A Tag-Based Post-Hoc Framework for Explainable Conversational Recommendation

Kerui Xu
Beijing University of Posts and Telecommunications
xukerui@bupt.edu.cn

Jun Xu*
Gaoling School of Artificial Intelligence
Renmin University of China
junxu@ruc.edu.cn

Sheng Gao†
Beijing University of Posts and Telecommunications
gaosheng@bupt.cn

Si Li
Beijing University of Posts and Telecommunications
lisi@bupt.edu.cn

Jun Guo
Beijing University of Posts and Telecommunications
guojun@bupt.edu.cn

Ji-Rong Wen
Gaoling School of Artificial Intelligence
Renmin University of China
jrwen@ruc.edu.cn

ABSTRACT

Explanations are important for conversational recommendation. They help users to understand how the recommender system works, and elicit user's responses by allowing users to provide informative feedback on them. During the interaction with users, explainable conversational recommender system provides explanations and collects user feedback to further refine the recommendations. Current conversational recommender systems, however, usually make recommendations with black-box prediction models, bringing difficulty in model explainability. Moreover, existing methods for explainable recommendation commonly provide one-shot explanations and fail to leverage user feedback. In this paper, we propose a **t**ag-based **p**ost-hoc framework for **e**xplainable **c**onversational **r**ecommendation (TPECR), which enables black-box recommendation models to provide explanations and refine recommendations based on user preference on tags (e.g., item attributes). Specifically, given the recommendation model being explained, TPECR trains a generation model to construct user embeddings based on their tag preferences. The explanations are provided by utilizing the generation model to estimate the contributions of different tags with respect to each item prediction. Given user feedback, the recommendation at the next turn is refined by tuning the tag preference and generating modified user embedding with the generation model. We instantiate the generation model with conditional variational auto-encoder (CVAE), which reconstructs user embedding conditioned on his tag preference. We conducted experiments by applying TPECR to different models and the results demonstrated the effectiveness of our TPECR on both synthetic and real datasets.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design theory, concepts and paradigms**; • **Information systems** → **Recommender systems**.

KEYWORDS

Conversational Recommendation, Explainable Recommendation, Post-Hoc Explanation

ACM Reference Format:
Kerui Xu, Jun Xu, Sheng Gao, Si Li, Jun Guo, and Ji-Rong Wen. 2022. A Tag-Based Post-Hoc Framework for Explainable Conversational Recommendation. In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '22), July 11–12, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539813.3545120

1 INTRODUCTION

Explainable recommendation, which provides both recommendation results and reasons underlying the system decisions, has attracted increasing attention in recent years. Explanations not only improve the transparency and user satisfaction of the recommendation system [41], but also elicit user's responses by allowing users to tell the system when it is wrong [31]. Traditionally, explainable recommendation methods only provide one-shot explanation and fail to leverage user feedback [6, 17, 24, 35]. The recently proposed explainable conversational recommendation integrates user feedback into explainable recommendation to enable bidirectional user-model communications [5] and achieves promising results.

Explainable conversational recommendation [5] introduces a new interaction paradigm, which triggers user feedback with explanations, as shown in Figure 1. The main tasks are to *make recommendation, generate explanations and refine recommendation considering user feedback*. Many previous works on conversational recommendation are not able to generate explanations. They collect user feedback by directly asking questions about user preference [7, 13, 30, 37, 42]. Users may feel difficult to answer these questions when they don't have a clear search target. Explainable

*Jun Xu is the corresponding author. Work partially done at Beijing Key Laboratory of Big Data Management and Analysis Methods.
†Work partially done at Guiyang National High-tech Zone.

ICTIR '22, July 11–12, 2022, Madrid, Spain

ACM ISBN 978-1-4503-9412-3/22/07…$15.00
https://doi.org/10.1145/3539813.3545120

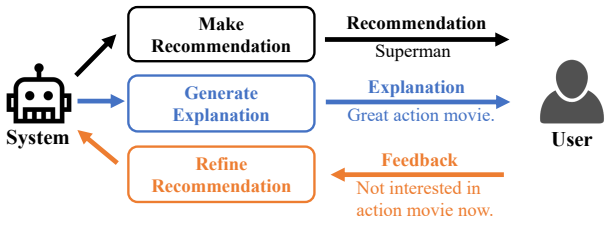


Figure 1: Pipeline of Explainable Conversational Recommendation. (Recommended Movie: Superman, Tag: Action)

conversational recommendation provides users with explanations to avoid cognitive overload and help users express their requests.

Explainable recommendation models can be either model-intrinsic or model-agnostic [41]. The model-intrinsic approach usually develops interpretable models with transparent decision mechanism, and model-agnostic (i.e., post-hoc) approach develops an explanation model to generate explanations after recommendations. In many cases, outstanding recommendation performances are usually achieved by complex models that are less interpretable [38]. While re-designing complex models into interpretable ones with similar performances is challenging, post-hoc approaches have the advantages of allowing the decision mechanism to be a blackbox and avoiding the trade-off between accuracy and explainability [41].

Explanations can serve several goals, including transparency (explaining how the system works) and scrutability (allowing users to tell the system it is wrong) [31]. During the interaction with users, the explanations help them understand the working mechanism of the system and trigger their feedback. Therefore, transparency and scrutability are especially important for explainable conversational recommendation. This paper tries to develop a model-agnostic explainable approach to achieve these two goals, which can be applied to different recommendation models. We utilize the tags as intermediary entity to bridge the gap between human-understandable concepts and uninterpretable black-box models. The tags can be item attributes or some keywords describing the items [2, 5] and are found to be helpful for both explainable recommendation [32] and conversational interaction [13, 45].

The proposed **t**ag-based **p**ost-hoc framework for **e**xplainable **c**onversational **r**ecommendation (TPECR) enables black-box recommendation model to provide explanations and refine recommendations based on user preference on tags. TPECR shows the user how the recommender system models his tag preference. The user can exam system's understanding of his preference and point out whether it's correct or not (scrutability). TPECR can also demonstrate how the system makes current recommendations based on the user's tag preference, helping the user understand how the system works (transparency). During the conversation, the user provides feedback to clarify his current interest and TPECR can refine recommendations by modifying tag preference accordingly.

Specifically, given the black-box recommendation model being explained, TPECR contains a generation model to construct user embedding based on his tag preference. The generation model is trained to capture the relation between user's latent vector and his tag preference, learning how the recommendation model expresses user's tag preference in latent space. Then the generation model, together with the recommendation model, formulates a composite function to describe how user's tag preference is utilized to make recommendations. To help users understand how the composite function maps their tag preferences to item predictions, TPECR generates explanations with feature importance methods [18, 26], which estimate the contributions of different tags with respect to each item prediction. After collecting user feedback on tags, TPECR modifies user's tag preference and uses the generation model to generate a new user embedding.

To implement TPECR framework, we instantiate the generation model with conditional variational auto-encoder (CVAE) [29], which generates the user's embedding conditioned on his tag preference. After mapping user's tag preference to current recommendations with the generation model, we apply LIME [26] to estimate the contributions of different tags. We evaluated the performance of CVAE as the generation model and explanations generated by TPECR on a synthetic dataset. We also applied TPECR to different recommendation models and conducted experiments on two real datasets to demonstrate the effectiveness of the proposed TPECR.

In summary, the contributions of this paper are as follow:

- We propose a tag-based post-hoc framework for explainable conversational recommendation (TPECR), which achieves transparency and scrutability with user's tag preference.
- TPECR contains a generation model, which is utilized to estimate the contributions of different tags to item prediction and generate new user embedding considering user feedback.
- We implemented the TPECR framework with CVAE as the generation model, and experimental results demonstrated the effectiveness of TPECR on both the synthetic dataset and two real datasets.

2 RELATED WORK

2.1 Conversational Recommendation

Conversational recommendation has been studied under different settings [8]. Christakopoulou et al. [7], Li et al. [16], Zhang et al. [40] focus on cold-start users in conversational recommendation with bandit-based algorithms. Lei et al. [13, 14], Sun and Zhang [30] study policy learning in multi-round conversational recommendation scenario. Luo et al. [19], Zhang et al. [42], Zou et al. [45] predict items by matching attribute query and item description. Chen et al. [4], Li et al. [15], Zhou et al. [44] model user preference through dialogue history and generate natural language response.

As for explainable conversational recommendation, Narducci et al. [22] build a graph composed of items and properties, and apply Personalized PageRank to make recommendations and provide explanations. Moon et al. [21] also utilize a knowledge graph to guide dialogue generation and provide explanations with walking path. Luo et al. [20], Wu et al. [36] revisit the critiquing approach with deep learning methods and use VAE-baed model to predict item score and its explanations. Chen et al. [5] design an incremental multitask learning framework considering recommendation prediction, explanation generation, and user feedback integration. Balog et al. [2] propose a transparent and scrutable user model, empowering users to understand recommendations made and improve the recommendations dynamically. Alkan et al. [1] shares similar

motivations and provides explanations by generating a justification text based on conversation history, while this paper tries to help user understand the working mechanism of the underlying model and trigger user's feedback.

2.2 Post-Hoc Explanations for Recommendation

Explainable recommendation methods can either be model-intrinsic or model-agnostic [41]. The model-intrinsic approach develops interpretable models while model-agnostic (i.e., post-hoc) approach develops an explanation model to generate explanations after recommendations. Peake and Wang [24] train association rules based on the outputs of a matrix factorisation black-box model. Cheng et al. [6] use the influence function to estimate the effect of training data on the predictions. Wang et al. [34] propose a reinforcement framework to extract sentences from reviews as explanations. Xu et al. [38] utilize a perturbation model to generate counterfactual examples and extracts causal rule for sequential recommendation. In this paper, we design a post-hoc framework for explainable conversational recommendation, which can provide explanations and incorporate user feedback to refine recommendations.

3 THE PROPOSED FRAMEWORK: TPECR

3.1 Problem Setting

TPECR utilizes tags as side information to generate explanations and refine recommendations. The explanation shows the user how the recommender system models his tag preference and utilizes it to make recommendation. Then the user provides his feedback on tags, which is further used to refine next-turn recommendations.

Formally, we denote the set of users as U and the set of items as V. To predict the affinity score r_{uv} between user $u \in U$ and item $v \in V$, the recommendation model first maps them into latent vectors $\boldsymbol{e}_u \in \mathbb{R}^d$ and $\boldsymbol{e}_v \in \mathbb{R}^d$, and then computes the score based on a function F of two vectors:

$$r_{uv} = F(\boldsymbol{e}_u, \boldsymbol{e}_v) \tag{1}$$

The set of tags is denoted as T and each tag $t \in T$ describes a feature of items (e.g., category of movies). $V_t \subseteq V$ denotes the set of items associated with tag t and $T_v \subseteq T$ denotes the set of tags related to item v. Please note that we don't require the recommendation model to take the tag information into consideration. The proposed framework can be applied to recommendation models without specialized module to encode tag information.

The framework needs to firstly infer user preference on all tags T, denoted as $\boldsymbol{c}_u \in \mathbb{R}^{|T|}$ where each element $c_{u,t} \in \boldsymbol{c}_u$ represents how likely the user will prefer tag t predicted by the recommendation model. Then, given the recommended item v, the framework shows how the tag preference $\boldsymbol{c}_u$ is utilized by estimating the importance scores of tag preference $\boldsymbol{I}_{uv} \in \mathbb{R}^{|T|}$. Each importance score $I_{uv,t} \in \boldsymbol{I}_{uv}$ represents the contribution of the corresponding tag t with respect to the predicted score r_{uv}. Finally, according to user feedback, the framework adjusts the predictions from the recommendation model and refines the recommendations.

3.2 Tag-Based Post-Hoc Framework

Figure 2 illustrates the proposed TPECR framework. It contains a generation model G, which constructs user embedding based on his tag preference. To generate explanations, TPECR estimates importance scores of tag preference $\boldsymbol{I}_{uv}$ based on the generation model and the recommendation model, showing how user's tag preference is modeled and contributes to current recommendation. To refine recommendations, TPECR modifies user's tag preference and utilizes the generation model to construct new user embedding, which is used to calculate refined item score by the recommendation model. Next, we will introduce the generation model G and how to generate explanations and refine recommendations in detail.

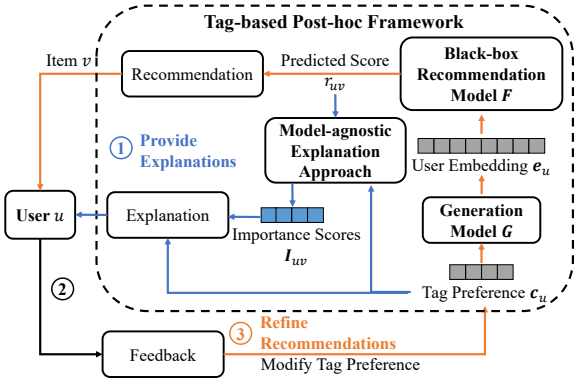


Figure 2: Overview of TPECR framework.

3.2.1 Generation Model. Given a user u, his tag preference $\boldsymbol{c}_u$ is inferred with item scores predicted by the recommendation model:

$$c_{u,t} = Agg(\{r_{uv} | v \in V_t\}), \forall t \in T \tag{2}$$

where $Agg(\cdot)$ refers to an aggregation function which maps a set of item scores into tag score. The function varies with different settings of the relationship between items and tags, which could be either a binary or real value [32]. Some recommendation models [13, 30] estimate user preference on both items and tags, and can directly predict $\boldsymbol{c}_u$ without inferring from item scores r_{uv}.

Given the tag preference $\boldsymbol{c}_u$ calculated in Equation (2), the generation model reconstructs user's corresponding embedding $\boldsymbol{e}_u$:

$$\boldsymbol{e}_u = G(\boldsymbol{c}_u)$$

where the function $G(\cdot)$ models the relation between user's latent vector and tag preference predicted by the recommendation model. Intuitively, the generation model learns how the recommendation model expresses user's tag preference in latent space.

3.2.2 Explanation Generation. TPECR provides explanations by showing the user how the recommender system models his tag preference and utilizes it to make recommendations. User's tag preference $\boldsymbol{c}_u$ is calculated in Equation (2). To infer predicted item score r_{uv} from tag preference $\boldsymbol{c}_u$, we formulate a composite function, which consists of the generation model G and the recommendation model F. Equation (1) can be rewritten as:

$$r_{uv} = F(G(\boldsymbol{c}_u), \boldsymbol{e}_v) = F_v \circ G(\boldsymbol{c}_u) \tag{3}$$

where $F_v(\cdot) = F(\cdot, \boldsymbol{e}_v)$.

After constructing the function that maps tag preference $\boldsymbol{c}_u$ to item prediction r_{uv}, TPECR applies existing model-agnostic explanation approaches [3] to help user understand how his tag preference

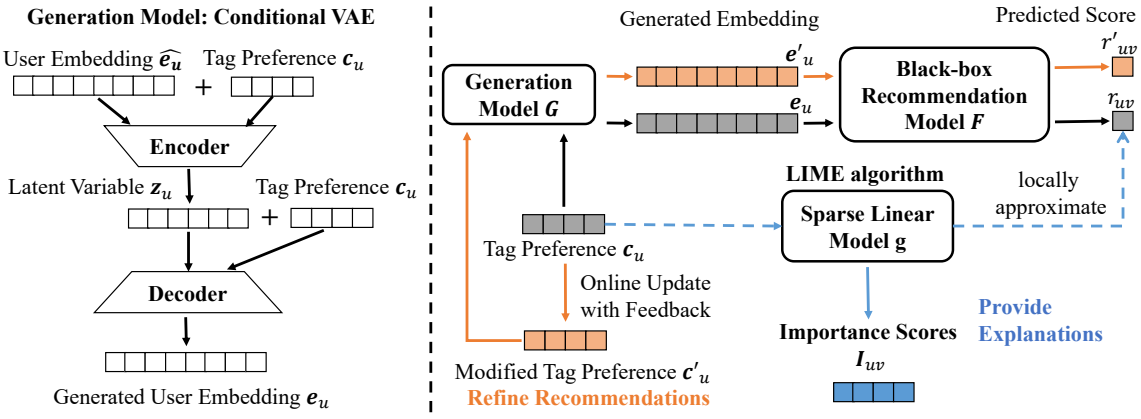


Figure 3: Our implementation of TPECR. We instantiate the generation model with conditional variational auto-encoder (left) and utilize the generation model to provide explanations and refine recommendations (right).

is utilized to make recommendation. Specifically, the contribution of each tag $I_{uv,t}$ with respect to the predicted score r_{uv} is estimated by applying feature importance methods [18, 26], which interpret model predictions by assigning an importance score to each input feature. Both the sign and the magnitude of importance score are considered to understand the contribution of each tag [3]. If $I_{uv,t} > 0$, the tag t contributes positively to the prediction r_{uv}; otherwise if $I_{uv,t} < 0$, it contributes negatively. The magnitude $|I_{uv,t}|$ represents how great the contribution of the tag t is to r_{uv}. Other choices of model-agnostic explanation methods would be a straightforward extension in future work. The way of presenting user's tag preference and importance scores can be different according to domain requirements, which is not constrained by TPECR.

3.2.3 Recommendation Refinement. TPECR refines recommendations by modifying user's tag preference and utilizing generation model G to generate new user embedding. After collecting user feedback, TPECR first changes original tag preference $\boldsymbol{c}_u$ correspondingly, resulting in new tag preference $\boldsymbol{c}'_u$. Here, we don't specify the way of adjusting tag preference, as it depends on the type of user feedback. For example, users can provide explicit signals, directly changing the value of $\boldsymbol{c}_u$. They can also provide implicit signals, telling the system whether they like or dislike the tags. Based on new tag preference $\boldsymbol{c}'_u$, the generation model G generates the corresponding user embedding $\boldsymbol{e}'_u$:

$$\boldsymbol{e}'_u = G(\boldsymbol{c}'_u)$$

The recommendation model F then predicts the refined item score r'_{uv} using the newly generated embedding $\boldsymbol{e}'_u$:

$$r'_{uv} = F(\boldsymbol{e}'_u, \boldsymbol{e}_v)$$

3.3 Implementation

To materialize TPECR, we provide an implementation considering implicit feedback from users. The recommendation model being explained is trained with user's interaction history and estimates the ranking scores of the unobserved items. User feedback on explanations informs the system whether he likes or dislikes the tags.

3.3.1 Generation Model: CVAE. We assume that the relationship between items and tags is binary, which means all tags associated with an item describe that item equally well [2]. User preference on tag $c_{u,t}$ is inferred by taking average of predicted scores of items associated with tag t:

$$c_{u,t} = \frac{\sum_{v \in V_t} r_{uv}}{|V_t|}, \forall t \in T$$

Despite its simplicity, Sharma et al. [27] found that for most users the rating on a set can be accurately approximated by the average rating of the items in that set.

We instantiate the generation model with conditional variational auto-encoder (CVAE) [29], a conditional directed graphical model whose input observations modulate the prior on Gaussian latent variables that generate the outputs. CVAE has been used to generate images conditioned on certain attributes [39] and diverse responses for neural dialog models [43]. Inspired by these works, we utilize CVAE to generate user embedding conditioned on his tag preference, as shown in Figure 3.

Given tag preference $\boldsymbol{c}_u$, CVAE samples latent variable $\boldsymbol{z}_u$ from the prior distribution $p_\theta(\boldsymbol{z}_u|\boldsymbol{c}_u)$, and then generates user embedding $\boldsymbol{e}_u$ conditioned on tag preference $\boldsymbol{c}_u$ and latent variable $\boldsymbol{z}_u$:

$$p_\theta(\boldsymbol{e}_u, \boldsymbol{z}_u|\boldsymbol{c}_u) = p_\theta(\boldsymbol{z}_u|\boldsymbol{c}_u) p_\theta(\boldsymbol{e}_u, |\boldsymbol{z}_u, \boldsymbol{c}_u).$$

CVAE is trained to maximizes the conditional log-likelihood of $\boldsymbol{e}_u$ given $\boldsymbol{c}_u$. As proposed in [12, 29], the parameters of CVAE can be estimated efficiently with stochastic gradient variational Bayes (SGVB), where the variational lower bound of the log-likelihood is used as a surrogate objective function to be maximized:

$$\begin{aligned}\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{q_\phi(\boldsymbol{z}_u|\boldsymbol{e}_u,\boldsymbol{c}_u)}[\log p_\theta(\boldsymbol{e}_u|\boldsymbol{z}_u,\boldsymbol{c}_u)] \\ -KL(q_\phi(\boldsymbol{z}_u|\boldsymbol{e}_u,\boldsymbol{c}_u)||p_\theta(\boldsymbol{z}_u)).\end{aligned} \tag{4}$$

Here, the prior distribution $p_\theta(\boldsymbol{z}_u)$ is assumed to follow isotropic multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The first term in Equation (4) represents the log likelihood of training samples and the second term is KL divergence term for regularization. In TPECR framework, CVAE works as a generation model to reconstruct user embedding, which is a high-dimensional variable with continuous values. Hence, we use Mean Squared Error to measure the similarity

between the reconstructed embedding $\boldsymbol{e}_u$ and original one $\widehat{\boldsymbol{e}_u}$ from the recommendation model. Equation (4) can be rewritten as:

$$\begin{aligned}\mathcal{L}_{\text{CVAE}} = &-\mathbb{E}_{q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u)}[MSE(\widehat{\boldsymbol{e}_u},\boldsymbol{e}_u)] \\ &-\lambda_{kl} * KL(q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u)||p_\theta(\boldsymbol{z}_u))\end{aligned} \tag{5}$$

where $q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u) \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$ is a multivariate Gaussian distribution, $\boldsymbol{e}_u$ is generated with $\boldsymbol{z}_u$ sampled from $q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u)$ and λ_{kl} is the regularization parameter to balance these two losses. Two Multi-Layer Perceptrons (MLPs), denoted as MLP_q and MLP_g, are used to approximate the distribution $q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u)$ and the generation process of $\boldsymbol{e}_u$ respectively:

$$[\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2)] = \text{MLP}_q([\widehat{\boldsymbol{e}_u},\boldsymbol{c}_u]),$$

$$\boldsymbol{e}_u = \text{MLP}_g([\boldsymbol{z}_u,\boldsymbol{c}_u])$$

where $[\cdot,\cdot]$ is the concatenation of the input vectors.

Given the modified tag preference $\boldsymbol{c}'_u$, new user embedding $\boldsymbol{e}'_u$ is generated as follows:

$$\boldsymbol{e}'_u = G(\boldsymbol{c}'_u) = \text{MLP}_g([\boldsymbol{z}^*_u,\boldsymbol{c}'_u]) \tag{6}$$

where the latent variable $\boldsymbol{z}^*_u$ is inferred without sampling: $\boldsymbol{z}^*_u = \mathbb{E}[q_\phi(\boldsymbol{z}_u|\widehat{\boldsymbol{e}_u},\boldsymbol{c}'_u)]$. Though CVAE is trained with the objective to reconstruct original user embedding, it can also generate new embedding with modified tag preference, as shown in our experiments.

3.3.2 Explanation Generation with LIME. TPECR applies feature importance methods to estimate the contribution of each tag with respect to item prediction. Specifically, we use LIME [26], an explanation method that explains the predictions of any classifier, to help users understand how the recommendation model works.

LIME explains the output of classifier by learning an interpretable model locally around the prediction [26]. To demonstrate how it works, we assume that the black-box model is denoted f and the instance being explained is x. The objective of LIME is to train a simple model g that minimizes following objective:

$$\xi(x) = \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

where $\mathcal{L}$ measures how unfaithful g is in approximating f in the locality defined by π_x and $\Omega(g)$ measures the complexity of model g. The model g is a sparse linear model, which is trained with perturbed samples drawing around x. The weights of the trained linear model g are treated as the feature importance explanations.

In TPECR framework, LIME is used to explain the composite function in Equation (3). However, directly applying LIME to ranking model is nontrivial since it's difficult to get the label for perturbed instance. Inspired by previous work [28], we reformulate ranking problem to classification problem and get the importance scores $\boldsymbol{I}_{uv}$ from the parameters of the trained linear model g.

Given user's tag preference $\boldsymbol{c}_u$ and the recommended item v being explained, $R(v|\boldsymbol{c}_u, V)$ denotes the rank of item v among the set of items V, i.e. ranking position in descending order according to predicted score $r_{uv} = F_v \circ G(\boldsymbol{c}_u)$. Let X be a random variable indicating whether user with tag preference $\boldsymbol{c}_u$ likes or dislikes the item v. We estimate the distribution of X as follow:

$$P(X = like|v, \overline{\boldsymbol{c}}_u, V) = \begin{cases} 1, & R(v|\overline{\boldsymbol{c}}_u, V) \le R(v|\boldsymbol{c}_u, V), \\ 0, & R(v|\overline{\boldsymbol{c}}_u, V) > R(v|\boldsymbol{c}_u, V), \end{cases}$$

$$P(X = dislike|v, \overline{\boldsymbol{c}}_u, V) = 1 - P(X = like|v, \overline{\boldsymbol{c}}_u, V).$$

where $\overline{\boldsymbol{c}}_u$ is the perturbed tag preference sampled by LIME and $R(v|\overline{\boldsymbol{c}}_u, V)$ is the rank of item v based on predicted score $\overline{r}_{uv} = F_v \circ G(\overline{\boldsymbol{c}}_u)$. With the setting of classification problem above, LIME learns feature importance considering how tag preference influences the rank of item v. Since it may be time-consuming to calculate the rank among all the items, we randomly sample k items as a smaller set $V_k \subset V$ and rank the item v among V_k.

3.3.3 Recommendation Refinement with Online Update. How to incorporate user feedback is an important problem for conversational recommender systems. Current works either develop specialized modules to adapt user preference [37, 42] or conduct online update to further optimize model parameters [7, 13, 19, 45]. TPECR utilizes online update to refine recommendations with user feedback, keeping the flexibility to be applied to different models.

Specifically, user feedback on tags is used to construct the loss function of online training $\mathcal{L}_{feedback}$. TPECR only updates user's tag preference $\boldsymbol{c}_u$ based on $\mathcal{L}_{feedback}$ and fixes other parameters (the generation model G and the recommendation model F) during the training process. After online update, the generation model G generates new user embedding $\boldsymbol{e}'_u$ with the modified tag preference $\boldsymbol{c}'_u$, and the recommendation model F makes refined recommendations based on new predicted score $r'_{uv} = F(\boldsymbol{e}'_u, \boldsymbol{e}_v)$. The loss function $\mathcal{L}_{feedback}$ is consistent with the loss function used for offline training the recommendation model F. During online update, the item scores are predicted with tag preference $\boldsymbol{c}_u$, i.e. $r_{uv} = F(G(\boldsymbol{c}_u), \boldsymbol{e}_v)$. For example, if the recommendation model F is Matrix Factorization (MF) trained with Bayesian Personalized Ranking loss [25], the loss function $\mathcal{L}_{feedback}$ should also be BPR loss with training instances constructed based on user feedback. Assuming that the user informs that he likes the tag t, we define the positive items for training as V_t and the negative items are sampled from other items unrelated to tag t. The loss function for positive feedback $\mathcal{L}_{feedback}$ is defined as:

$$\mathcal{L}_{feedback} = \sum_{(v^+,v^-)\in\mathcal{D}_{pos}} -\ln\sigma\left(F(G(\boldsymbol{c}_u), \boldsymbol{e}_{v^+}) - F(G(\boldsymbol{c}_u), \boldsymbol{e}_{v^-})\right)$$

where $\mathcal{D}_{pos} := \{(v^+, v^-)|v^+ \in V_t, v^- \in V \setminus V_t\}$ denotes the set of item pairs for positive feedback and σ is the sigmoid function. The training instances for user's negative feedback can be built in a similar way: $\mathcal{D}_{neg} := \{(v^+, v^-)|v^+ \in V_u, v^- \in V_t\}$, where V_u contains the items historically interacted by the user.

To conduct efficient training, we add a regularization term to the loss function $\mathcal{L}_{feedback}$, which concerns about balancing user's long-term and short-term preference. User feedback only reflects his current interest and the modified tag preference $\boldsymbol{c}'_u$ needs to consider both user's original preference and current feedback. Hence, the generated user embedding should not be far from original embedding $\boldsymbol{e}_u$ to avoid forgetting user's long-term preference:

$$\mathcal{L}_{reg} = MSE(G(\boldsymbol{c}_u), \boldsymbol{e}_u) \tag{7}$$

where MSE is Mean Squared Error loss.

The final loss function of online update is defined as follow:

$$\mathcal{L}_{online} = \mathcal{L}_{feedback} + \lambda_{reg}\mathcal{L}_{reg} \tag{8}$$

where λ_{reg} is the regularization parameter.

Table 1: NDCG and MSE of generated embedding.

Metrics	NDCG				
Num of Changes	1	2	3	4	5
Original	0.9702	0.9395	0.9054	0.8850	0.8537
MLP	0.9747	0.9500	0.9191	0.9004	0.8713
CVAE	**0.9999**	**0.9999**	**0.9999**	**0.9999**	**0.9999**
Metrics	MSE				
Num of Changes	1	2	3	4	5
Original	0.0321	0.0655	0.1027	0.1323	0.1735
MLP	0.0261	0.0531	0.0835	0.1075	0.1413
CVAE($\times 1 \times 10^{-6}$)	**1.782**	**2.046**	**3.521**	**7.635**	**2.763**

Table 2: Importance scores of all tags given an item, which is associated with the tag in brackets.

Scores	Tag1	Tag2	Tag3	Tag4	Tag5
Item(Tag1)	**0.2865**	-0.0767	-0.0774	-0.0658	-0.0711
Item(Tag2)	-0.0765	**0.2781**	-0.0701	-0.0687	-0.0707
Item(Tag3)	-0.0675	-0.0642	**0.2941**	-0.0700	-0.0776
Item(Tag4)	-0.0686	-0.0605	-0.0619	**0.2856**	-0.0607
Item(Tag5)	-0.0769	-0.0654	-0.0576	-0.0638	**0.2901**

4 EXPERIMENTS

4.1 Verifying CVAE and LIME on Synthetic Data

We first conducted experiments[1] to verify the correctness of CVAE and LIME used in TPECR, trying to answer two research questions:

RQ1: How is the performance of CVAE as generation model?

RQ2: Can LIME learn how the recommendation model utilizes tag preference to predict item score?

It's difficult, if not impossible, to get the ground-truth of user embedding generated with modified tag preference and know exactly how the recommendation model uses tag preference. To mitigate the issue, we construct a synthetic dataset with a simple recommendation model. Assuming there are N_u users, N_v items and N_t tags, the recommendation model predicts score as follows:

$$r_{uv} = \boldsymbol{e}_u^\intercal \cdot \boldsymbol{e}_v$$

where item embedding $\boldsymbol{e}_v \in \mathbb{R}^{N_v}$ is defined as one-hot encoded vector. We further assume that each item is related to only one tag and the recommendation model predicts item score solely based on user preference on its related tag, i.e. the related tag should be assigned with highest importance score:

$$r_{uv} = \boldsymbol{e}_u^\intercal \cdot \boldsymbol{e}_v = c_{u,t}, T_v = \{t\} \tag{9}$$

With the assumptions above, we randomly sample tag preference $c_{u,t}$ from $[0, 1)$ for each user and construct his embedding according to Equation (9). We set $N_u = 1000$, $N_v = 50$ and $N_t = 5$, and the items are evenly distributed to all tags, i.e., $|V_t| = 10, \forall t \in T$. The users are split into training data (90%) and test data (10%). As for generation model, MLP_q and MLP_g are Multi-Layer Perceptrons with one hidden layer whose size is 16 and the size of latent variable z_u is 8. Adam [11] is used as optimizer with learning rate 0.001 and the regularization parameter λ_{kl} is set to 1.

To answer RQ1, we randomly change user's tag preference by sampling from $[0, 1)$ again to simulate user's explicit feedback. The corresponding embedding is constructed according to Equation (9) as the ground truth. We use NDCG and MSE as evaluation metrics and compare the CVAE-generated embedding with the original one. We also trained a Multi-Layer Perceptron (MLP) as baseline model to generate new embedding based on the concatenation of user embedding and modified tag preference, which is same as CVAE. Table 1 reports the performance with different number of changed tags. From the results, we can see that original user embedding gradually fails to represent user's new preference as the number of changed tags increases, while CVAE can precisely reconstruct new embedding with modified tag preference. The results also showed that CVAE outperformed simple MLP significantly, indicating its effectiveness of reconstructing user embedding as generation model.

To answer RQ2, we demonstrate how LIME provides explanations given item predictions. For each tag t, we randomly sample an item v from V_t and calculate the importance scores for all the tags. Table 2 reports the average importance scores over 100 users. From the results, we find that LIME assigns the largest value to the tag associated with the given item, indicating that the recommendation model predicts item score mostly based on its related tag preference, which is consistent with our setting of the model.

4.2 Experiments on Real Data

We also conducted experiments on real datasets and applied TPECR to different models to answer the following questions[2]:

RQ3: Can the generation model learn from the recommendation model and express user's tag preference in latent space?

RQ4: Can TRECR incorporate user's feedback effectively?

RQ5: How does TRECR explain different models?

4.2.1 Datasets and Recommendation Models Being Explained. We used two datasets, Movielens-10M [9] and Goodreads-Comics [33]. Movielens-10M contains 10 million user-movie interactions and provides category information of each movie, which is used as tags. Goodreads-Comics contains user-book interactions for comics books with shelf names and we selected the informative shelf names as tags to ensure tag quality [23]. We filtered out the items with fewer than 5 interactions and the users fewer than 10 interactions. Table 3 reports the final statistics of these two datasets.

Table 3: Statistics of Movielens and Goodreads.

Dataset	#User	#Item	#Tags	#Interactions
Movielens-10M	69,878	7254	19	9,242,191
Goodreads-Comics	58,318	35,146	6	5,351,077

We applied TPECR framework to two widely-used latent factor models, Matrix Factorization (MF) and Neural Collaborative Filtering (NCF) [10], and one variant of factorization machine (FM) used

[1] The source code and the experiments have been shared at https://github.com/xxkkrr/TPECR.

[2] We focus more on evaluating the generation model since LIME can generate better explanations if the generation model approximates the composite function defined in Equation (3) more accurately.

Table 4: MAPE and MSE of tag scores calculated with generated embeddings.

Dataset	Movielens					
Model	MF		NCF		FM	
Metrics	MSE	MAPE	MSE	MAPE	MSE	MAPE
Original	0.0662	6.2693	27.2638	0.7515	0.3464	4.8758
MLP	0.0096	2.5566	24.6689	0.7035	0.0138	0.8933
CVAE	**0.0001**	**0.0700**	**1.8922**	**0.1027**	**0.0012**	**0.0854**
Dataset	Goodreads					
Model	MF		NCF		FM	
Metrics	MSE	MAPE	MSE	MAPE	MSE	MAPE
Original	0.0493	5.4942	7.5577	0.3706	1.0327	7.7594
MLP	0.0262	4.2085	6.2366	0.3296	0.0323	0.7988
CVAE	**0.0001**	**0.0879**	**1.2410**	**0.1345**	**0.0019**	**0.1556**

in previous conversational recommender systems as recommendation model [13, 30]. MF and NCF only predict item scores, while FM can also predict tag scores. To learn from user's implicit feedback, the MF and FM model are trained with BPR loss [25], and the NCF model is trained with binary cross-entropy loss [10]. We set the size of latent vector $d = 64$ for all the models.

To train the generation model, we split all users into training data (80%), validation data (10%) and test data (10%). We use Adam [11] as optimizer to train the generation model with loss function defined in Equation (5). MLP_q and MLP_g are Multi-Layer Perceptrons with two hidden layers whose size is 128 and the size of latent variable z_u is 48. We choose LeakyReLU as activation functions of MLP layers. For CVAE trained with MF, the learning rate is 0.0002 and the regularization parameter λ_{kl} is 0.0001. For CVAE trained with NCF, the learning rate is 0.001 and λ_{kl} is 0.001.

4.2.2 Performance of Generation Model. To answer RQ3, we change user's tag preference $c_{u,t}$ by adding different values d to it and see how the rank of item $v \in V_t$ changes with different tag preference $c_{u,t} + d$. If $d > 0$, the user prefers the tag t more and the generated embedding should make the recommendation model rank items in V_t higher. Otherwise, items in V_t should be ranked lower. Following [20], we use V_t to label ground truth "relevance", and Falling MAP (F-MAP) and Falling Recall (F-Recall) are calculated to measure the ranking difference of item set V_t:

$$F\text{-}MAP(u,t,K) = MAP@K_{V_t}^{before} - MAP@K_{V_t}^{after}$$

$$F\text{-}Recall(u,t,K) = Recall@K_{V_t}^{before} - Recall@K_{V_t}^{after}$$

We also calculate $F\text{-}MAP(u,t',K)$ and $F\text{-}Recall(u,t',K)$ of another randomly sampled tag t' to see whether the change of $c_{u,t}$ mainly affects the items related to tag t. Specifically, K is set as 1000. We randomly sample 1000 users from test data. For each user, we sample two tags: one is treated as the changed tag t and the other is the unrelated tag t'. The value d is taken from -0.5 to 0.5 at intervals of 0.1 for MF and from -1. to 1. at intervals of 0.2 for NCF and FM.

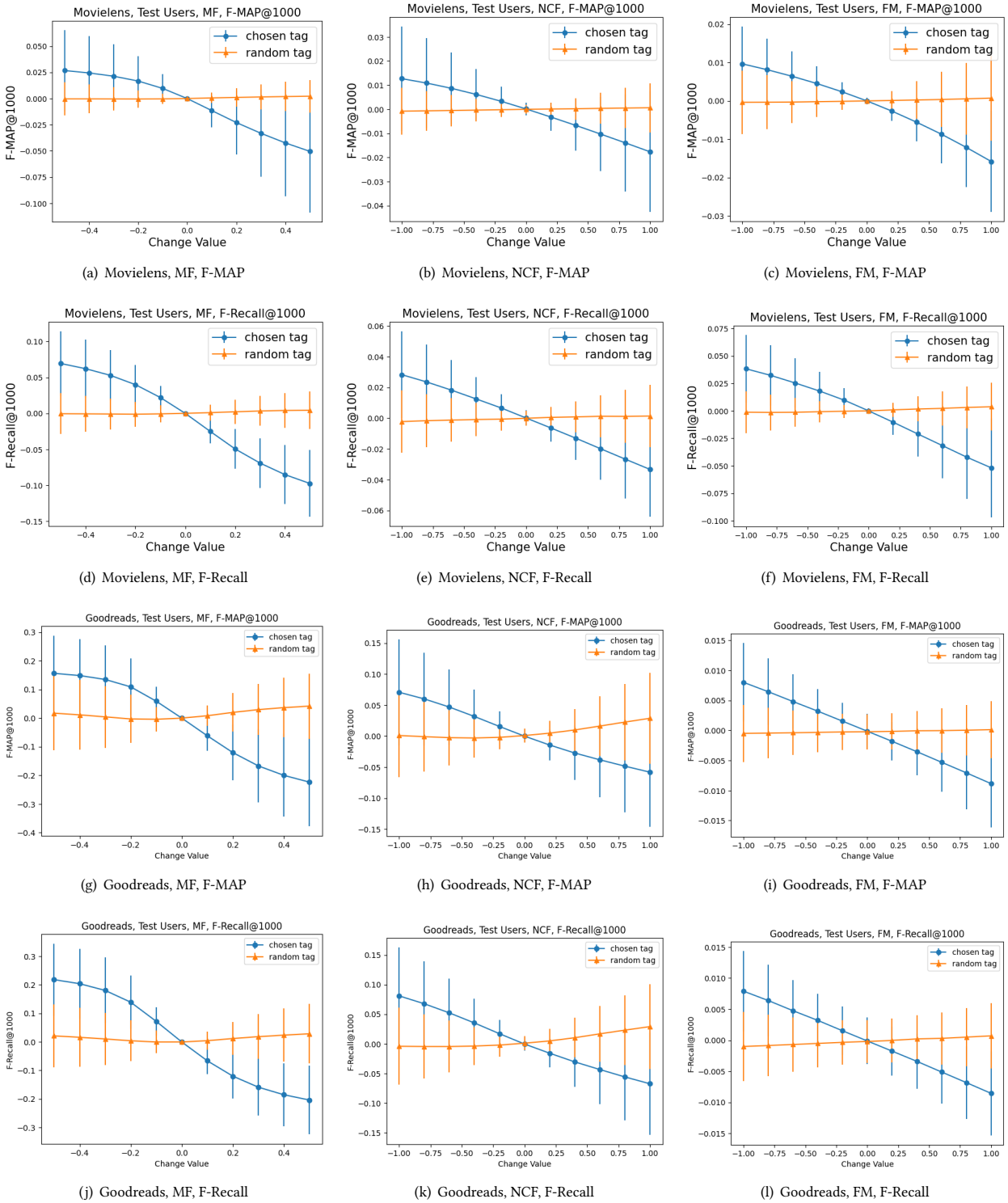
Figure 4 shows how F-MAP and F-Recall vary with different value d on two datasets. From the results, we can see that F-MAP and F-Recall gradually decrease when the value d increases from negative value to positive value, indicating that the items in V_t are ranked lower when $d < 0$ and ranked higher when $d > 0$. Moreover, the rank of item in V_t changes more with larger absolute value of d. We also find that when $d = 0$ (reconstructing the original embeddings), the generated embeddings produced similar rank lists with original ones (F-MAP and F-Recall close to 0), which means that CVAE can faithfully reconstruct original user embeddings. This phenomenon is observed in CVAE trained with all the models on both datasets. On the other hand, the rank of item in $V_{t'}$ is hardly affected. We note that $F\text{-}MAP(u,t',K)$ and $F\text{-}Recall(u,t',K)$ slightly increase with the value d, because higher rank of item in V_t leads to lower rank of other items $v \notin V_t$ and item in $V_{t'} \subseteq V \setminus V_t$ may be pulled down. From the results, we can see that CVAE can generate new user embedding based on the change of tag preference accordingly.

We also conducted experiments to see whether the generated embedding can represent user's new preference when he changes multiple tag preference. For each user, we randomly sampled another user's tag preference $\boldsymbol{c}'_u$ as modified preference and generated new user embedding $\boldsymbol{e}'_u$ with generation model. We treated $\boldsymbol{c}'_u$ as ground-truth scores and evaluated the tag scores calculated with user embedding $\boldsymbol{e}'_u$. Mean squared error (MSE) and mean absolute percentage error (MAPE) are used as evaluation metrics. We randomly samples 5000 users from test data and reported the average MSE and MAPE. We also use a MLP as baseline generation model, which generates new embedding based on the concatenation of user embedding and modified tag preference like CVAE.

Table 4 reports the experimental results on both datasets. We can see that CVAE outperformed baseline MLP on both datasets. CVAE also achieved better performance of generating embeddings given different recommendation models, which demonstrated the generalization ability of CVAE as the generation model. From the results, we can see that CVAE can learn how to express user's tag preference in the latent space and generate user embedding consistent with his tag preference.

4.2.3 Performance of Recommendation Refinement. To answer RQ4, we compare the recommendation result given user's original tag preference $\boldsymbol{c}_u$ and modified tag preference $\boldsymbol{c}'_u$ after training. Due to the lack of real user feedback in dataset, we simulate user feedback on tags and similar methods are adopted in previous works [13, 30, 37]. We compare our method with directly updating user embedding $\boldsymbol{e}_u$ to find out whether updating user's tag preference $\boldsymbol{c}_u$ can refine the recommendation effectively.

Specifically, we select user-item pairs from the test data and treat the item as user's target item representing his current interest. The user is assumed to like the tags associated with the target item and dislike others. He will provide feedback to inform the system whether it models his tag preference incorrectly. For example, if the tag preference $c_{u,t}$ is lower than the average score of all items, the recommendation model may think the user dislikes the tag t. When user's target item is related to the tag t, he will provide positive feedback to point out that he likes the tag t and we refer to this scenario as "PosTag". Similarly, "NegTag" means that the user provides negative feedback if he dislikes the tag t and the tag preference $c_{u,t}$ is higher than the average score. We conduct experiments in both scenarios. Besides using F-MAP@K and F-Recall@K to evaluate the ranking difference of items in V_t, we also report the ranking difference of the target item as F-Rank. Here,



(a) Movielens, MF, F-MAP (b) Movielens, NCF, F-MAP (c) Movielens, FM, F-MAP

(d) Movielens, MF, F-Recall (e) Movielens, NCF, F-Recall (f) Movielens, FM, F-Recall

(g) Goodreads, MF, F-MAP (h) Goodreads, NCF, F-MAP (i) Goodreads, FM, F-MAP

(j) Goodreads, MF, F-Recall (k) Goodreads, NCF, F-Recall (l) Goodreads, FM, F-Recall

Figure 4: F-MAP and F-Recall when adding different values to tag preference.

Table 5: Performance of Recommendation Refinement (M: Movielens, G: Goodreads).

Data&Rec	Update	PosTag			NegTag		
		F-MAP@1000	F-Recall@1000	F-Rank	F-MAP@1000	F-Recall@1000	F-Rank
M+MF	$\boldsymbol{e}_u$	-0.0486±0.0697	-0.0460±0.0589	30.59±166.58	0.0050±0.0060	0.0067±0.0058	2.85±19.60
	$\boldsymbol{c}_u$	-0.0430±0.0522	-0.0470±0.0460	36.00±182.07	0.0109±0.0134	0.0134±0.0113	7.53±75.12
M+NCF	$\boldsymbol{e}_u$	-0.0119±0.0159	-0.0137±0.0155	15.67±85.08	0.0018±0.0023	0.0039±0.0034	1.13±11.67
	$\boldsymbol{c}_u$	-0.0111±0.0130	-0.0144±0.0137	23.11±122.45	0.0069±0.0055	0.0169±0.0130	2.18±51.26
M+FM	$\boldsymbol{e}_u$	-0.0570±0.0695	-0.0539±0.0587	39.49±164.45	0.0055±0.0085	0.0058±0.0067	3.71±25.95
	$\boldsymbol{c}_u$	-0.0570±0.0551	-0.0578±0.0434	62.63±182.28	0.0106±0.0138	0.0137±0.0128	12.11±80.23
G+MF	$\boldsymbol{e}_u$	-0.0525±0.0560	-0.0693±0.0628	122.94±883.55	0.0187±0.0234	0.0163±0.0184	62.93±254.90
	$\boldsymbol{c}_u$	-0.0519±0.0496	-0.0697±0.0559	114.38±1098.46	0.0221±0.0279	0.0204±0.0240	24.20±499.96
G+NCF	$\boldsymbol{e}_u$	-0.0248±0.0291	-0.0351±0.0292	57.65±649.35	0.0231±0.0220	0.0212±0.0172	80.72±309.40
	$\boldsymbol{c}_u$	-0.0317±0.0350	-0.0483±0.0399	42.03±951.98	0.0121±0.0196	0.0119±0.0169	25.57±267.23
G+FM	$\boldsymbol{e}_u$	-0.1071±0.0990	-0.1050±0.0963	170.56±907.58	0.0289±0.0363	0.0275±0.0296	98.28±452.41
	$\boldsymbol{c}_u$	-0.1554±0.1674	-0.1434±0.1541	194.50±1292.99	0.0180±0.0252	0.0177±0.0212	24.56±387.19

Table 6: Ablation Study of Online Update on Movielens.

MF, PosTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	-0.1404±0.1372	-0.1253±0.0992	-232.12±1001.67
Ours	-0.0469±0.0521	-0.0503±0.0453	34.27±191.52
MF, NegTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	0.0194±0.0275	0.0257±0.0224	5.39±125.17
Ours	0.0108±0.0139	0.0135±0.0114	7.93±48.38
NCF, PosTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	-0.0509±0.0731	-0.0552±0.0684	-335.93±883.31
Ours	-0.0114±0.0130	-0.0146±0.0134	9.37±104.81
NCF, NegTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	0.0144±0.0189	0.0306±0.0218	4.49±120.02
Ours	0.0066±0.0076	0.0161±0.0123	5.00±44.79
FM, PosTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	-0.2010±0.1732	-0.1757±0.1176	-867.56±1849.78
Ours	-0.0574±0.0578	-0.0582±0.0457	42.25±169.64
FM, NegTag			
Metrics	F-MAP@1000	F-Recall@1000	F-Rank
$\lambda_{reg} = 0$	0.0211±0.0292	0.0290±0.0269	4.21±158.48
Ours	0.0099±0.0133	0.0132±0.0121	6.62±48.32

the items are ranked in descending order according to their scores. Specifically, we randomly sample 1000 user-item pairs and select a tag related to item for "PosTag" or "NegTag" as discussed above. We set the online update loss function as BPR loss for MF and FM model, and BCE loss for NCF model, and the training dataset are constructed as introduced in Section 3.3.3.

Table 5 reports the performance of recommendation refinement[3]. We observe that online training user's tag preference $\boldsymbol{c}_u$ achieves comparable performance with directly updating user embedding $\boldsymbol{e}_u$, indicating that TPECR can incorporate user feedback and refine recommendations effectively by modifying user's tag preference. We find that both positive and negative feedback help to refine recommendations and positive feedback contributes more considering the F-Rank value, since it directly reflects the characteristics of the target item [37]. However, we also note that online update isn't always helpful when the target item is already ranked high [13].

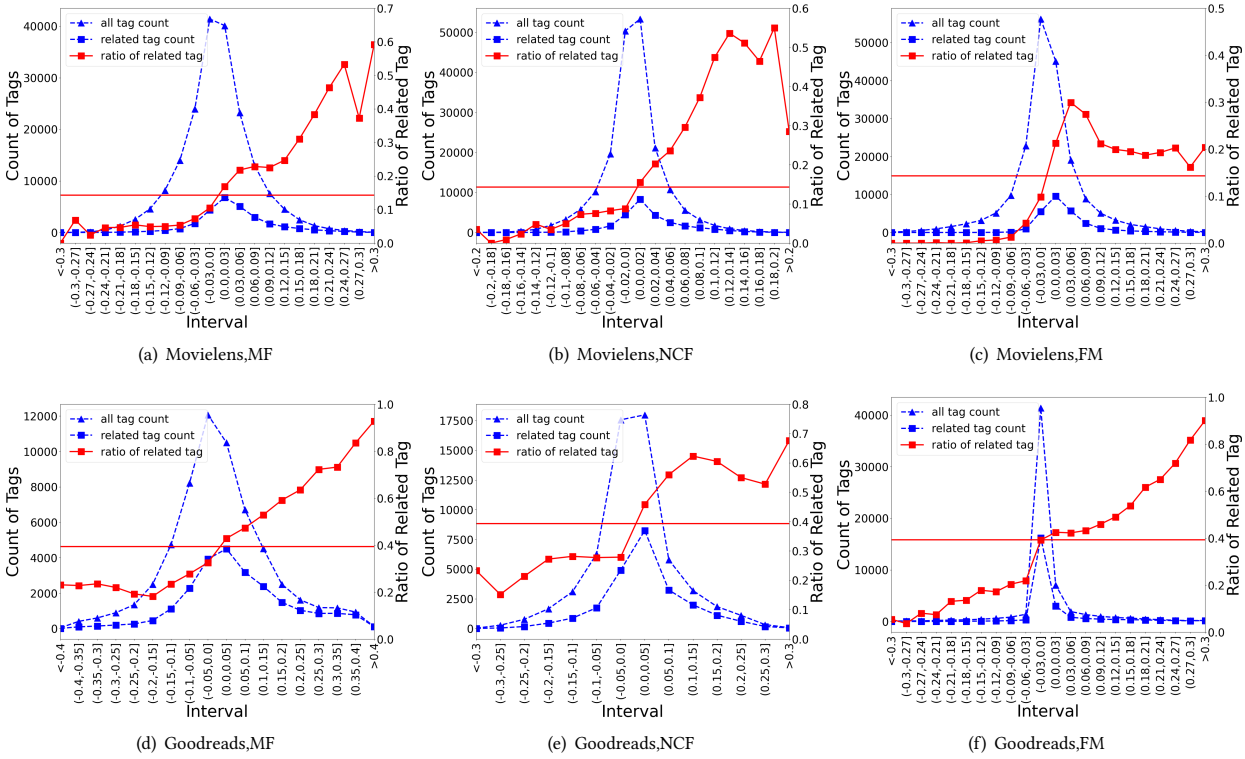
We further conduct an ablation study on Movielens dataset to see the contribution of the regulation term $\mathcal{L}_{reg}$ defined in Equation (7). The regulation term is removed by setting $\lambda_{reg} = 0$ and Table 6 reports the results. For "PosTag" scenario, the online update without $\mathcal{L}_{reg}$ can incorporate user feedback well but user's target item is ranked lower, which means that the generated embedding $\boldsymbol{e}'_u = G(\boldsymbol{c}'_u)$ focuses on user's current feedback only and forgets user's long-term preference. For "NegTag" scenario, since the training dataset $\mathcal{D}_{neg}$ contains user's historically interacted items, the generated embedding $\boldsymbol{e}'_u$ is less likely to forget long-term preference completely. However, without $\mathcal{L}_{reg}$, the generated embedding $\boldsymbol{e}'_u$ pays more attention to pulling down the rank of item in V_t instead of ranking the target item higher. The experimental results show that introducing the regulation term $\mathcal{L}_{reg}$ helps online update balance user's long-term and short-term preference.

4.2.4 Qualitative Evaluation of Explanation. To answer RQ5, we qualitatively evaluate the generated explanations. We demonstrate the importance scores assigned to different tags and draw the distribution of them. Specifically, we randomly sample 10000 user-item pairs and utilize LIME introduced in Section 3.3.2 to calculate the importance scores of all the tags with respect to item prediction. The size of V_k is 99, which consists of randomly sampled items.

Figure 5 demonstrates the distribution of importance scores. As shown in blue lines, we count the number of tags having importance scores in each interval and the number of tags related to the item being explained. The red lines represent the ratio of related tag count over all tag count in each interval, and the horizon one represents the average ratio of related tag count among all sampled items, i.e., $\sum_{v \in V_{sample}} |T_v| / (|V_{sample}| * |T|)$.

From these figures, we observe that most importance scores are centered around zero, indicating that only part of tag preference is

[3]We try updating user embedding with the regulation term, and the results are similar.



(a) Movielens,MF (b) Movielens,NCF (c) Movielens,FM

(d) Goodreads,MF (e) Goodreads,NCF (f) Goodreads,FM

Figure 5: Distribution of Importance Scores.

important when the recommendation model makes prediction. We also find that the ratio of related tags is higher than the average ratio when importance scores are larger than zero and it gradually increases considering higher importance scores interval. This phenomenon demonstrates that the tags related to the explained item usually have larger positive contributions than other unrelated tags, indicating that the recommendation model is more likely to rank the given item higher if the user prefers its related tags more.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a tag-based post-hoc framework for explainable conversational recommendation (TPECR), which enables black-box recommendation model to provide explanations and refine recommendations based on user's tag preference. TPECR contains a generation model to generate user embedding given his tag preference, and utilizes it to estimate the contributions of tag preference and generate new embedding considering user feedback. Experiments are conducted on both synthetic and real datasets, and the results validate the effectiveness of proposed framework.

Our work took the first step towards post-hoc framework for explainable conversational recommendation. In the future, we will explore other models besides CVAE as the generation model. We will also try to extend TPECR with more complex form of feedback, like pairwise tag interactions [2].

ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program of China (2019YFE0198200), the National Natural Science Foundation of China (61872338, 61832017), Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098), Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, and Public Policy and Decision-making Research Lab of Renmin University of China. This work was supported by project "Research on Key Technologies of Knowledge Graph in Power System Fault Management" in State Grid Corp. of China (52010119000F).

REFERENCES

[1] Öznur Alkan, Massimiliano Mattetti, Elizabeth M. Daly, Adi Botea, Inge Vejsbjerg, and Bart P. Knijnenburg. 2021. IRF: A Framework for Enabling Users to Interact with Recommenders through Dialogue. *Proc. ACM Hum. Comput. Interact.* 5, CSCW1 (2021), 1–25. https://doi.org/10.1145/3449237

[2] 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.*

[3] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. 2021. Benchmarking and Survey of Explanation Methods for Black Box Models. *CoRR* abs/2102.13076 (2021).

[4] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing.*

[5] Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. 2020. Towards Explainable Conversational Recommendation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020.*

[6] Weiyu Cheng, Yanyan Shen, Linpeng Huang, and Yanmin Zhu. 2019. Incorporating Interpretability into Latent Factor Models via Fast Influence Analysis. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[7] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[8] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *CoRR* abs/2101.09459 (2021).

[9] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* (2016).

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*.

[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

[12] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*.

[13] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*.

[14] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive Path Reasoning on Graph for Conversational Recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[15] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*.

[16] Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2020. Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-Start Users. *CoRR* abs/2005.12979 (2020).

[17] Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi. 2020. Explainable Recommender Systems via Resolving Learning Representations. In *The 29th ACM International Conference on Information and Knowledge Management*.

[18] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*.

[19] Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. 2020. Latent Linear Critiquing for Conversational Recommender Systems. In *The Web Conference 2020*.

[20] Kai Luo, Hojin Yang, Ga Wu, and Scott Sanner. 2020. Deep Critiquing for VAE-based Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*.

[21] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable Conversational Reasoning with Attention-based Walks over Knowledge Graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*.

[22] Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2018. Improving the User Experience with a Conversational Recommender System. In *AI*IA 2018 - XVIIth International Conference of the Italian Association for Artificial Intelligence*.

[23] Preksha Nema, Alexandros Karatzoglou, and Filip Radlinski. 2021. Untangle: Critiquing Disentangled Recommendations. https://openreview.net/forum?id=pdsec2YIOCx

[24] 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*.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

[26] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[27] Mohit Sharma, F. Maxwell Harper, and George Karypis. 2019. Learning from Sets of Items in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* (2019).

[28] Jaspreet Singh and Avishek Anand. 2019. EXS: Explainable Search Using Local Model Agnostic Interpretability. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*.

[29] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*.

[30] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[31] Nava Tintarev and Judith Masthoff. 2015. Explaining Recommendations: Design and Evaluation. In *Recommender Systems Handbook*.

[32] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*.

[33] Mengting Wan and Julian J. McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*.

[34] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A Reinforcement Learning Framework for Explainable Recommendation. In *IEEE International Conference on Data Mining*.

[35] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[36] Ga Wu, Kai Luo, Scott Sanner, and Harold Soh. 2019. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*.

[37] Kerui Xu, Jingxuan Yang, Jun Xu, Sheng Gao, Jun Guo, and Ji-Rong Wen. 2021. Adapting User Preference to Online Feedback in Multi-round Conversational Recommendation. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining*.

[38] Shuyuan Xu, Yunqi Li, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2020. Learning Post-Hoc Causal Explanations for Recommendation. *CoRR* abs/2006.16977 (2020).

[39] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2Image: Conditional Image Generation from Visual Attributes. In *Computer Vision - ECCV 2016 - 14th European Conference*.

[40] Xiaoying Zhang, Hong Xie, Hang Li, and John C. S. Lui. 2020. Conversational Contextual Bandit: Algorithm and Application. In *The Web Conference 2020*.

[41] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* (2020).

[42] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.

[43] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

[44] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving Conversational Recommender Systems via Knowledge Graph based Semantic Fusion. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[45] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards Question-based Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval,*.