2.199**	**2.115**	**2.727**	**0.958**
PEPLER	12.403	0.889	15.046	13.730	19.306	1.874	1.790	2.443	0.976
IWRER(PEPLER)	**13.281**	**1.012**	**15.484**	**14.490**	**19.496**	**2.067**	**2.030**	**2.591**	**0.967**
	YELP19 dataset								
MF	-	-	-	-	-	-	-	-	1.154
SVD++	-	-	-	-	-	-	-	-	1.127
Att2seq	11.884	0.706	14.181	12.939	18.202	1.461	1.410	1.850	-
NRT	11.761	0.584	13.773	12.305	18.171	1.228	1.150	1.597	1.031
PETER	10.679	0.669	13.760	12.065	18.723	1.453	1.348	1.963	1.024
IWRER(PETER)	**12.049**	**0.761**	**14.519**	**12.942**	**19.240**	**1.602**	**1.579**	**2.006**	**1.018**
PEPLER	10.097	0.675	13.651	12.036	18.827	1.571	1.472	2.139	1.020
IWRER(PEPLER)	**11.592**	**0.795**	**14.141**	**12.956**	**18.940**	**1.652**	**1.639**	**2.193**	**1.017**

Table 2: Overall performance about representative baselines and our framework IWRER on PETER and PEPLER, in which we highlight the best results in bold fonts. "B-1", "B-4", "R" and "P" denote "BLEU-1", "BLEU-4", "Recall" and "Presion", respectively. We omit the "%" when reporting the values of BLEU and ROUGE, and leverage "-" for unavailable computations on some metrics. For RMSE, the sign "↓" indicates that lower values are related to better recommendation performance.

tively. For rating prediction, we additionally compare our framework with two collaborative filtering methods: **MF** (Shi et al., 2012) leverages latent vector representations for user-item preference estimation based on inner products to calculate ratings; **SVD++** (Koren, 2008) is an advanced collaborative filtering algorithm that incorporates implicit feedback along with explicit ratings.

5.1.3 Implementation Details

To construct a scenario where part of the reviews are missing, we randomly select 20% of the interaction data for each user and mask their reviews to get review-missing data. For the remaining 80% of the interaction records, we divide them into training, validation, and testing sets in an ratio of 8:1:1 , where we ensure that all the user-item interactions in the testing set have been seen in the training set. We evaluate baselines and our framework based on three well-known metrics including **RMSE** (Hotelling, 1992) for rating prediction, and **BLEU** (Papineni et al., 2002), **ROUGE** (Lin, 2004) for explanation generation. We constrain the length of the generated sentence to 15 and set the size of the vocabulary V to 20,000 following (Li et al., 2020, 2021). During the process of adversarial training, we optimize the generator $P_{\boldsymbol{\theta}}$ and discriminator J_{φ} alternately, with 4 rounds for $P_{\boldsymbol{\theta}}$ and 2 rounds for J_{φ}. For all models, we set the batch size to 128 and fix embedding size at 512. For hyperparameters such as hidden size and learning rate,

Method	TA-HK			AZ-MT			YELP19		
	B-1↑	B-4↑	RMSE↓	B-1↑	B-4↑	RMSE↓	B-1↑	B-4↑	RMSE↓
PEPLER	15.593	0.993	0.823	12.403	0.889	0.976	10.097	0.675	1.020
IWRER(PEPLER)-GAN-SPL	14.605	0.946	0.805	12.067	0.940	0.968	10.358	0.715	1.018
IWRER(PEPLER)-GAN	15.620	0.994	0.810	12.711	0.943	0.968	10.756	0.719	**1.017**
IWRER(PEPLER)-SPL	15.813	1.065	**0.804**	13.062	0.955	0.968	10.572	0.729	1.018
IWRER(PEPLER)	**15.906**	**1.083**	**0.804**	**13.281**	**1.012**	**0.967**	**11.592**	**0.795**	**1.017**

Table 3: Performance comparison on review generation and rating prediction between baseline PEPLER, our framework IWRER(PEPLER) and its variants, where "B-1", "B-4" and "SPL" denote "BLEU-1", "BLEU-4" and "self-paced learning", respectively. When reporting BLEU scores, the percent sign "%" is omitted. Bold fonts represent the best results between our framework and other models.

grid search is employed to determine them. Due to space constraints, we introduce them in detail in the appendix. See Appendix B for more information.

5.2 Overall Performance

The comprehensive comparison results are summarized in Table 2. We can see: among these baselines, PETER and PEPLER can obtain better results against other baselines in most cases, and can usually beat Att2seq and NRT in each evaluation metric for all datasets. This is consistent with the previous work (Li et al., 2021, 2023), and demonstrates the effectiveness of incorporation of user and item ID when generating explanations. NRT always performs better than MF and SVD++, because reviews also reflect user preferences, which can be leveraged to promote rating prediction. MF could achieve the worst performance, and SVD++ always beats MF benefiting from the incorporation of user history behaviors. Encouragingly, applying our framework to explainable recommendation models such as PETER and PEPLER can always enhance the model performance on all of the three datasets. This is because our method is capable of incorporating samples with missing reviews to capture user preferences more comprehensively with the help of reinforcement learning and self-paced learning, improving the performance of explanation generation and rating prediction tasks. On average, our method can enhance the best performance against PETER by about 11.88%, 4.12%, 10.03% on BLEU, ROUGE-1 and ROUGE-2 for explanation generation, and around 1.75% on RMSE for rating prediction. Compared with PEPLER, it can improve by around 11.71%, 4.36%, 6.99%, and 1.05% on these metrics separately. Overall, the above results demonstrate that by leveraging the user interaction records without reviews, our framework can predict more precise ratings and generate more accurate explanations.

5.3 Ablation Studies

To determine the necessity of each component and assess their individual contributions to the overall performance, we conduct a series of ablation studies. We remove single or both components of our framework and get the following three variants. IWRER(PEPLER)-GAN-SPL is a variant, in which we remove adversarial learning and self-paced learning of our framework when generating the missing reviews. IWRER(PEPLER)-GAN is a method, where we generate missing reviews without adversarial training. IWRER(PEPLER)-SPL is a model, where we remove self-paced learning and generate explanations in greedy search. The comparison between the variants and our framework is summarized in Table 3. We can see: for IWRER(PEPLER)-GAN-SPL, the performance may even decline on some metrics for the three datasets. This shows that directly using the pre-trained model for data augmentation does not always benefit generating more accurate explanations. Compared with our framework, the performance on BLEU and RMSE decreases when the adversarial training or self-paced learning module is removed. And their performance always beats IWRER(PEPLER)-GAN-SPL on BLEU-1 and BLEU-4. The reason is that for IWRER(PEPLER)-GAN, SPL enables the model to generate explanations from easy to difficult, making the learning process smoother. For IWRER(PEPLER)-SPL, GAN helps to optimize the generator by judging whether the generated user behaviors can form a reasonable interaction. Finally, the best result is obtained when combining each part of our framework. Overall, the above analyses demonstrate the necessity and effectiveness of adversarial learning as well as self-paced learning in our framework.

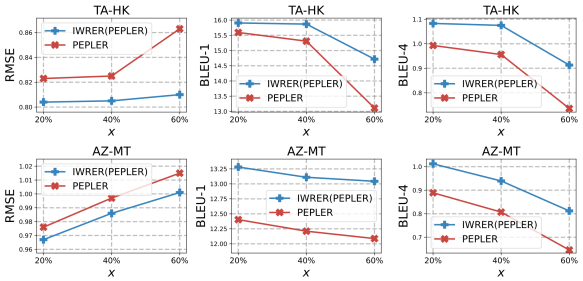

Figure 3: Influence of review missing ratio x on the model performance for RMSE, BLEU-1 and BLEU-4.

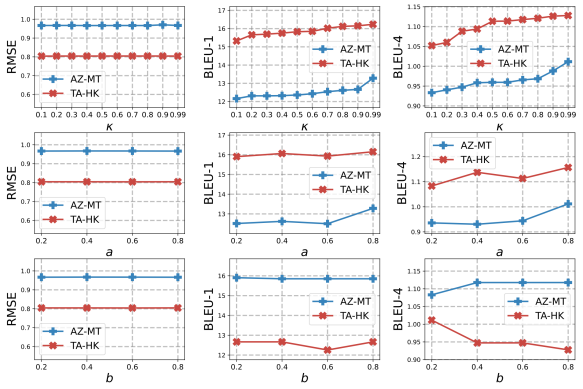

Figure 4: Influence of κ, a and b for self-paced learning on the model performance for RMSE, BLEU-1 and BLEU-4.

5.4 Parameter Analysis

5.5 Influence of review missing ratio x.

To identify whether our framework is always effective when different ratios of reviews are absent, we set it to 20%, 40%, and 60% and conduct verification based on PEPLER. The experiment results are presented in Figure 3, from which we can see: with the increase of the review missing ratio x, the effectiveness of both PEPLER and IWRER(PEPLER) decreases. Besides, for different absences of reviews, IWRER(PEPLER) can always outperform PEPLER on RMSE and BLEU. In addition, when there are more missing reviews, the performance improvement is even more effective for our framework IWRER(PEPLER) compared with PEPLER.

5.6 Influence of the key parameters in self-paced learning.

In our model, κ represents the stop probability in the review generation process. a and b are parameters that can affect how $g(\boldsymbol{\mu})$ grows up, which further indirectly determines when the sequence generation is likely to stop according to the equation(3). The experiment results are presented in Figure 4, from which we can see: with the rise of κ, a and b, the value of RMSE is little affected. This is because in our framework, self-paced learning is applied to explanation generation, rather than rating prediction. For BLEU-1 and BLEU-4, larger κ tends to lead to better results. The reason for this is that as the value of κ increases to a sufficiently high level, the function $g(\boldsymbol{\mu})$ has the capability to "halt" the process of text generation, so that the model can learn easier sentences at first. This demonstrates the effectiveness of self-paced learning. As for a, the best performance of BLEU-1 and BLEU-4 is achieved when it is equal to 0.8 for both of the above two datasets. Finally, when b is equal to 0.8 and 0.2, the best performance is achieved for **TA-HK** and **AZ-MT** datasets separately, indicating that the value of b needs to be carefully tuned in the experiments.

5.7 Explanation Evaluation

To determine whether our framework can intuitively improve the explainability of the generated explanations by incorporating data without reviews, we comprehensively evaluate these explanations. We first conduct a qualitative study to provide an intuitive understanding of our explanations. Then, we employ numerous labelers for a quantitative comparison of our framework against PEPLER.

5.7.1 Qualitative Analysis

Table 4 presents many examples of explanations generated by PEPLER and our framework IWRER(PEPLER), along with the ground truth. As we can see, IWRER(PEPLER) can always generate more accurate explanations than PEPLER. For the first example, the user pay attention to the distance between the hotel and the airport. PEPLER points out that the hotel is conveniently located but fails to describe its proximity to the "airport". IWRER(PEPLER) further takes it into account and generates the word "airport", which is more consist with the ground truth. In the second case, the consumer looks out for both the "bathroom" and the "shower". Our framework generates a reasonable explanation from these two aspects, while PEPLER only focuses on the "bathroom". In the last example, "lobby" and "staff" are mentioned in the review. IWRER(PEPLER) successfully pays attention to both of them. However, PEPLER incorrectly describes the size of the "bathroom" and fails to generate explanations on "staff". To sum up, benefiting from the incorporation of samples without reviews,

Model	Explanation
Ground Truth	**very convenient close to** the **airport**.
PEPLER	the hotel is located in a **very convenient location**.
IWRER(PEPLER)	the hotel is **very well located for access to** the **airport**.
Ground Truth	the **bathroom was large** and the **water pressure great** in the **shower**
PEPLER	the **bathroom was large** and the bed is very comfortable.
IWRER(PEPLER)	the **bathroom was very large** and the **shower was great**.
Ground Truth	the design of the **the lobby is absolutely gorgeous** and the **staffs were very nice**.
PEPLER	the **bathroom** was large and **well-appointed**.
IWRER(PEPLER)	the **lobby is well appointed** and the **staff is very friendly and helpful**.

Table 4: Qualitative analysis on the explanation generation, in which we highlight the matching words between ground truth and the generated explanations by PEPLER and our framework IWRER(PEPLER).

Dataset	TA-HK		AZ-MT		YELP19		Average	
Question	Q1	Q2	Q1	Q2	Q1	Q2	Q1	Q2
PEPLER	3.078	3.662	3.368	3.692	3.318	3.502	3.255	3.587
IWRER(PEPLER)	**3.736**	**4.210**	**3.598**	**3.876**	**3.654**	**3.818**	**3.694**	**3.968**

Table 5: Quantitative studies on the generated explanations by PEPLER and IWRER(PEPLER). We label larger values to bold fonts between them for each question, which indicate better model performance.

our model can capture user interests more comprehensively and accurately, and generate more accurate explanations.

5.7.2 Quantitative Analysis

To assess the real-world effectiveness of our model, we also perform a manual questionnaire evaluation. Specifically, we randomly select 50 sentences from each of the three datasets, with explanations generated by both PEPLER and IWRER(PEPLER), respectively. Then, we recruited 10 volunteers from the laboratory to manually score the above reviews on the following two questions (Wang et al., 2018):

- **Q1**: *Does the explanation help you to get more information about the recommended item?*
- **Q2**: *Does the explanation help you to make decisions more efficiently?*

For Q1 and Q2, the volunteers are asked to give a rating score from 5 to 1, representing strongly agree, agree, neutral, disagree, and strongly disagree, respectively. To maintain fairness, the volunteers are unaware of whether the explanations are generated by PEPLER or our framework IWRER(PEPLER) during the evaluation process. We compute the average scores for Q1 and Q2 across each dataset, and report the final results in Table 5. As illustrated, our framework IWRER(PEPLER) consistently achieves higher scores than PEPLER for both questions across all datasets. On average, our framework obtains 13.49% and 10.62% improvement on Q1 and Q2 separately. Overall, the above quantitative analysis demonstrates that our framework can successfully leverage the samples without reviews to generate more informative and accurate explanations.

6 Conclusion

In this paper, we propose an explainable recommendation framework to leverage samples without reviews. We employ reinforcement learning to complete the missing reviews and use self-paced learning to smooth the word sequence generation. Extensive experiments demonstrate that our framework can effectively leverage samples without reviews to improve the informativeness and persuasiveness of the generated explanations.

7 Limitation

Despite the promising results of our proposed IWRER framework, there are still several limitations. For one thing, we do not extract features from the generated reviews, and utilize them to benefit the explanation generation process. For another, our framework lacks the integration of large language models, which could be leveraged to model user profiles and historical behaviors to produce more reasonable and personalized explanations.

References

Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 265–274.

Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2021. Generate natural language explanations for recommendation. *arXiv preprint arXiv:2101.03392*.

Xu Chen, Hanxiong 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 *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 765–774.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

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

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

Harold Hotelling. 1992. Relations between two sets of variates. In *Breakthroughs in statistics: methodology and distribution*, pages 162–190. Springer.

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

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017a. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428.

Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 755–764.

Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized transformer for explainable recommendation. *arXiv preprint arXiv:2105.11601*.

Lei Li, Yongfeng Zhang, and Li Chen. 2023. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4):1–26.

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

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Georgina Peake and Jun Wang. 2018. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2060–2069.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 297–305.

Amit Sharma and Dan Cosley. 2013. Do social explanations work? studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1133–1144.

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146.

Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 165–174.

Zhidan Wang, Wenwen Ye, Xu Chen, Wenqiang Zhang, Zhenlei Wang, Lixin Zou, and Weidong Liu. 2022. Generative session-based recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2227–2235.

Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101.

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

A Algorithm of Our Framework

Algorithm 1 Learning Algorithmic of IWRER

Set N, N_P, N_J, M for iteration.
Initialize κ, a and b for self-paced learning.
Set the learning rate α.
Initialize $\boldsymbol{\mu}$, $\boldsymbol{\varphi}$, and $\boldsymbol{\theta}$.
Pretrain $P_{\boldsymbol{\theta}}$ based on the original dataset D_L, in which all the user-item interactions are accompanied by reviews.
for *k in [1, N]* **do**
  Select a user-item pair (u, i) from D_U, and generate a word sequence $\boldsymbol{s}_{ui}$ based on $P'_{\boldsymbol{\theta}}$, (u, i) and the rating score r_{ui}.
  Calculate the reward $r = J_{\boldsymbol{\varphi}}(u, i, r_{ui}, \boldsymbol{s}_{ui})$.
  for *i in [1, N_P]* **do**
    $\boldsymbol{\mu} := \boldsymbol{\mu} + \alpha \bigtriangledown_{\boldsymbol{\mu}} L(\boldsymbol{\theta}, \boldsymbol{\mu})$.
    $\hat{\boldsymbol{\mu}} := \boldsymbol{\mu}$.
    $\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha \bigtriangledown_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \hat{\boldsymbol{\mu}})$.
  end
  for *k in [1, N_J]* **do**
    Update the parameters $\boldsymbol{\varphi}$ in the discriminator $J_{\boldsymbol{\varphi}}$ based on:
    $\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathrm{E}_{\boldsymbol{s} \sim P_{\text{data}}}[\log J_{\boldsymbol{\varphi}}(\boldsymbol{s})] +$
    $\mathrm{E}_{\boldsymbol{s} \sim P_{\text{gen}}}[\log (1 - J_{\boldsymbol{\varphi}}(\boldsymbol{s}))]$,
    where $\boldsymbol{s} = (u, i, r_{ui}, w_1, w_2, ..., w_T)$, and T is the length of $\boldsymbol{s}_{ui}$.
  end
end
for *j in [1, M]* **do**
  Generate review $\boldsymbol{s}_{ui}$ based on the updated $P_{\boldsymbol{\theta}}$.
  Add $\boldsymbol{s}_{ui}$ to the sample set without reviews:
  $\hat{D}_U \leftarrow D_U \cup \{\boldsymbol{s_{ui}} | (u, i) \in D_U\}$.
end
Train $P_{\boldsymbol{\theta}}$ based on $D_L \cup \hat{D}_U$.

B Hyper-Parameter Settings in Detail

In specific, we tune the learning rate and hidden size in [0.0001, 0.001, 0.01, 0.1, 1.0] and [32, 64, 128, 256, 512] separately. The number of MLP layers is searched in [1, 2, 3, 4, 5], and the dropout ratio is tuned in [0.0, 0.1, 0.2, 0.3, 0.4]. For self-paced learning in our framework, we search a and b in [0.2, 0.4, 0.6, 0.8], and κ in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99]. The weight of l2 loss is searched in [0.0001, 0.001, 0.01, 0.1, 0, 1.0]. And the weight of rating loss, context loss and text loss are determined in the range of [1.0, 0.1, 0.01, 0.001, 0.0001].