

Personalized Prompt Learning for Explainable Recommendation

LEI LI, Hong Kong Baptist University, China
YONGFENG ZHANG, Rutgers University, USA
LI CHEN, Hong Kong Baptist University, China

Providing user-understandable explanations to justify recommendations could help users better understand the recommended items, increase the system's ease of use, and gain users' trust. A typical approach to realize it is natural language generation. However, previous works mostly adopt recurrent neural networks to meet the ends, leaving the potentially more effective pre-trained Transformer models under-explored. In fact, user and item IDs, as important identifiers in recommender systems, are inherently in different semantic space as words that pre-trained models were already trained on. Thus, how to effectively fuse IDs into such models becomes a critical issue. Inspired by recent advancement in prompt learning, we come up with two solutions: find alternative words to represent IDs (called discrete prompt learning), and directly input ID vectors to a pre-trained model (termed continuous prompt learning). In the latter case, ID vectors are randomly initialized but the model is trained in advance on large corpora, so they are actually in different learning stages. To bridge the gap, we further propose two training strategies: sequential tuning and recommendation as regularization. Extensive experiments show that our continuous prompt learning approach equipped with the training strategies consistently outperforms strong baselines on three datasets of explainable recommendation.

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

Additional Key Words and Phrases: Explainable Recommendation; Transformer; Pre-trained Language Model; Prompt Learning

ACM Reference Format:

Lei Li, Yongfeng Zhang, and Li Chen. 2023. Personalized Prompt Learning for Explainable Recommendation. *J. ACM* 37, 4, Article 111 (January 2023), 26 pages. <https://doi.org/10.1145/xxxxxxx.xxxxxxx>

1 INTRODUCTION

Traditional recommender systems help users overcome the information overload problem by providing personalized recommendations (e.g., movies or songs) that cater to their interests. Meanwhile, explanations that justify why these recommendations are made are becoming more and more important, as they can help users make better and faster decisions, increase the system's ease of use, and gain their trust in the system [52, 62]. There is a variety of explanation style, such as pre-defined templates [25, 51, 64], highlighted image regions [10] and automatically generated sentences [6, 27, 29]. The last type has gained increasing attention recently, mainly due to the availability of textual data on online commercial platforms, such as Amazon and Yelp, which encourage

Authors' addresses: Lei Li, Hong Kong Baptist University, 34 Renfrew Road, Hong Kong, China, csleili@comp.hkbu.edu.hk; Yongfeng Zhang, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey, USA, 08854-8019, yongfeng.zhang@rutgers.edu; Li Chen, Hong Kong Baptist University, 34 Renfrew Road, Hong Kong, China, lichen@comp.hkbu.edu.hk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

0004-5411/2023/1-ART111 \$15.00

<https://doi.org/10.1145/xxxxxxx.xxxxxxx>



Fig. 1. A review example from Yelp. The user and the restaurant are omitted for privacy protection.

users to express their opinions by writing reviews (see Fig. 1), as well as the advancement of natural language generation techniques, such as Recurrent Neural Networks (RNN), Transformer [55] and pre-trained language models [16, 18, 44].

In particular, recent years have witnessed the stronger and stronger language modeling capability of large pre-trained models. Taking Generative Pre-Training (GPT) series [4, 44, 45] as an example, the first generation GPT [44] after fine-tuning achieves the state-of-the-art in 9 natural language understanding tasks out of 12. Further, GPT-2 [45] without fine-tuning is able generate news articles that resemble authentic ones. More surprisingly, GPT-3 [4] could even do simple arithmetic (e.g., 2 digit multiplication) that the model was not trained or fine-tuned for. In the meantime, the size of these models and the volume of training data are becoming prohibitively large. Regarding model size, GPT has 117 million parameters, while GPT-2 and GPT-3 are increased dramatically to 1.5 billion and 175 billion, respectively. With respect to data, GPT takes as input 7000 books (approximately 7GB if a book has the size of 1MB), while GPT-2 and GPT-3 are fed 40GB and 570GB textual data, respectively.

As a consequence, it is nearly impossible to do customized modifications on the structure of these models. Moreover, it would also be challenging to incorporate into them user and item IDs, which are indispensable in recommender systems but are in very different semantic space as words that these models were trained on. No wonder most previous works [6, 14, 27, 50, 60] adopt RNN, such as Long Short-Term Memory (LSTM) [23] and Gated Recurrent Unit (GRU) [15], or small unpretrained Transformer [29] for explanation generation. This, however, makes the more effective pre-trained models less explored.

Fortunately, recent progress made in prompt learning [37] points out a promising way. Instead of modifying the structure of pre-trained models, researchers seek to adapt a given task to the models, so that they can directly model text probability. For instance, a prompt for sentiment classification could be constructed with the format of “I love this book. This book is”, where the underlined text is a specific sample and the remaining words are a hand-crafted template. This type of conditioning textual string is referred to as *discrete prompt*. After feeding it to a pre-trained model, a word prediction can be made at the end of the string, such as “good” or “bad”, indicating a positive or negative sentiment.

Likewise, we could also design discrete prompts for recommendation explanation generation. As IDs are inherently different from words, one naive and straightforward way is to convert IDs into words, such as movie titles and item features. We opt for the latter, and utilize features related to both the target user and the target item, since they represent the user’s explicit preferences as well as the item’s fine-grained attributes. Moreover, these features could guide the model to

talk about certain topics when generating explanations, such as “room” and “location” for hotel recommendations.

However, the conversion process from IDs into features may lose certain information, e.g., the identification role. Specifically, it is not very likely to convert an ID back from some features. For example, from the fact that Jerry loves cheese, we would not be able to certify that someone who enjoys eating cheese must be Jerry. Moreover, prompts do not have to be text strictly. They could be vectors, either randomly initialized or produced by another model. This type of prompt is formally termed *continuous/soft prompt*. In a similar way, we can also input ID vectors to a pre-trained model for explanation generation. Specifically, they are concatenated with the word vectors of an explanation before passing through the pre-trained model. It is unnecessary to do so for the aforementioned discrete prompt, because discrete prompt is composed of words (i.e., features) and thus is consistent with the model.

A follow-up problem of continuous prompt is that the model is already trained, but the ID vectors are randomly initialized, so they are actually in different learning stages. Recent study [2] finds out that such randomly initialized vectors could not be well optimized via stochastic gradient descent, and thus may lead to sub-optimal results. To cope with the problem, we propose two training strategies. The first strategy is called *sequential tuning*, where we separate the training into two stages: fine-tune continuous prompts (i.e., ID vectors) with the model frozen, and then update the parameters of both. The first stage would enable the continuous prompts to reach the same learning stage as the model, so that in the second stage they could be trained together. Our second strategy named *recommendation as regularization* is inspired by recent findings [11, 30, 49] in explainable recommendation that the explanation performance could be improved by the recommendation task. Indeed, the rating scores represent how much a user appreciates an item, which makes them an informative signal to the learning of explanation generation. Hence, we also leverage rating prediction task to augment the explanation task, and test two typical recommendation models, including Matrix Factorization (MF) [41] and Multi-Layer Perceptron (MLP).

We name our method PEPLER¹, which stands for “PErsonalized Prompt Learning for Explainable Recommendation”, where personalization is reflected by the IDs, either implicitly in the discrete prompts or explicitly in the continuous prompts. Without bells and whistles, our method consistently achieves the best performance against strong baselines (built on top of LSTM [23], GRU [15], Transformer [55] or BERT [16]) in terms of both text quality and explainability on three datasets.

In summary, our key contributions are:

- We propose PEPLER that generates natural language explanations for recommendations by treating user and item IDs as prompts. To the best of our knowledge, we are the first to introduce prompt learning to the community of recommender systems.
- We propose two training strategies to bridge the gap between continuous prompts and the pre-trained model, in order to enhance the explanation generation performance. In a broader sense, this may inspire researchers on how to better tune pre-trained language models.
- We evaluate the generated explanations on not only text quality metrics (such as BLEU and ROUGE), but also metrics that particularly focus on explainability from the angle of item features. Extensive experiments show that our method consistently outperforms state-of-the-art baselines.
- Our work may shed light on a broader scope of natural language generation fields that also need personalization, e.g., personalized conversational systems. In addition, it may point out a way for pre-trained models to deal with multi-modal data, e.g., image and text in captioning systems.

¹Codes available at <https://github.com/lileipisces/PEPLER>

In what follows, we first summarize related literature in section 2, and then present our explanation generation method PEPLER in section 3. Experimental setup and results analysis are given in section 4 and 5, respectively. We make a final conclusion and discuss future works in section 6.

2 RELATED WORK

2.1 Explainable Recommendation

Explainable recommendation [52, 62] has been studied from two major perspectives: human-computer interaction and machine learning. The former investigates how people perceive different styles of explanation [8, 9, 20], while the latter provides explanations by designing new explainable recommendation algorithms, to which our work is more related. There exist various types of explanation style, such as pre-defined templates [25, 51, 64], item features [21, 56], ranked text [5, 12, 28], image visualizations [10], knowledge graph paths [1, 19, 58, 59], and reasoning rules [7, 48, 67]. In this work, we focus on generating natural language explanations because they can be easily incorporated into different application scenarios, such as food recommender systems (e.g., Meituan² [61]) and conversational recommender systems [13, 33, 63]. However, previous works [6, 14, 27, 60] mostly rely on RNN, e.g., LSTM [23] and GRU [15], or unpretrained Transformer [29] for explanation generation, leaving the potentially more effective pre-trained models underexplored, which motivates this work.

2.2 Transformer and Pre-trained Models

Transformer [55] was first brought to the domain of machine translation with the architecture of encoder-decoder. Later works [16, 38] show that it remains effective, even when the encoder or the decoder is removed, reducing nearly half of model parameters. Under the paradigm of pre-training plus fine-tuning, Transformer's effectiveness has been confirmed on a wide range of natural language understanding tasks [16, 44], such as commonsense reasoning and question answering. More recently, it has been shown that pre-trained Transformer is able to perform novel tasks on which it was not targeted during training, e.g., arithmetic, after increasing both the magnitude of model size and the volume of training corpus [4, 45]. However, re-training such models may not be friendly to researchers who do not possess large amounts of computing resources. Therefore, there emerges a new research direction: prompt learning [37], where researchers adapt their tasks to pre-trained models, without the need of modifying or re-training them. Prompt learning has been successfully applied to many applications, such as domain adaptation [3], text summarization [34] and image captioning [54], because it allows pre-trained models that contain rich world knowledge to perform different tasks with task-specific prompts. In this work, we aim to provide users with high-quality recommendation explanations, so as to improve their experiences. To this end, we explore recommendation-related prompts, including discrete prompt and continuous prompt.

2.3 Personalized Natural Language Generation

Personalization of natural language generation plays a vital role in a large spectrum of tasks, such as explainable recommendation [6, 27, 29], review summarization [24], and dialog systems [63, 65]. In these tasks, user and item IDs are important identifiers for personalization. Previous approaches typically adopt MLP to encode the IDs into a context vector, from which RNN can decode a word sequence. This strategy can be found in many applications, such as review generation [17, 53], tip generation [31, 32] and explanation generation [14, 27]. However, it does not fit pre-trained models that were already trained on a massive amount of raw text. Probably because a proper solution to deal with heterogeneous data (i.e., IDs and words) is yet to be invented, previous works with

²<https://www.meituan.com/>

Table 1. Key notations and concepts.

Symbol	Description
\mathcal{T}	training set
\mathcal{U}	set of users
\mathcal{I}	set of items
\mathcal{V}	set of words
\mathcal{F}	set of features
\mathcal{E}	set of explanations
\mathbf{U}	embeddings of users
\mathbf{I}	embeddings of items
\mathbf{u}	embedding of user u
\mathbf{i}	embedding of item i
\mathbf{c}	probability distribution over the vocabulary
\mathbf{W}	weight matrix
\mathbf{w}, \mathbf{b}	weight vector
b	weight scalar
\mathbf{M}	attention masking matrix
Θ	model parameters
E	word sequence of an explanation
d, d_{ff}, d_h	dimension of representation
m	number of attention heads
n	number of Transformer layers
z	number of MLP hidden layers
$\text{ReLU}(\cdot)$	ReLU activation function
$\sigma(\cdot)$	sigmoid activation function
$\text{softmax}(\cdot)$	softmax function

Transformer or pre-trained models for personalized natural language generation replace IDs with text segments, such as persona attributes [65], movie titles [66] and item features [42], which is somewhat similar to our discrete prompt learning. But besides this, we further investigate how to incorporate into pre-trained models continuous prompts (i.e., ID vectors), in order to retain as much information as possible.

3 METHODOLOGY

The goal of our explanation task is to generate a natural language sentence $\hat{E}_{u,i}$ for a given user-item pair (u, i) to justify why i is recommended to u . The item i could be predicted for the user u by a recommendation model, e.g., matrix factorization [41], or resulted from his/her true behavior. At both training and testing stages, only user u and item i are used as input for producing the explanation. Hence, our proposed explanation generation approaches are compatible with any recommendation model, in which user and item IDs are indispensable.

In this section, we present the details of our methodology. First, we briefly go through Transformer, pre-trained language models, and prompt learning. Then, we introduce our proposed two methods for explanation generation, including discrete prompt learning and continuous prompt learning. After that, we illustrate how an explanation is generated during the inference stage. At last, we present two strategies for continuous prompt learning: sequential tuning, and recommendation as regularization.

Before introducing the technical details, we briefly explain the key terminology and notations. A *token* is a general term that can refer to *user ID*, *item ID*, *word* and *sub-word*. An *item feature* (e.g., “room”) is also a *word*, and thus can be seen as a *token*. A *discrete prompt* is a word sequence, e.g., several item features, while a *continuous prompt* is a sequence of vectors, e.g., user and item embeddings in this work. Key notations and concepts are given in Table 1. We use italic upper-case to denote a sequence of tokens, e.g., S , and italic lower-case to indicate its composing units, e.g., s . Meanwhile, a matrix is represented with bold upper-case, e.g., \mathbf{S} , and a vector is denoted as bold lower-case, e.g., \mathbf{s} , no matter whether they carry subscript or superscript or not.

3.1 Transformer, Pre-trained Language Models and Prompt Learning

To better demonstrate our work of PErsonalized Prompt Learning for Explainable Recommendation (PEPLER), we briefly go through Transformer and pre-trained language models that this work is built upon. Transformer [55] consists of n identical layers. The l -th layer encodes the previous layer’s output \mathbf{S}_{l-1} into $\mathbf{S}_l \in \mathbb{R}^{|S| \times d}$, where $l \in [1, n]$, $|S|$ is the length of the input token sequence, and d denotes the dimension of token representations/embeddings. Each layer is composed of two sub-layers: multi-head self-attention (MHSA) and position-wise feed-forward network (FFN). The latter is a two-layer FFN with the ReLU activation function. It performs linear transformations on the MHSA’s output $\mathbf{O}_l \in \mathbb{R}^{|S| \times d}$, and converts \mathbf{O}_l into \mathbf{S}_l ,

$$\mathbf{S}_l = \text{ReLU}(\mathbf{O}_l \mathbf{W}_{l,1} + \mathbf{b}_{l,1}) \mathbf{W}_{l,2} + \mathbf{b}_{l,2} \quad (1)$$

where $\mathbf{W}_{l,1} \in \mathbb{R}^{d \times d_{ff}}$, $\mathbf{b}_{l,1} \in \mathbb{R}^{d_{ff}}$, $\mathbf{W}_{l,2} \in \mathbb{R}^{d_{ff} \times d}$, $\mathbf{b}_{l,2} \in \mathbb{R}^d$ are weight parameters.

The MHSA sub-layer aggregates m attention heads, each of which is computed identically with the scaled dot-product attention (e.g., the h -th head in the l -th layer $\mathbf{A}_{l,h} \in \mathbb{R}^{|S| \times \frac{d}{m}}$). Formally, the computation of this sub-layer is defined as follows:

$$\begin{aligned} \mathbf{O}_l &= [\mathbf{A}_{l,1}, \dots, \mathbf{A}_{l,m}] \mathbf{W}_l^O \\ \mathbf{A}_{l,h} &= \text{softmax}\left(\frac{\mathbf{Q}_{l,h} \mathbf{K}_{l,h}^\top}{\sqrt{d}} + \mathbf{M}\right) \mathbf{V}_{l,h} \\ \mathbf{Q}_{l,h} &= \mathbf{S}_{l-1} \mathbf{W}_{l,h}^Q, \mathbf{K}_{l,h} = \mathbf{S}_{l-1} \mathbf{W}_{l,h}^K, \mathbf{V}_{l,h} = \mathbf{S}_{l-1} \mathbf{W}_{l,h}^V \\ \mathbf{M} &= \begin{cases} 0, & \text{Allow to attend} \\ -\infty, & \text{Prevent from attending} \end{cases} \end{aligned} \quad (2)$$

where $[\cdot, \cdot]$ represents the concatenation of matrices/vectors, $\text{softmax}(\cdot)$ denotes the softmax function, $\mathbf{W}_l^O \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_{l,h}^Q, \mathbf{W}_{l,h}^K, \mathbf{W}_{l,h}^V \in \mathbb{R}^{d \times \frac{d}{m}}$ are projection matrices to be learned, $\mathbf{S}_{l-1} \in \mathbb{R}^{|S| \times d}$ is the $(l-1)$ -th layer’s output, and $\mathbf{M} \in \mathbb{R}^{|S| \times |S|}$ is the attention masking matrix.

Each element in \mathbf{M} controls whether a token in the sequence can attend to another. For example, in bidirectional language models such as BERT [16], \mathbf{M} is a zero matrix that allows all tokens in the sequence to attend to each other. Owing to the bidirectionality nature, this type of model is more suitable for natural language understanding tasks. In the case of natural language generation, future tokens would be exposed to bidirectional language models, making them incapable of predicting these tokens. As a comparison, left-to-right unidirectional language models, e.g., GPT [44], are particularly designed for natural language generation. Specifically, in these models, the lower triangular part of \mathbf{M} is set to 0 and the remaining part $-\infty$, so as to allow each token to attend to past tokens (including itself), but prevent it from attending to future tokens. A graphical comparison between the two types of attention masking mechanism is shown in Fig. 2.

With the two types of masking mechanism, there are also two corresponding pre-training objectives: cloze task, which is formally termed Masked Language Model (MLM) [16], for bidirectional

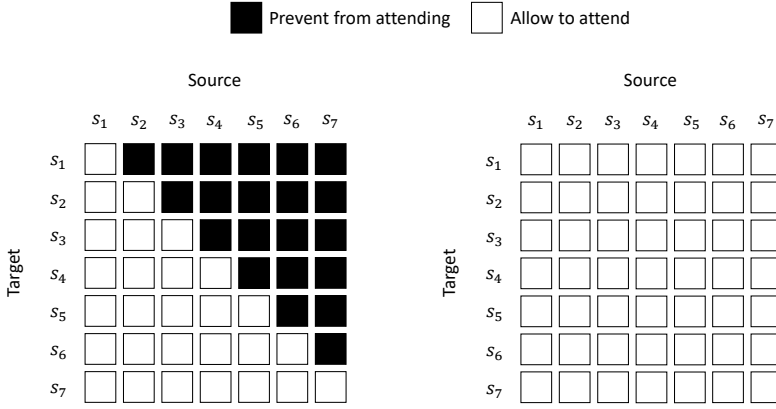


Fig. 2. A comparison between left-to-right unidirectional attention masking (left) and bidirectional attention masking (right).

language models, and auto-regressive generation for unidirectional language models. Because our explanation generation task is closely related to the latter, we describe it in more details. Specifically, given the output vectors $S_n = [s_{n,1}, \dots, s_{n,|S|}]$ resulting from the last layer of Transformer, we pass them through a linear layer to obtain the probability distribution over all tokens in the dataset. With the token probability distribution, we then make the next-token prediction based on preceding tokens, which can be achieved by minimizing the following negative log-likelihood:

$$\mathcal{L} = \sum_t -\log p(s_t | s_{t-k}, \dots, s_{t-1}; \Theta_{LM}) \quad (3)$$

where s_t is the next token to be predicted, k denotes the size of the sliding context window, and Θ_{LM} represents all parameters in Transformer.

The pre-trained language models refer to those Transformers that have a great number of parameters (e.g., 1 billion) and were trained on a large volume of textual data (e.g., 100GB). As a consequence, unlike small unpretrained Transformer [29], it is less likely to do customized modifications on them. In the meantime, re-training a large Transformer model would be unaffordable for most researchers who do not possess much computing resources. Fortunately, there is a promising solution called *prompt learning* [37], where different natural language processing tasks are adapted to a pre-trained language model so as to enable direct modeling of text. In this way, the knowledge exhibited in the model could also be made good use of.

Taking sentiment classification as an example, conventionally the prediction made by a model for a sample “I love this book” should be close to 1 (e.g., 0.97), indicating a positive sentiment. In prompt learning, a template such as “X The book is Y” is constructed firstly. Then, the input placeholder X is filled in with a sample, e.g., “I love this book. The book is Y”, which is termed *prompt*. With this, the model can be instructed to make a prediction at the output placeholder Y, e.g., “great” or “boring”. At last, the prediction is mapped onto a sentiment, i.e., 1 or 0. Clearly, there are two major steps that cost human efforts. The first one is to manually design templates for different application scenarios, and to find the one that best fits a target application. The second is the answer mapping stage, where a number of answer words need to be prepared in advance.

³It is an excerpt from a dialogue in a famous French novella *The Little Prince*. The complete dialogue is “Il y a une fleur. Je crois qu'elle m'a apprivoisé”, meaning “There is a flower. I think that she has tamed me”.

Table 2. Prompt learning for typical natural language processing tasks [37]. In the Template column, \underline{X} and \underline{Y} denote Input and Output, respectively. In our explanation generation task, the template words “Explain the recommendation:” are removed.

Task	Input (X)	Template	Output (Y)
Sentiment Classification	I love this book.	\underline{X} The book is \underline{Y}	great boring ...
Text Summarization	The Omicron ...	\underline{X} TL;DR: \underline{Y}	COVID-19 ... Pandemic
Machine Translation	Elle m'a apprivoisé. ³	French: \underline{X} English: \underline{Y}	She tamed ... The flower
Explanation Generation	room location ...	\underline{X} (Explain the recommendation:) \underline{Y}	The room ... The breakfast ...
	X1: user123abc	$\underline{X1}$ $\underline{X2}$ (Explain the recommendation:) \underline{Y}	The location ...
	X2: item456def	(Explain the recommendation:) \underline{Y}	...

But it does not have to be so sophisticated for natural language generation tasks, whose input and output are both text *per se*. For example, the template for text summarization could simply be “ \underline{X} TL;DR: \underline{Y} ”⁴, and that for machine translation “French: \underline{X} English: \underline{Y} ”. In a similar way, we could also define the template for explanation generation as “ \underline{X} Explain the recommendation: \underline{Y} ”. Although intuitively the template words may look useful, it was found that they could not always guide pre-trained language models to perform the specified task (e.g., “summarize the table”) [34]. Moreover, our key focus is to automatically generate explanations for recommendations rather than manually constructing templates. Therefore, we omit these template words, which gives us “ \underline{X} \underline{Y} ” and “ $\underline{X1}$ $\underline{X2}$ \underline{Y} ”. A comparison of prompt learning between the aforementioned tasks is given in Table 2. In the following, we describe our proposed two methods for explainable recommendation: discrete prompt learning and continuous prompt learning.

3.2 Discrete Prompt Learning

Pre-trained language models, such as BERT [16] and GPT-2 [45], were trained on a large amount of words, which are inherently in a different semantic space as ID tokens, but IDs (e.g., user ID) are indispensable in recommender systems. To resolve this issue, a straightforward way is to find some domain-specific words to represent the IDs, such as movie titles and item features (e.g., “bedroom” for hotel recommendation). In this way, a pre-trained model can be prompted to generate recommendation-specific text. In this work, we explore item features for recommendation explanation generation, and denote the proposed approach as PEPLER-D, where “D” stands for “discrete prompt learning”. A graphical illustration of PEPLER-D is shown in Fig. 3.

From the training set, we can obtain all the item features F_u (or F_i) associated with a user u (or an item i). Suppose $F_u = \{\text{gym, bathroom, breakfast}\}$, and $F_i = \{\text{gym, breakfast, subway, Wi-Fi}\}$. For efficiency, we set the discrete prompt to a fixed size (e.g., 4 in this toy example), which is a common strategy in recommender systems. Under this setting, we need to ensure that the discrete prompt contains as many informative item features as possible, so as to allow the pre-trained model to generate high-quality explanations. For each user-item pair (u, i) , the features in

⁴“TL;DR” stands for “Too Long; Did/Do not Read”.

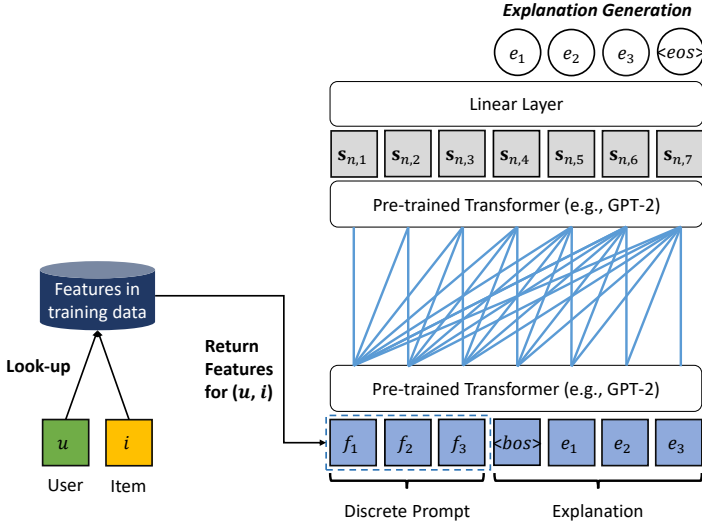


Fig. 3. Our proposed method PEPLER-D that utilizes item features as discrete prompt for explanation generation.

$F_u \cap F_i = \{\text{gym, breakfast}\}$ are more informative because they are related to both user u and item i . However, when $F_u \cap F_i$ is small and does not reach the size of the discrete prompt, we also take the other features in $(F_u \cup F_i) / (F_u \cap F_i) = \{\text{bathroom, subway, Wi-Fi}\}$ into consideration. Though less informative, they are at least associated with either user u or item i . Then, the discrete prompt for the user-item pair is defined as:

$$F_{u,i} = [(F_u \cap F_i), (F_u \cup F_i) / (F_u \cap F_i)] \quad (4)$$

Because the prompt size in the example is fixed to 4, we only use [gym, breakfast, bathroom, subway] in $F_{u,i}$ for explanation generation, and drop the other item features.

During the training stage, the input sequence to the pre-trained model can be represented as $S = [f_1, \dots, f_{|F_{u,i}|}, e_1, \dots, e_{|E_{u,i}|}]$, where $f_1, \dots, f_{|F_{u,i}|}$ are the discrete prompt consisting of features, $e_1, \dots, e_{|E_{u,i}|}$ are the explanation's word sequence, and $|F_{u,i}|$ and $|E_{u,i}|$ denote the number of features and explanation words, respectively. Because all the tokens in sequence S are of the same type, i.e., words, we can perform embedding look-up once for them all, which gives the sequence's token representation $[f_1, \dots, f_{|F_{u,i}|}, e_1, \dots, e_{|E_{u,i}|}]$. The input representation of the sequence to the model is the addition of the token representation, and the positional representation $[p_1, \dots, p_{|S|}]$ that encodes the position of each token in the sequence. We denote the input representation as $S_0 = [s_{0,1}, \dots, s_{0,|S|}]$, where $|S|$ is the length of the sequence.

After passing S_0 through pre-trained Transformer, we obtain the sequence's final representation $S_n = [s_{n,1}, \dots, s_{n,|S|}]$. Then, we apply a linear layer to each token's final representation to map it onto a $|\mathcal{V}|$ -sized vector. As an example, $s_{n,t}$ becomes c_t after passing through this layer:

$$c_t = \text{softmax}(\mathbf{W}^v s_{n,t} + \mathbf{b}^v) \quad (5)$$

where $\mathbf{W}^v \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{b}^v \in \mathbb{R}^{|\mathcal{V}|}$ are weight parameters, and $\text{softmax}(\cdot)$ is the softmax function. The vector c_t represents the probability distribution over the vocabulary \mathcal{V} . For model learning, we adopt negative log-likelihood (NLL) as the loss function, and compute the mean of user-item

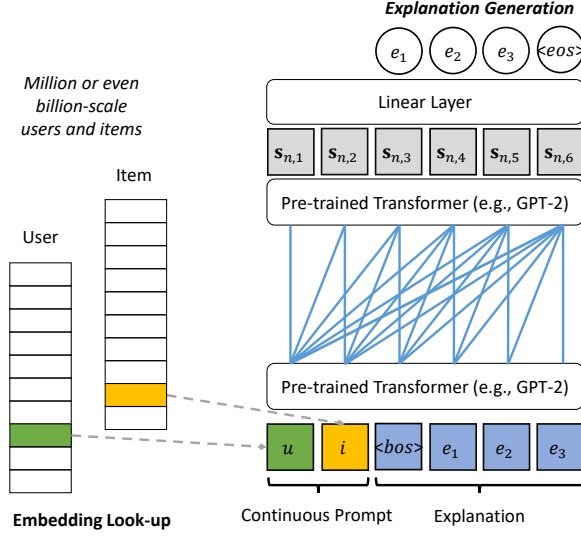


Fig. 4. Our proposed method PEPLER that treats user and item IDs as continuous prompt for explanation generation.

pairs in the training set:

$$\mathcal{L}_D = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} \frac{1}{|E_{u,i}|} \sum_{t=1}^{|E_{u,i}|} -\log c_{|F_{u,i}|+t}^{e_t} \quad (6)$$

where the probability $c_t^{e_t}$ is offset by $|F_{u,i}|$ positions because the explanation is placed at the end of the sequence.

3.3 Continuous Prompt Learning

We have shown that it is feasible to use item features as discrete prompt to a pre-trained model for explanation generation. However, the conversion from IDs to words (i.e., features) may lose some important information of IDs. Taking the identification role of IDs as an example, it is nearly impossible to convert the features back into IDs. Meanwhile, prompts do not necessarily have to be words or even readable. They can be vector representations, either produced by other models or randomly initialized. This type of human-incomprehensible prompts are formally termed *continuous/soft prompt*. Thus, ID vectors could also be directly used as continuous prompts to generate recommendation explanations. Next, we show how to encode the two types of ID u and i into vector representations.

Conceptually, the input sequence can be represented as $S = [u, i, e_1, \dots, e_{|E_{u,i}|}]$, as shown in Fig. 4. Intuitively, one may regard the IDs as special word tokens, and add them to the pre-trained model's vocabulary \mathcal{V} . However, there could be millions or even billions of users and items in recommender systems (e.g., in e-commerce). When generating explanations, predicting a word out of the huge amount of IDs would be time-consuming. Therefore, we do not add the IDs to \mathcal{V} , but instead treat them as two additional types of tokens. Specifically, we prepare two sets of token embeddings: $\mathbf{U} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and $\mathbf{I} \in \mathbb{R}^{|\mathcal{I}| \times d}$, where $|\mathcal{U}|$ and $|\mathcal{I}|$ represent the number of users and items in a dataset, respectively. Then, a user u 's vector representation can be retrieved via:

$$\mathbf{u} = \mathbf{U}^\top g(u) \quad (7)$$

where $g(u) \in \{0, 1\}^{|\mathcal{U}|}$ denotes a one-hot vector, whose non-zero element corresponds to the position that user u 's vector locates in \mathbf{U} . In a similar way, we can obtain i from \mathbf{I} for item i . Notice that, the embeddings \mathbf{U} and \mathbf{I} are randomly initialized, but will be updated by back-propagation during the training process. Then, the sequence's token representation can be denoted as $[\mathbf{u}, \mathbf{i}, \mathbf{e}_1, \dots, \mathbf{e}_{|E_{u,i}|}]$.

The follow-up steps are identical to discrete prompt learning in Section 3.2: perform addition for token representation and positional representation to obtain $\mathbf{S}_0 = [\mathbf{s}_{0,1}, \dots, \mathbf{s}_{0,|S|}]$, pass \mathbf{S}_0 through pre-trained Transformer for producing $\mathbf{S}_n = [\mathbf{s}_{n,1}, \dots, \mathbf{s}_{n,|S|}]$, apply a linear layer with softmax function to each token's final representation $\mathbf{s}_{n,t}$ for next-word prediction, and employ NLL loss function on the word probability distribution \mathbf{c}_t :

$$\mathcal{L}_C = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} \frac{1}{|E_{u,i}|} \sum_{t=1}^{|E_{u,i}|} -\log c_{2+t}^{e_t} \quad (8)$$

where $c_t^{e_t}$ is offset by 2 positions (i.e., user ID and item ID), which is slightly different multiple positions of features in Eq. (6).

3.4 Explanation Generation

During the inference stage, our goal is to instruct the model to generate a word sequence E^* , which has the maximum log-likelihood, as explanation.

$$E^* = \arg \max_{E \in \hat{\mathcal{E}}} \sum_t^{|E|} \log c_{|prompt|+t}^{e_t} \quad (9)$$

where $\hat{\mathcal{E}}$ is the set of all generated word sequences, and $|prompt|$ denotes the prompt's length, i.e., 2 for $[u, i]$ and $|F_{u,i}|$ for $F_{u,i}$.

There are various methods to find the sequence E^* , such as greedy decoding and beam search. Since it is not our key focus to develop searching algorithms, we adopt the simple greedy decoding, which treats the word with the largest probability as the prediction at each step. More precisely, along with the prompt u and i (or $F_{u,i}$), we first feed the model a special begin-of-sequence token $\langle bos \rangle$. From the resulting word probability distribution $\mathbf{c}_{\langle bos \rangle}$, we can select the highest probability word as prediction. Then, we concatenate this predicted word at the end of the sequence to form a new input sequence for generating another word. We do this repeatedly until the model produces a special end-of-sequence token $\langle eos \rangle$, or the generated explanation reaches a pre-defined length.

3.5 Sequential Tuning Strategy

In the case of discrete prompt learning, the prompts are features, which are of the same type as words that pre-trained language models were trained on. As a result, no additional model parameters are introduced, so we can simply optimize Eq. (6) with the following objective function:

$$\mathcal{J} = \min_{\Theta_{LM}} \mathcal{L}_D \quad (10)$$

where Θ_{LM} denotes all the trainable parameters in the pre-trained language model.

However, in the case of continuous prompt learning, we introduced additional prompt parameters, i.e., two sets of embeddings for users and items. Therefore, the model parameters Θ to be updated include pre-trained language model parameters Θ_{LM} and prompt parameters Θ_P . Obviously, the two types of parameters are in different learning stages, since the former are already trained from a large amount of textual data, while the latter are randomly initialized. For example, it is easy to distinguish one word from another with the embeddings from Θ_{LM} , e.g., "hotel" and "room", but it may not be that distinguishable for two users with random embeddings from Θ_P , such as "Tom"

Table 3. Different strategies for tuning pre-trained language models [37]. “Para.” stands for parameters. “N/A” means that there is no prompt, while “None” indicates that the prompts do not have additional parameters.

Strategy	LM Para.	Prompt Para.	Typical Example
Promptless Fine-tuning	Tuned	N/A	BERT [16]
Tuning-free Prompting	Frozen	None	GPT-3 [4]
Fixed-LM Prompt Tuning	Frozen	Tuned	Prefix-Tuning [34]
Fixed-prompt LM Tuning	Tuned	None	Our PEPLER-D
Prompt+LM Fine-tuning	Tuned	Tuned	P-Tuning [39]
Sequential Tuning: Fixed-LM Prompt Tuning → Prompt+LM Fine-tuning	Tuned	Tuned Twice	Our PEPLER

and “Jerry”. Also, previous study [2] shows that randomly initialized parameters could only be updated in a small neighborhood with stochastic gradient descent (SGD). Hence, how to effectively bridge the two types of parameters becomes a critical issue.

To tackle this problem, we propose a sequential tuning strategy. Specifically, we first freeze the language model parameters Θ_{LM} , and optimize the prompt parameters Θ_P with Eq. (8). Once Θ_P has converged, we fine-tune all the model parameters (i.e., Θ_{LM} and Θ_P) with Eq. (8) again. This two-step procedure can be demonstrated with the following formula:

$$\mathcal{J} = \min_{\Theta_P} \mathcal{L}_C \xrightarrow{\text{followed by}} \mathcal{J} = \min_{\Theta=\{\Theta_{LM}, \Theta_P\}} \mathcal{L}_C \quad (11)$$

In fact, our sequential tuning strategy is a combination of two typical tuning strategies [37]: Fixed-LM Prompt Tuning and Prompt+LM Fine-tuning (see Table 3). In section 5.2, we conduct an effect comparison to prove that this strategy is indeed more useful than either of them. We omit the other three strategies, i.e., Promptless Fine-tuning, Tuning-free Prompting and Fixed-prompt LM Tuning. The first is usually used in pre-training plus fine-tuning paradigm, and the second is particularly suitable for zero-shot learning scenario, so they are not applicable to our methods. The last one is adopted in our PEPLER-D.

3.6 Recommendation as Regularization

To bridge the aforementioned gap between pre-trained language models and continuous prompts, we come up with another approach: regularizing the learning of explanation generation via an additional rating prediction task (see Fig. 5). The intuition behind this idea is that each rating score $r_{u,i}$ was assigned by a user u to an item i , so it to some extent captures the relation between this user-item pair. Hence, the ratings could be used to better learn the continuous prompts. Moreover, recent studies find out that the two task of recommendation and an additional task (such as feature ranking [11], explanation ranking [30] and review generation [49]) could help the learning of each other. Inspired by this, we propose to leverage recommendation task to help the learning of explanation generation. Since there is a great number of off-the-shelf recommendation models and our key focus is on explanation generation, we adopt and test two typical recommendation models: Matrix Factorization (MF) [41] and Multi-Layer Perceptron (MLP) [32].

Specifically, for MF the rating score $\hat{r}_{u,i}$ is resulted from the dot product of the target user and item’s representations \mathbf{u} and \mathbf{i} :

$$\hat{r}_{u,i} = \mathbf{u}^\top \mathbf{i} \quad (12)$$

Because the two types of representations are already available, this operation does not introduce additional model parameters. In the case of MLP with z hidden layers, the rating score is computed

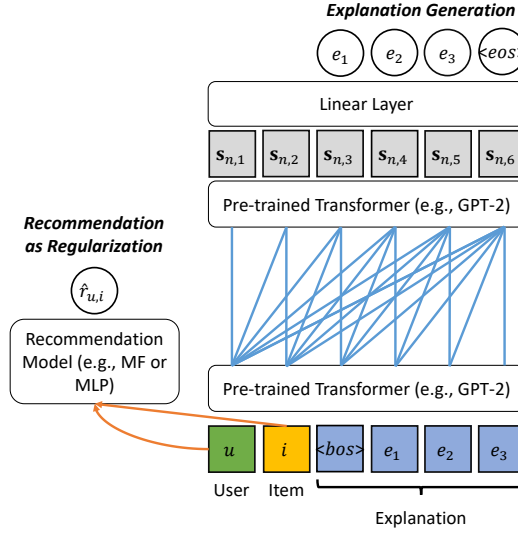


Fig. 5. Our proposed method PEPLER that regards the rating prediction task as a type of regularization for better learning of the explanation generation task.

as follows:

$$\begin{cases} \mathbf{a}_0 = \sigma(\mathbf{W}_0[\mathbf{u}, \mathbf{i}] + \mathbf{b}_0) \\ \mathbf{a}_1 = \sigma(\mathbf{W}_1\mathbf{a}_0 + \mathbf{b}_1) \\ \dots\dots\dots \\ \mathbf{a}_z = \sigma(\mathbf{W}_z\mathbf{a}_{z-1} + \mathbf{b}_z) \end{cases} \quad \text{and } \hat{r}_{u,i} = \mathbf{w}^\top \mathbf{a}_z + b \quad (13)$$

where $\mathbf{W}_0 \in \mathbb{R}^{d_h \times 2d}$, $\mathbf{b}_0 \in \mathbb{R}^{d_h}$, $\mathbf{W}_* \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{b}_* \in \mathbb{R}^{d_h}$, $\mathbf{w} \in \mathbb{R}^{d_h}$, $b \in \mathbb{R}$ are additional parameters for the recommendation task, and $\sigma(\cdot)$ denotes the sigmoid function. For both MF and MLP, mean square error is adopted as the loss function:

$$\mathcal{L}_R = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} (r_{u,i} - \hat{r}_{u,i})^2 \quad (14)$$

where $r_{u,i}$ is the ground-truth rating that user u assigned to item i .

Then, the two tasks can be integrated into a multi-task learning framework with the following objective function:

$$\mathcal{J} = \min_{\Theta = \{\Theta_{LM}, \Theta_P, \Theta_{REC}\}} (\mathcal{L}_C + \lambda \mathcal{L}_R) \quad (15)$$

where the model parameters Θ consist of pre-trained language model parameters Θ_{LM} , continuous prompt parameters Θ_P (i.e., user and item representations) and recommendation model parameters Θ_{REC} (\emptyset for MF). Since the recommendation task is used as a regularization term, we can adjust the regularization coefficient λ to control the learning of the explanation generation task.

4 EXPERIMENTAL SETUP

4.1 Datasets

For experimentation, we adopt three publicly available explainable recommendation datasets, and their data splits [27]. During the splitting process, each dataset is randomly divided into training, validation and testing sets with ratio 8:1:1 for 5 times, and the training set holds at least one record

Table 4. Statistics of the three datasets.

	TripAdvisor	Amazon	Yelp
#users	9,765	7,506	27,147
#items	6,280	7,360	20,266
#records	320,023	441,783	1,293,247
#features	5,069	5,399	7,340
#records / user	32.77	58.86	47.64
#records / item	50.96	60.02	63.81
#words / explanation	13.01	14.14	12.32

for each user and each item. The three datasets are from TripAdvisor⁵ (hotel), Amazon⁶ (movies & TV) and Yelp⁷ (restaurant), respectively. Each record in the datasets is comprised of a user ID, an item ID, a rating in the scale of 1 to 5, an explanation and an item feature. The explanations are sentences extracted from user reviews. Each explanation contains at least one item feature, such as “bedroom” and “breakfast”, which ensures the explanation quality. Statistics of the datasets are shown in Table 4. We can see that Yelp is much larger than the other two in terms of size, making it closer to the real-world situation where there are millions of users and items.

4.2 Evaluation Metrics

To evaluate explanation performance, we measure the generated explanations from two main perspectives: text quality and explainability. For the former, we adopt **BLEU** [43] in machine translation and **ROUGE** [36] in text summarization, and report BLEU-1 and BLEU-4, and Precision, Recall and F1 of ROUGE-1 and ROUGE-2. Notice that, BLEU is a precision-oriented metric, while ROUGE is a recall-oriented metric. Though being widely used, BLUE and ROUGE are not flawless. For example, it is difficult for them to detect the problem of identical sentences, i.e., many explanations for different user-item pairs are exactly the same for some methods, as shown in our experiments. Treating these identical sentences as explanations is less appropriate, because they are less likely to well explain the special property of different recommendations. To quantitatively measure this, we adopt **USR** that computes the Unique Sentence Ratio of generated explanations [27]:

$$USR = \frac{|\mathcal{E}|}{N} \quad (16)$$

where \mathcal{E} represents the set of unique sentences generated by a model, and N is the total number of testing samples. Note that, \mathcal{E} only holds one of the exactly matched explanations.

Moreover, text quality is not equal to explainability. In the case of explainable recommendation, users may value more an explanation that justifies a recommendation’s advantage on certain item features [6, 27]. To this end, we adopt the other three metrics proposed in [27]: Feature Matching Ratio (**FMR**), Feature Coverage Ratio (**FCR**) and Feature Diversity (**DIV**).

FMR measures whether a generated explanation contains the feature in the ground-truth text. Formally, it is defined as follows:

$$FMR = \frac{1}{N} \sum_{u,i} \delta(f_{u,i} \in \hat{E}_{u,i}) \quad (17)$$

⁵<https://www.tripadvisor.com>

⁶<http://jmcauley.ucsd.edu/data/amazon>

⁷<https://www.yelp.com/dataset/challenge>

where $\hat{E}_{u,i}$ is the generated explanation for the user-item pair, $f_{u,i}$ is the feature in the ground-truth, and $\delta(x) = 1$ when x is true, or $\delta(x) = 0$ otherwise.

FCR is computed as the number of distinct features contained in all the generated explanations, divided by the total number of features in the whole dataset:

$$FCR = \frac{N_g}{|\mathcal{F}|} \quad (18)$$

where \mathcal{F} is the collection of unique features in ground-truth explanations, and N_g denotes the amount of distinct features appeared in the generated explanations.

DIV measures the diversity of features between all generated explanations. The intuition is that explanations are expected to discuss different features in accordance with the given user-item pairs. Hence, it computes the intersection of features between any two generated explanations:

$$DIV = \frac{2}{N \times (N - 1)} \sum_{u,u',i,i'} |\hat{\mathcal{F}}_{u,i} \cap \hat{\mathcal{F}}_{u',i'}| \quad (19)$$

where $\hat{\mathcal{F}}_{u,i}$ and $\hat{\mathcal{F}}_{u',i'}$ represent two feature sets contained in two generated explanations, and $|\cdot|$ denotes the number of features in the resulting set.

For DIV, the lower, the better, while it is opposite for the rest of metrics.

4.3 Compared Methods

We introduce four state-of-the-art baselines, which are based on representative language models, including BERT [16], Transformer [55], GRU [15] and LSTM [23], respectively. For these baselines, their whole model parameters are trained all together. We divide them into two groups, depending on whether IDs are directly used or not.

We first compare our PEPLER-D with the following method, because both of them do not directly make use of IDs but instead map IDs onto item features.

- **Aspect Conditional Masked Language Model (ACMLM)** [42] is a fine-tuned BERT [16], where an attention layer is introduced to encode the features for both the user and the item. By predicting masked tokens, this model can produce diverse sentences.

Then, we make comparison with the following three methods for our PEPLER, since they all leverage only user and item IDs to generate explanations.

- **Neural Rating and Tips generation (NRT)** [32] can predict a rating and generate a tip simultaneously based on user and item IDs. The generation component is a GRU [15]. We take the explanations in the datasets as tips. Moreover, we find that the model's problem of generating identical sentences (as reported in [27]) is caused by the L2 regularization in its original design. For fair comparison, we removed it.
- **Attribute-to-Sequence (Att2Seq)** [17] is a review generation approach with a two-layer LSTM [23]. We take the explanations as reviews. This model has an attention module, but we find that it makes the generated content unreadable. To be fair, we removed it as well.
- **Personalized Transformer for Explainable Recommendation (PETER)** [29] is a small unpretrained Transformer [55] particularly designed for explanation generation. To bridge the gap between IDs and words, an additional task named "context prediction" is introduced. This model can also make recommendations.

We conducted a user survey in NETE [26, 27] and showed that the explanations generated by NETE were perceived useful by participants. Moreover, the explanation quality of PETER [29] is much better than that of NETE on the same automatic evaluation metrics. Hence, as long as the explanations produced by our new approach in this work are of even better quality than PETER on

the same evaluation metrics, they shall be useful to real users as well. This is evidenced by [57] that users' perception towards machine-generated explanations are highly correlated with the factors of relevance, repetition and feature appearance, which correspond to BLEU/ROUGE, USR and FMR in this work.

4.4 Implementation Details

We train each model on the training set, tune the hyper-parameters on the validation set, and report the performance on the testing set. The results are averaged on the 5 data splits. We adopt the code of ACMLM, and implement the other baselines (i.e., NRT, Att2Seq and PETER) by ourselves. For our models PEPLER and PEPLER-D, we implement them in Python⁸ with PyTorch⁹, and load pre-trained GPT-2 [45] from huggingface¹⁰ as their backbone. GPT-2 uses Byte Pair Encoding (BPE) [46] for vocabulary construction. This technique could effectively mitigate Out-Of-Vocabulary (OOV) problem by encoding rare words into multiple sub-word units. For example, the word "restaurant" is encoded into three sub-words "rest", "aur" and "ant", while the word "room" is still "room". In total, there are 50,257 BPE tokens in GPT-2. For fair comparison, we apply BPE to all the models, and set the length of explanations to 20 BPE tokens. For our model PEPLER-D, the number of input features is also set to 20 BPE tokens. We reuse the other default settings of the baselines.

The size of embeddings/representations d in GPT-2 is 768. We optimize our models PEPLER and PEPLER-D with AdamW [40], and set batch size to 128. The learning rate is set to 0.001 for PEPLER, and 0.0001 for PEPLER-D. At each epoch, we save the model if it achieves the lowest loss on the validation set. When the loss does not decrease for 5 times, we stop training and load the saved model for prediction. In the case of recommendation as regularization in PEPLER, the number of hidden layers z in MLP is set to 2, and the dimension of hidden layers d_h 400. We search the regularization coefficient λ from $[10^{-5}, 10^{-4}, \dots, 10^3]$.

5 RESULTS AND ANALYSIS

In this section, we first quantitatively compare the performance of different explanation methods with automatic metrics. We then further study the effect of our proposed two training strategies. Next, we qualitatively examine two explanation samples as generated by all the methods. After that, we visualize our method's attention weights to demonstrate that IDs can indeed be fused into the pre-trained model. At last, we study the effect of model size on explanation generation performance.

5.1 Quantitative Analysis on Explanations

The performance comparison between different explanation generation methods is shown in Table 5. These methods are divided into two groups. We first examine those that map IDs onto item features, i.e., ACMLM and PEPLER-D. Our PEPLER-D consistently and significantly outperforms ACMLM on the three datasets in terms of text quality measured by BLEU and ROUGE. This demonstrates its effectiveness in generating high-quality sentences that are semantically close to the ground-truth text. Also, we notice that the performance gap between our PEPLER-D and ACMLM (a fine-tuned BERT) is extremely large, because the latter's generation is achieved by predicting masked tokens, which is quite different from conventional auto-regressive generation. This may explain why ACMLM produces diverse sentences (high USR) and features (low DIV). However, they could be

⁸<https://www.python.org>

⁹<https://pytorch.org>

¹⁰<https://huggingface.co/gpt2>

Table 5. Performance comparison of explanation generation methods in terms of Explainability and Text Quality on three datasets. The methods are divided into two groups according to whether IDs are directly used or not. PEPLER employs the default sequential tuning strategy, while the other two variants use recommendation as regularization with MLP and MF, respectively. B1 and B4 stand for BLEU-1 and BLEU-4. R1-P, R1-R, R1-F, R2-P, R2-R and R2-F denote Precision, Recall and F1 of ROUGE-1 and ROUGE-2. BLEU and ROUGE are percentage values (% symbol omitted for table clarity), while the others are absolute values. The best performing values are boldfaced, and ** and * indicate the statistical significance over the best baseline for $p < 0.01$ and $p < 0.05$ via Student's t-test, respectively.

	Explainability				Text Quality							
	FMR↑	FCR↑	DIV↓	USR↑	B1↑	B4↑	R1-P↑	R1-R↑	R1-F↑	R2-P↑	R2-R↑	R2-F↑
Yelp												
ACMLM	0.05	0.31	0.95	0.95	7.01	0.24	7.89	7.54	6.82	0.44	0.48	0.39
PEPLER-D	0.05	0.24	1.53	0.13	9.17**	0.40**	15.67**	10.47**	11.73**	1.09**	0.78**	0.83**
NRT	0.06	0.12	1.67	0.20	10.92	0.60	16.73	11.91	12.89	1.63	1.21	1.26
Att2Seq	0.05	0.05	2.25	0.05	10.25	0.54	17.13	11.44	12.72	1.49	1.13	1.16
PETER	0.08	0.15	1.62	0.15	10.74	0.63	16.18	11.90	12.63	1.60	1.32	1.28
PEPLER	0.08	0.30**	1.52	0.35**	11.23	0.73	17.51	12.55	13.53	1.86*	1.42	1.46
PEPLER (MLP)	0.08	0.24	1.58	0.25	10.95	0.68	17.52	12.31	13.36	1.83	1.34	1.40
PEPLER (MF)	0.08	0.27	1.66	0.30	11.70	0.75**	17.52	12.85**	13.72**	1.86*	1.45*	1.48**
Amazon												
ACMLM	0.10	0.31	2.07	0.96	9.52	0.22	11.65	10.39	9.69	0.71	0.81	0.64
PEPLER-D	0.08	0.19	1.85*	0.15	10.94**	0.49**	16.31**	11.80**	12.80**	1.43**	1.13**	1.16**
NRT	0.10	0.04	2.71	0.09	12.06	0.69	17.17	13.15	13.83	1.94	1.68	1.64
Att2Seq	0.09	0.04	2.64	0.05	12.07	0.73	18.35	12.86	14.14	2.01	1.56	1.61
PETER	0.09	0.09	2.16	0.20	11.75	0.89	16.51	13.10	13.55	1.96	1.76	1.68
PEPLER	0.11	0.27	2.06	0.38	13.19	1.05	18.51	14.16	14.87	2.36*	1.88	1.91
PEPLER (MLP)	0.11	0.34**	2.10	0.48**	13.59**	1.08**	17.94	14.50*	14.82	2.29	1.96*	1.93**
PEPLER (MF)	0.12*	0.24	2.18	0.35	13.46	1.02	18.30	14.37	14.92*	2.29	1.92	1.90
TripAdvisor												
ACMLM	0.07	0.41	0.78	0.94	3.45	0.02	4.86	3.82	3.72	0.18	0.20	0.16
PEPLER-D	0.05	0.22	2.69	0.08	14.61**	0.87**	18.07**	14.83**	15.32**	1.76**	1.66**	1.58**
NRT	0.05	0.02	6.07	0.00	13.76	0.80	19.01	14.57	15.58	2.10	1.59	1.68
Att2Seq	0.06	0.05	4.74	0.02	15.20	0.96	18.74	16.42	16.38	2.42	2.32	2.19
PETER	0.07	0.09	3.62	0.05	15.13	1.00	18.30	16.15	16.00	2.24	2.23	2.06
PEPLER	0.07	0.21**	2.71**	0.24**	15.49	1.09	19.48	15.67	16.24	2.48	2.21	2.16
PEPLER (MLP)	0.07	0.10	3.33	0.08	15.70	1.04	18.87	16.21	16.24	2.35	2.26	2.12
PEPLER (MF)	0.07	0.21**	2.89	0.21	16.02	1.15*	19.82	16.31	16.69	2.53	2.32	2.22

less useful to real users and might even hurt user experience, since their text quality cannot be guaranteed (see the generated examples in Table 6).

Next, we analyze the results of models that directly leverage user and item IDs for explanation generation, i.e., NRT, Att2Seq, PETER and PEPLER. As we can see, the text quality of these methods are largely improved compared with those that convert IDs into item features (i.e., ACMLM and PEPLER-D), because the conversion process may lose certain information of IDs, e.g., identification. Among the four ID-based methods, NRT and Att2Seq generally achieve the same performance on all metrics, but neither of them are as comparable as PETER and PEPLER. Because NRT and Att2Seq are based on recurrent neural networks (i.e., GRU or LSTM), they may suffer from the notorious long-term dependency problem, and thus their sequence modeling capability could be impaired. As a comparison, PETER and PEPLER do not have such an issue, since in Transformer future tokens at any time step are given access to all the past tokens. Moreover, given the fact that PETER is a small unpretrained Transformer, it does not outperform PEPLER that is pre-trained on

large textual corpora and hence possesses rich linguistic knowledge. In the meantime, it proves the rationale of our continuous prompt learning approach that could effectively make use of such knowledge for generating better explanations.

We then make a comparison for our proposed two training strategies. The default PEPLER employs sequential tuning, while the other two variants utilize recommendation as regularization with MLP and MF, respectively, and therefore are denoted as PEPLER (MLP) and PEPLER (MF). Compared with PEPLER, PEPLER (MF) greatly improves the text quality most of the time. In the meantime, PEPLER (MLP) maintains comparable text quality to PEPLER, but often cannot keep up explainability, e.g., the decrease on FCR and USR. This can be explained by the difference between MF and MLP in terms of additional parameters for recommendation task. For MF, the prediction is simply made by the dot product between user and item embeddings, in which case no additional parameters are involved. In contrast, MLP must go through a stack of hidden layers that consist of many parameters, which might help to predict ratings but adversely affect the learning of the explanation task. Since the recommendation task requires extra rating data for training, which may not always be available in other natural language generation tasks (e.g., dialogue systems), we set sequential tuning as the default training strategy for PEPLER. Depending on the specific application, one may consider PEPLER (MF).

From the experimental results, we also observe two special cases on the TripAdvisor dataset, where Att2Seq obtains the largest ROUGE scores. The reasons are as follows. First, we fixed its generation issue (see the discussion in Section 4.3), which makes it a competitive baseline. Second, the dataset is quite small and thus the training samples are limited, so our large model may underfit. This is not a problem in real-world applications where there are abundant training samples (e.g., in e-commerce), since our model already outperformed state-of-the-art baselines on the largest dataset Yelp, which contains approximately 1.3 million samples.

5.2 Effect of Sequential Tuning

To validate the superiority of our proposed Sequential Tuning strategy, we compare it with its two composite training strategies: Fixed-LM Prompt Tuning and Prompt+LM Fine-tuning [37]. The results of Sequential Tuning (utilized in the default PEPLER) on the three datasets are presented in Table 5. Given the consistent performance across different metrics, in Fig. 6 we only show BLEU-4 with varied learning rates on three datasets.

As it can be seen, the highest BLEU-4 score is achieved by our Sequential Tuning strategy (purple), when the learning rate is set to 10^{-3} . This manifests its advantage in bridging the gap between the randomly initialized continuous prompts and the pre-trained language model. In particular, the pattern of our Sequential Tuning and that of Prompt+LM Fine-tuning (green) is quite similar, because they both tune all the model parameters, including both prompts and the pre-trained model. Obviously, the curve of our Sequential Tuning is on the top of that of Prompt+LM Fine-tuning. The difference is that the former's prompts are already trained, which could help to reduce the gap between prompts and the pre-trained model. This supports the rationale of our two-staged Sequential Tuning strategy. Moreover, when the learning rate is large (i.e., 10^{-2}), the performance of both strategies goes down dramatically, nearly reaching 0, because large learning rates lead to significant changes of parameters in the pre-trained model. Hence, smaller learning rates are more appreciated to fine-tuning. In contrast, the performance of Fixed-LM Prompt Tuning (brown) is relatively stable, regardless of the changing learning rates. However, it does not outperform the other two strategies, because the model is frozen and only prompts can be tuned, and therefore could not be well adjusted to the target explanation task.

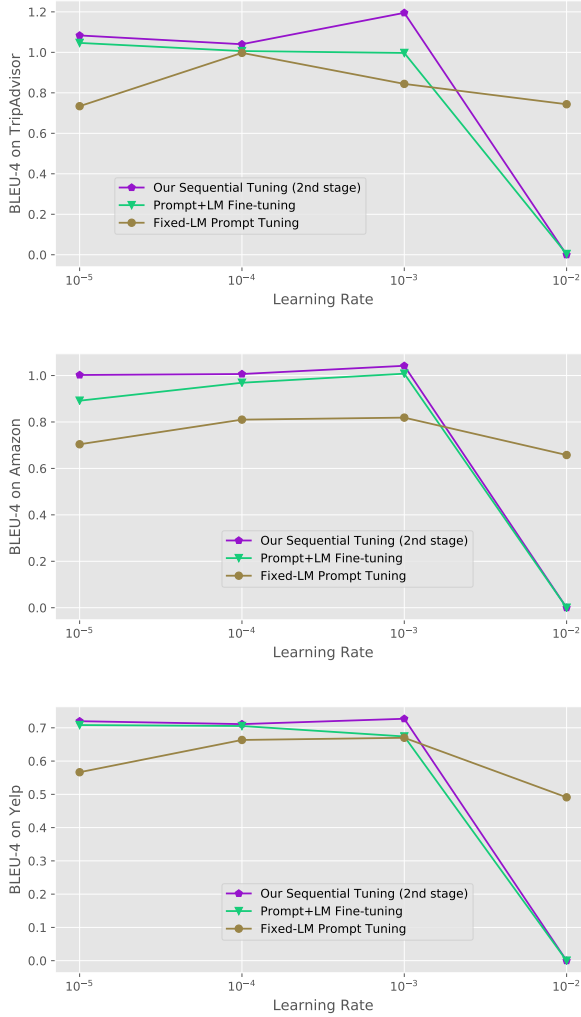


Fig. 6. A comparison of three tuning strategies for continuous prompt learning in terms of BLEU-4 with varying learning rates on three datasets.

5.3 Effect of Recommendation as Regularization

In this work, we propose two training strategies to bridge continuous prompts and the pre-trained model, including sequential tuning and recommendation as regularization. We analyze the latter in more details, because the former is already presented in the previous subsection.

In Fig. 7, we investigate how PEPLER (MF) and PEPLER (MLP) react to varying λ , the regularization coefficient on the recommendation task. For better comparison, PETER is included since it is the previous state-of-the-art, and can also perform recommendation. The accuracy of this task is measured by root mean square error (RMSE), and a lower score indicates a better performance. By comparing the first two sub-figures, we can clearly see that there is a trade-off between explanation text quality (evaluated by BLEU-4) and recommendation accuracy (measured by RMSE)

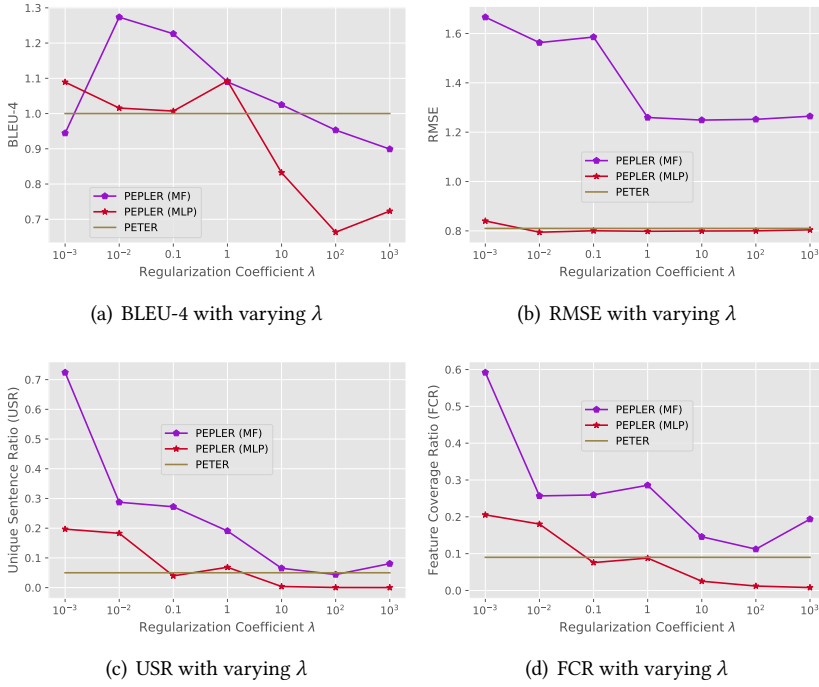


Fig. 7. The effect of regularization coefficient λ on the recommendation task with MF or MLP for PEPLER on the TripAdvisor dataset. For better comparison, the results of PETER are shown.

for PEPLER (MF). For example, when $\lambda = 10^{-2}$, its explanation performance reaches an optimal, but its recommendation performance is greatly deteriorated. It actually supports our design of this training strategy that leverages the recommendation task to help the learning of explanation generation. As a comparison, PEPLER (MLP) is not so sensitive to varying λ . We also notice that there is a huge gap between PEPLER (MF) and PEPLER (MLP) in terms of recommendation accuracy. Owing to the linearity of MF, its representation ability could be largely limited [22], and thus could not accurately estimate the ratings. But because of the simple dot product operation, the relation between users and items encoded in ratings could in turn be easily propagated to better learn the explanation task, i.e., higher BLEU-4 for PEPLER (MF). Since the purpose of PEPLER (MF) is not to make recommendations, when deploying it for real-world applications, one can use the predictions from another effective recommendation model, e.g., neural matrix factorization [22].

The last two sub-figures show a decline of explainability as measured by Unique Sentence Ratio (USR) and Feature Coverage Ratio (FCR) for both PEPLER (MF) and PEPLER (MLP), with the increase of λ . It suggests that a smaller λ could lead to larger USR and FCR. However, this pattern does not match that of text quality as measured by BLEU-4. When text quality cannot be guaranteed, the explanations could be unreadable to users and thus may affect their experience. In such cases, large explainability scores would be pointless. Therefore, we give priority to text quality when tuning λ for both PEPLER (MF) and PEPLER (MLP).

Table 6. Explanations on two different cases as generated by different methods on the TripAdvisor dataset. Special tokens used to perform generation (i.e., *<bos>* and *<eos>*) are removed for the ease of readability. The boldfaced words in the ground-truth are the key features. Matched features in the generated explanations are also boldfaced.

Ground-truth	the swimming pool is fantastic
ACMLM	swimming pool swimming pools pool strip beach area
NRT	the hotel is located in a great location
Att2Seq	the hotel is located in the heart of the city and the main shopping area is also within walking distance
PETER	the hotel is located in the heart of the city and the harbour
PEPLER-D	the room was very nice and the bed was very comfortable
PEPLER	the pool is amazing and the pool is very relaxing
Ground-truth	this is one of the finest hotels in all of Europe
ACMLM	swimming pool area pool ja ##cu ##zzi pool city area gym building pool area spa gym pool area
NRT	the hotel is located in a great location
Att2Seq	the hotel is located in the heart of the city and the main shopping area is also within walking distance
PETER	the hotel is in a great location
PEPLER-D	the hotel is a short walk from the old town
PEPLER	the hotel is located in the heart of the city and is very well maintained

5.4 Qualitative Case Study on Explanations

In Table 6, we present two examples generated by all the methods for hotel recommendations on the TripAdvisor dataset. In the first case, the ground-truth explanation gives a positive comment about the hotel’s swimming “pool”. Only two methods, i.e., ACMLM and our PEPLER, successfully capture this key feature. However, ACMLM’s explanation is not even readable, because it is just a bunch of unordered random words. These meaningless explanations are not very likely to be useful to real users. As a comparison, the explanations generated by the other approaches are all readable and fluent. This actually echoes their performances on BLEU and ROUGE, which emphasize more text quality and readability. But BLEU and ROUGE are not perfect, because they fail to detect the problem of identical explanations (see the same sentences generated by NRT or Att2Seq for two different cases). This is why we also adopt some explainability metrics [27] that particularly care about item features and sentence diversity. Moreover, Att2Seq tends to generate long explanations, which may explain why it obtains good performance regarding ROUGE on the TripAdvisor dataset (see Table 5), because ROUGE is a recall-oriented metric and favors long sentences. The explanations generated by the other three approaches, i.e., PETER, PEPLER-D and PEPLER, are quite good, because they all adopt the Transformer model, which has strong language modeling capability. Despite of that, the explanations from our PEPLER are semantically closer to the ground-truth. Taking the second case as an example, the ground-truth explanation evaluates the overall quality of the hotel (“one of the finest hotels”), but PETER and PEPLER-D respectively talks about location (“great location”) and distance (“short walk”), while our PEPLER comments about not only the hotel’s location (“located in the heart of city”) but also its quality (“well maintained”). We attribute this to the effectiveness of our proposed continuous prompt learning and the sequential tuning strategy. Moreover, we see that the expression of PEPLER’s explanations is quite rich, which

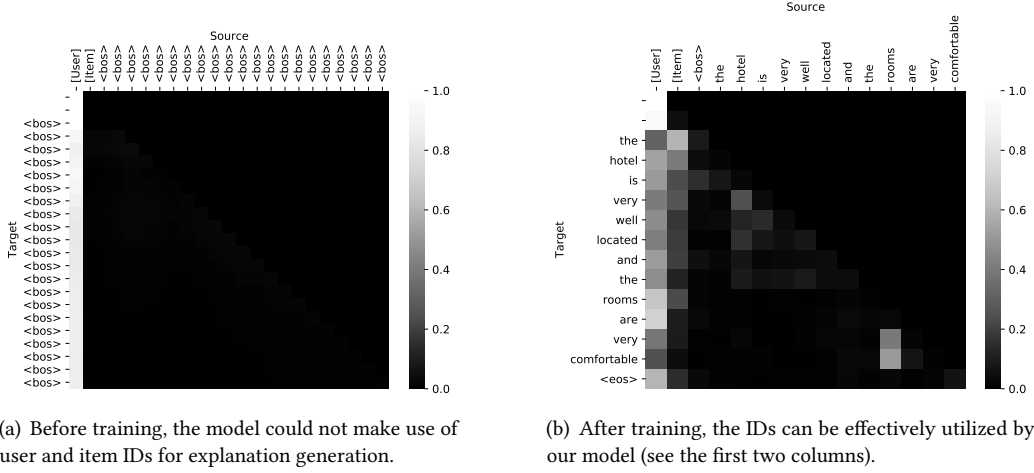


Fig. 8. Visualization of our PEPLER model's last attention layer, before and after training. The larger the attention weights, the lighter the cells.

could be brought by the linguistic knowledge contained in the pre-trained model, as it is already trained on large text corpora.

5.5 Attention Visualization

In our continuous prompt learning approach, we directly incorporate user and item IDs into the pre-trained model for natural language explanation generation for recommendations. To see whether the IDs are really fused into the model, we visualize its last attention layer before and after training in Fig. 8. In both sub-figures, the larger an attention weight, the lighter the corresponding cell. Before training, the ID representations are randomly initialized, but the model is already trained on large textual corpora. This semantic gap makes the pre-trained model difficult to perform natural language generation based on IDs. From Fig. 8 (a), we can see that the model cannot utilize both user and item IDs before training, resulting in an unreadable sequence of multiple *<bos>*. But after training, the model is able to make use of the IDs and thus can generate a fluent and readable explanation, e.g., “the hotel is very well located and the rooms are very comfortable”. It confirms that the IDs can indeed be well fused into the model. We attribute this to the effectiveness of our proposed sequential tuning approach.

5.6 Effect of Model Size

The pre-trained GPT-2 model [45] has four varying sizes, including Small, Medium, Large and XL. This work is based on the default 12-layered small model, while the others have 24, 36, and 48 layers, respectively. Here, we investigate whether larger models with more attention layers could lead to better explanation generation performance. In Fig. 9, we present their text quality as measured by BLEU-4 on the three datasets, where the XL model is omitted because it is too large and ran out of memory in our every experimental trial. From the three sub-figures, we do not observe an increasing trend with the increase of model size, and therefore cannot certify that a larger model always leads to a better performance. We conjecture that large models might suffer from data-hungry problem and therefore may need more data to perform well. Nevertheless, the small model consistently reaches a reasonably good performance on three datasets, while it has

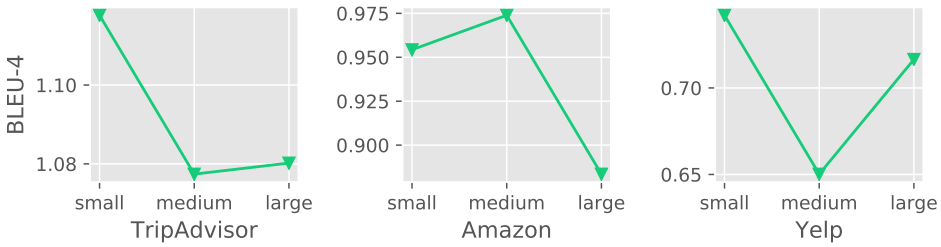


Fig. 9. The effect of model size on text quality in terms of BLEU-4 on three datasets.

less model parameters and thus takes less time to fine-tune. It actually supports our choice of the default model.

6 CONCLUSION

In this work, we propose two prompt learning approaches to exploit the rich knowledge contained in pre-trained language models for recommendation explanation generation. To bridge the gap between continuous prompts and pre-trained models, we come up with two effective learning strategies. Extensive experiments demonstrate the effectiveness of our approaches in generating high-quality explanations as measured by text quality and explainability metrics.

As future works, we are immensely interested in whether the generated explanations possess bias or stereotype against certain groups of users and how to mitigate them, as reported in recent studies [35, 47], pre-trained models may exhibit societal bias towards different demographics. Moreover, since the biased generation was triggered by discrete prompts [47], we wonder whether it is possible to design some other discrete prompts that can help us diagnose the behavior of pre-trained models, which would certainly increase their interpretability. Besides explanation generation for recommender systems, we also plan to adopt our approaches to other applications of personalized natural language generation, such as personalized question answering systems and personalized conversational agents. Moreover, it would also be interesting to incorporate item images into pre-trained models to generate visual explanations for recommendations, since “a picture is worth a thousand words”. Another meaningful extension is to adapt pre-trained models to cross-lingual explanation generation, since international platforms, e.g., Amazon, may serve users who speak different languages.

ACKNOWLEDGMENTS

This work was supported by Hong Kong RGC GRF project (RGC/HKBU12201620), Hong Kong Baptist University IG-FNRA project (RC-FNRA-IG/21-22/SCI/01), and partially supported by NSF IIS-1910154, 2007907, and 2046457. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

- [1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.
- [2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*. PMLR, 242–252.
- [3] Eyal Ben-David, Nadav Oved, and Roi Reichart. 2022. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of the Association for Computational Linguistics* 10 (04 2022), 414–433.

- [4] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*.
- [5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*. 1583–1592.
- [6] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate natural language explanations for recommendation. In *Proceedings of SIGIR'19 Workshop on Explainable Recommendation and Search*. ACM.
- [7] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *Proceedings of The Web Conference 2021*.
- [8] Li Chen and Feng Wang. 2017. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. 17–28.
- [9] Li Chen, Dongning Yan, and Feng Wang. 2019. User evaluations on sentiment-based recommendation explanations. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 9, 4 (2019), 1–38.
- [10] 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*. 765–774.
- [11] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 305–314.
- [12] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic explainable recommendation based on neural attentive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 53–60.
- [13] 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*.
- [14] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-Attentive Multi-Task Learning for Explainable Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2137–2143.
- [15] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1724–1734.
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.
- [17] 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*. 623–632.
- [18] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*. 13063–13075.
- [19] Zuhui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-Aware Explainable Recommendation over Knowledge Graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [20] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should I explain? A comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies* 72, 4 (2014), 367–382.
- [21] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. 1661–1670.
- [22] 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*. International World Wide Web Conferences Steering Committee, 173–182.
- [23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [24] Junjie Li, Haoran Li, and Chengqing Zong. 2019. Towards personalized review summarization via user-aware sequence network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6690–6697.
- [25] Lei Li, Li Chen, and Ruihai Dong. 2021. CAESAR: context-aware explanation based on supervised attention for service recommendations. *Journal of Intelligent Information Systems* 57 (2021), 147–170. Issue 1.

- [26] Lei Li, Li Chen, and Yongfeng Zhang. 2020. Towards Controllable Explanation Generation for Recommender Systems via Neural Template. In *Companion Proceedings of the Web Conference 2020*. 198–202.
- [27] 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*. 755–764.
- [28] Lei Li, Yongfeng Zhang, and Li Chen. 2021. EXTRA: Explanation Ranking Datasets for Explainable Recommendation. In *Proceedings of the 44th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2463–2469.
- [29] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 4947–4957.
- [30] Lei Li, Yongfeng Zhang, and Li Chen. 2022. On the Relationship between Explanation and Recommendation: Learning to Rank Explanations for Improved Performance. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2022).
- [31] Piji Li, Zihao Wang, Lidong Bing, and Wai Lam. 2019. Persona-Aware Tips Generation. In *The World Wide Web Conference*. 1006–1016.
- [32] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 345–354.
- [33] 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*.
- [34] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 4582–4597.
- [35] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*. PMLR, 6565–6576.
- [36] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
- [37] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2022. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* (2022).
- [38] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *The Sixth International Conference on Learning Representations*.
- [39] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *arXiv preprint arXiv:2103.10385* (2021).
- [40] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- [41] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
- [42] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 188–197.
- [43] 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*. 311–318.
- [44] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- [45] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- [46] Rico Sennrich, Barry Haddow, and Alexandra Irch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1715–1725.
- [47] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3407–3412.
- [48] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural Logic Reasoning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1365–1374.
- [49] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual Learning for Explainable Recommendation: Towards Unifying User Preference Prediction and Review Generation. In *Proceedings of The Web Conference 2020*. 837–847.
- [50] Peijie Sun, Le Wu, Kun Zhang, Yu Su, and Meng Wang. 2021. An Unsupervised Aspect-Aware Recommendation Model with Explanation Text Generation. *ACM Transactions on Information Systems (TOIS)* 40, 3 (2021), 1–29.

- [51] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1784–1793.
- [52] Nava Tintarev and Judith Masthoff. 2015. Explaining Recommendations: Design and Evaluation. In *Recommender Systems Handbook* (2 ed.), Bracha Shapira (Ed.). Springer, Chapter 10, 353–382.
- [53] Quoc-Tuan Truong and Hady Lauw. 2019. Multimodal review generation for recommender systems. In *The World Wide Web Conference*. 1864–1874.
- [54] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. In *Advances in Neural Information Processing Systems*.
- [55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
- [56] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 1543–1552.
- [57] Bingbing Wen, Yunhe Feng, Yongfeng Zhang, and Chirag Shah. 2022. ExpScore: Learning Metrics for Recommendation Explanation. In *Proceedings of the ACM Web Conference 2022*. 3740–3744.
- [58] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 285–294.
- [59] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1645–1654.
- [60] Aobo Yang, Nan Wang, Hongbo Deng, and Hongning Wang. 2021. Explanation as a Defense of Recommendation. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining*. ACM, 1029–1037.
- [61] Yang Yang, Junmei Hao, Canjia Li, Zili Wang, Jingang Wang, Fuzheng Zhang, Rao Fu, Peixu Hou, Gong Zhang, and Zhongyuan Wang. 2020. Query-aware Tip Generation for Vertical Search. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2893–2900.
- [62] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.
- [63] 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*. 177–186.
- [64] 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*. 83–92.
- [65] Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. 2020. A Pre-Training Based Personalized Dialogue Generation Model with Persona-Sparse Data.. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 9693–9700.
- [66] 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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1006–1014.
- [67] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully Explainable Recommendation via Neural Logic Reasoning. In *2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.