Abstract—Counterfactual explanations interpret the recommendation mechanism by exploring how minimal alterations on items or users affect recommendation decisions. Existing counterfactual explainable approaches face huge search space, and their explanations are either action-based (e.g., user click) or aspect-based (i.e., item description). We believe item attribute-based explanations are more intuitive and persuasive for users since they explain by fine-grained demographic features, e.g., brand. Moreover, counterfactual explanations could enhance recommendations by filtering out negative items. In this work, we propose a novel Counterfactual Explainable Recommendation (CERec) to generate item attribute-based counterfactual explanations meanwhile to boost recommendation performance. Our CERec optimizes an explanation policy upon uniformly searching candidate counterfactuals within a reinforcement learning environment. We reduce the huge search space with an adaptive path sampler by using rich context information of a given knowledge graph. We also deploy the explanation policy to a recommendation model to enhance the recommendation. Extensive explainability and recommendation evaluation demonstrate CERec’s ability to provide explanations consistent with user preferences and maintain improved recommendations.

Index Terms—Explainable recommendation, counterfactual explanation, counterfactual reasoning, reinforcement learning.

I. INTRODUCTION

M

ODERN recommendation models become sophisticated and opaque by modeling complex user/item contexts, e.g., social relations [1] and item profiles [2]. Hence, there is a pressing need for faithful explanations to interpret user preferences while enabling model transparency. Explainable Recommendation Systems (XRS) aim to provide personalized recommendations complemented with explanations that answer why particular items are recommended [3]. It is fairly well-accepted that high-quality explanations help improve users’ satisfaction and recommendation persuasiveness [4], [5]. Explanations also facilitate system designers in tracking the decision-making of recommendation models for model debugging [6].

Existing XRS approaches can be roughly categorized into model-intrinsic methods [7], [8] and model-agnostic methods [9], [10], [11], [12]. Model-intrinsic methods seek recommendation models that are inherently interpretable, e.g., decision trees [7] and association rules [8], and thus are limited in their applications to prominent deep learning models [3]. Recently, model-agnostic methods have attracted increasing attention [13] to allow the recommendation model to be black-box models, e.g., neural networks [10]. Model-agnostic approaches identify correlations between input user-item interactions and output recommendations from black-box models [13], to help users understand how their behaviors [10] (e.g., click) or which item features [11], [14] (e.g., brand) contribute to the recommendation. However, existing model-agnostic approaches suffer from major limitations: 1) They mainly concentrate on factual scenarios using historical user-item interactions to identify item features. However, they typically neglect the counterfactual scenario where changing explanations can affect future recommendations. This restricts their ability to predict changes in future recommendations if explanations are modified. Incorporating these counterfactual scenarios is vital for a deeper understanding of the system and to understand how varied explanations influence recommendations. 2) They construct explanations using a fixed number of influential factors without accounting for explanation complexity [9]. Instead of seeking the minimal set of influential factors for recommendations, these methods might incorporate redundant or irrelevant elements. This can result in unnecessary computational complexity. For example, given the condition that the brand been “Apple” well explains the user’s preference, using “Apple”, “Electronic”, “California” as the explanation becomes redundant and decreases the explanation efficiency.

Recently, counterfactual explanation [15] has emerged as a favorable opportunity to solve the above questions. Counterfactual explanation interprets the recommendation mechanism via exploring how minimal alterations on items or users affect the recommendation decisions. Typically, counterfactual

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
An explanation is defined as a minimal set of influential factors that, if applied, flip the recommendation decision. To build counterfactual explanations, we have to fundamentally address: what the recommendation result would be if a minimal set of factors (e.g., user behaviors/item features) had been different [16]. With counterfactual explanations, users can understand how minimal changes affect recommendations and conduct counterfactual thinking under the “what-if” scenario [17].

Existing counterfactual explanation-based XRS can be summarized into two main categories. The first category consists of search-based approaches [18], [19], which conduct greedy searches on counterfactual instances given by perturbing user interactions or item features. These approaches determine counterfactual explanations by searching for the counterfactual instances with the highest scores evaluated by heuristic quality metrics. For example, Kaffes et al. [18] generate counterfactual instances by removing items from users’ interactions. They then use normalized length and candidate importance to measure counterfactual instances and search high-scored ones as counterfactual explanations. Xiong et al. [19] perform constrained feature perturbations on item features and search perturbed features as counterfactual explanations. However, these approaches usually incur high computational costs due to the large-scale search space of the greedy search [16]. Another category of optimization-based approaches [15], [20], [21], [22] formulates counterfactual reasoning as an optimization problem to alleviate the computation burden. The optimization goal is to find simple but essential user actions or item aspects that cause changes in user preferences. For example, Ghazimatin et al. [21] perform random walks over an interaction graph and calculate PageRank scores after removing user action edges from the graph. User actions that change PageRank scores are used as counterfactual explanations. Tan et al. [15] modify item aspect scores to observe user preference changes based on two pre-defined matrices. However, these approaches either focus on user action [20], [21], [22] or item aspect explanations [15], leaving the item attribute-based counterfactual explanations largely unexplored.

Nevertheless, item attribute-based counterfactual explanations could benefit both users’ trust and recommendation performance. On the one hand, item attribute-based counterfactual explanations are usually more intuitive and persuasive to seek users’ trust. This is because users prefer to know detailed information, e.g., which item attribute caused the film “Avatar” not to be recommended anymore? Is it the director of “Avatar”? Besides, it is essential to search for a minimal change in item attributes that would alter the recommendation. We take Fig. 1 as an example. $u_1$ gave negative feedback on $i_2$ with attributes $\{p_1, p_2, p_3, p_4\}$. We could infer attributes $\{p_1, p_2, p_3, p_4\}$ reveal negative preferences of $u_1$. However, item $i_1$ with attributes $\{p_1, p_2\}$ was liked by $u_2$, which somehow reflect positive user preferences on $\{p_1, p_2\}$. Thus, merely using attributes $\{p_1, p_2, p_3, p_4\}$ as explanations for $u_1$’s recommendations would be controversial and misunderstand the user’s preference. In fact, the slight changes between $i_1$’s attributes and $i_2$’s attributes, i.e., $\{p_3, p_4\}$, could be the true determinant factors for explaining $u_1$’s dislike. On the other hand, counterfactual explanations could boost recommendation performance since they offer high-quality negative signals of user preferences. Particularly, if we know the slightly changes $\{p_3, p_4\}$ explain $u_1$’s dislike, we could infer that item $i_3$ with $\{p_3, p_4\}$ would be disliked by $u_1$ as well. As a result, the counterfactual explanation helps generate more precise recommendations by filtering out negative items.

Following the above intuition, this work leverages knowledge graphs (KGs) that represent relations among real-world entities of users, items, and attributes to infer attribute-based counterfactual explanations meanwhile boosting recommendation performance. We propose a new Counterfactual Explainable Recommendation (CERec) that crafts the counterfactual optimization problem as a reinforcement learning (RL) task. The RL agent optimizes an explanation policy upon uniformly searching candidate counterfactuals. To reduce the search space, we propose an adaptive path sampler over a KG with a two-step attention mechanism and select item attributes as the counterfactual explanation candidates. With explanation candidates, the RL agent optimizes the explanation policy to find optimal attribute-based counterfactual explanations. We also deploy the explanation policy to a recommendation model to enhance the recommendation. The contributions are:

- To the best of our knowledge, we are the first to leverage the rich attributes in knowledge graphs to provide attribute-based counterfactual explanations for recommendations.
- We propose an RL-based framework to find the optimal counterfactual explanations, driven by an adaptive path sampler and a counterfactual reward function.
- We use counterfactual explanations to augment the recommendation model for boosting the recommendation and explainability.
- Extensive results on explainability and recommendation performance demonstrate the effectiveness of our method.

II. RELATED WORK

A. Explainable Recommendation

The explainable recommendation is proven to improve user satisfaction [23] and system transparency [24]. Research in explainable recommendation can be divided into two main categories [3], [25]. The first category of model-intrinsic approaches [7], [8], [26], [27], [28], [29] designs interpretable recommendation models to facilitate explanations.
These model-intrinsic methods train explainable models to identify influential factors for users’ preferences, and then construct explanations accordingly. For example, Shulman et al. [7] propose a per-user decision tree model to predict users’ ratings on items. Each decision tree is built with a single user as a root and items as leaves, while a linear regression is applied to learn leaf node values to build decision rules. The explanation is formed as a sequence of decisions along the decision tree. Lin et al. [8] propose a personalized association rule mining method targeted at a specific user. Explanations are generated by identifying association rules between users and items. Other works [26], [27], [28], [29], [30], [31] explore content-based approach [26] and neighborhood-based collaborative filtering [27], [28] to generate explanations based on users’ neighbors [27], [30], [31], item similarity [28], user/item features [26], or combinations thereof [29].

As modern recommendation models become complicated and black-box, e.g., embedding-based models [32], [33], [34] and deep neural networks [35], [36], developing model-intrinsic approaches [3] is not practical anymore. Thus, another category of model-agnostic methods [10], [14], [37] aims to explain black-box models in a post-hoc manner. For instance, Ghazimatin et al. [10] perform graph learning on a heterogeneous information network that unifies users’ social relations and historical interactions. The learned graph embeddings are used to train an explainable learning-to-rank model to output explanation paths. Singh et al. [14] train a Learning-to-rank latent factor model to produce ranking labels. These ranking labels are then used to train an explainable tree-based model to generate interpretable item features for ranking lists.

### B. Counterfactual Explainable Recommendation

Counterfactual explanations have been considered as satisfactory explanations [38], [39], [40] and elicit causal reasoning in humans [41], [42], [43]. Successful works on the counterfactual explanation for recommendations can be categorized into search-based and optimization-based approaches. The first category of search-based approaches [18], [19] performs the greedy search for counterfactual explanations. Kaffes et al. [18] perturb user-item interactions by deleting items from user interaction queues to generate perturbed user interactions as counterfactual explanations. Then, a breadth-first search is used to find counterfactual explanations that achieve the highest normalized length and candidate impotence scores. Xiong et al. [19] propose constrained feature perturbations on the features of items and consider the perturbed item features as counterfactual explanations. The second category of optimization-based approaches [15], [20], [21], [22] optimizes explanation models to find counterfactual explanations with minimal changes. Ghazimatin et al. [21] perform random walks over a heterogeneous information network and calculate the PageRank scores after removing user action edges from the graph. Those minimal sets of user actions that change PageRank scores are deemed counterfactual explanations. Tran et al. [22] identify a minimal set of user actions that updates the parameters of neural models.

### III. PRELIMINARY

We first introduce the collaborative knowledge graph that offers real-world attributes for explanation learning, then give the definition of attribute-based counterfactual explanation and our task formulation. The primary notations used throughout this paper are listed in Table I.

### A. Collaborative Knowledge Graph

A collaborative knowledge graph (CKG) [44] encodes user-item interactions and item attributes as a unified relational graph. Let $\mathcal{U} \subseteq \mathbb{R}^M$, $\mathcal{I} \subseteq \mathbb{R}^N$ denote the sets of users and items, respectively. The historical user-item interaction matrix $\mathcal{Y} = \{y_{ui} \mid u \in \mathcal{U}, i \in \mathcal{I}\}$ is defined according to users’ implicit feedback, where each entry $y_{ui} = 1$ indicates there is an interaction between $u$ and $i$ and otherwise, $y_{ui} = 0$. Apart from user-item interactions, we also have an external knowledge graph (KG) that profiles item attributes. Inspired by [44], we integrate rich item attributes and interaction data into a collaborative knowledge graph. Let $\mathcal{E}$ and $\mathcal{R}$ denote the entity and relation sets in the KG. We first build an item-entity alignment function $\psi : \mathcal{I} \Rightarrow \mathcal{E}$ to map items in the interaction data into KG entities, in which each item $i \in \mathcal{I}$ is mapped to an item entity $e \in \mathcal{E}$ in the KG. Then, CKG is defined as $G = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$, and $\mathcal{R}' = \mathcal{R} \cup \{(y_{ui})\}$. $I(\cdot)$ is an edge indicator that denotes the observed edge between user $u$ and item $i$ when $y_{ui} = 1$, i.e., $I(y_{ui}) = 1$ when $y_{ui} = 1$, otherwise $I(y_{ui}) = 0$.
Each triplet \((h, r, t)\) describes that there is a link \(r\) from \(h\) to \(t\), which contains rich semantic relations among real-world entities. For example, \((\text{Michael Jackson}, \text{SingerOf}, \text{Beat it})\) states the fact that Michael Jackson is the singer of the song Beat it.

### B. Counterfactual Explanation for Recommendation

The counterfactual explanation is defined as a minimal perturbation set of the original sample that, if applied, results in the desired prediction of the sample [45]. Formally,

**Definition 1 (Counterfactual Explanation [46]):** Suppose a prediction function \(f_R : \mathbb{R}^d \rightarrow \mathbb{Y}\) is given. The counterfactual explanation is a minimal perturbation set \(\Delta\) on the original input \(x \in \mathbb{R}^d\). The instance \(x_{cf} \in \mathbb{R}^d\) is obtained by applying \(\Delta\) on \(x\), and is termed the counterfactual instance for \(x\). The solution to the counterfactual explanation \(\Delta\) for a given input \(x\) is computed through an optimization problem:

\[
\arg \min_{x_{cf} \in \mathbb{R}^d} \mathcal{L}(\Delta) = \ell(f_R(x_{cf}), y) + C \cdot \theta(x_{cf}, x) \tag{1}
\]

where \(\mathcal{L}(\Delta)\) is the optimization loss for the counterfactual explanation \(\Delta\), \(\ell(\cdot)\) denotes a suitable loss function, e.g., mean squared error [47], comparing the deviation of the desired prediction \(y\) from the counterfactual prediction \(f_R(x_{cf})\). \(\theta(\cdot)\) is a penalty term for deviations of \(x_{cf}\) from the original input \(x\), i.e., encourage \(\Delta\) as minimal perturbation. \(C > 0\) denotes the regularization strength. We refer to a counterfactual explanation \(\Delta\) and a counterfactual instance \(x_{cf}\) given \(x\) and \(y \sim f_R\) as any solution to the optimization problem.

This work focuses on finding the attribute-based counterfactual explanations, in which the minimal perturbation set \(\Delta\) is defined as a set of item attributes, e.g., genre, brand. Following Definition 1, we define \(f_R\) as a Top-K recommendation model that gives the recommendation list \(Q_u\) with length \(K\) for a user \(u\), and we say \(i \in Q_u\) if item \(i\) is recommended. For a user-item pair \((u, i)\), we aim to search for a minimal item attributes set \(\Delta_{ui} = \{a_{ui}^1, \ldots, a_{ui}^l\}\). Each \(a_{ui}^i \in \Delta_{ui}\) is an attribute entity selected from a collaborative knowledge graph, which contains real-world semantics that describes the item. With \(f_R\) and \(\Delta_{ui}\), our optimization goal is to estimate whether applying the \(\Delta_{ui}\) on the original recommended item \(i\) results in replacing the \(i\) with a counterfactual item \(j\). If the optimization goal is met, \(\Delta_{ui}\) is termed an attribute-based counterfactual explanation that flips the recommendation result, and \(j\) is the counterfactual item for the original item \(i\). The above intuition is formalized as the following definition:

**Definition 2 (Attribute-based Counterfactual Explanation):** Given \(f_R\) as the prediction function of a Top-K recommendation model, and the original recommended item \(i \in Q_u \sim f_R\) for a user \(u\). An attribute-based counterfactual explanation for \((u, i)\) is defined as \(\Delta_{ui}\), such that applying the \(\Delta_{ui}\) on \(i\) results in replacing the \(i\) with a counterfactual item \(j\) for user \(u\). Meanwhile, the counterfactual explanation \(\Delta_{ui}\) is minimal such that there is no smaller set \(\Delta'_{ui}\) satisfying \(|\Delta'_{ui}| < |\Delta_{ui}|\) when \(\Delta'_{ui}\) also meets the optimization goal. With the optimized \(\Delta_{ui}\), we can generate the counterfactual explanation for recommending item \(i\) to user \(u\), which takes the following form:

\[
\text{Had a minimal set of attributes } [\Delta_{ui}(s)] \text{ been different for item } i, \text{ the recommended item would have been } j \text{ instead.}
\]

### C. Task Formulation

Given user interaction records and a knowledge graph, we can build a collaborative knowledge graph (CKG) \(G\) that unifies user-item interactions and item knowledge as a relational graph. With \(G\), we aim to construct a counterfactual explanation model that exploits rich relations among \(G\) to produce counterfactual items for original recommend items. We formulate our counterfactual explanation model as:

\[
j \sim \pi_E(u, i, f_R, G|\Theta_E) \tag{2}
\]

where \(\pi_E(\cdot)\) is the counterfactual explanation model parameterized with \(\Theta_E, f_R\) is the Top-K recommendation model that produces the recommendations. The counterfactual explanation model generates empirical distribution over items among \(G\) to yield counterfactual item \(j\) for the recommended item \(i\), which is expected to meet the counterfactual goal with a minimal set of item attributes that reverse the recommendation.

We also aim to use counterfactual items produced by \(\pi_E\) to improve the recommendation model \(f_R\). Conceptually, a counterfactual item offers high-quality negative signals to the recommendation, since it owns attributes that make the positive item not match the user’s preference anymore. Inspired by pairwise ranking [48], we pair one positive user-item interaction with one counterfactual item to aggregate counterfactual signals into the recommender \(f_R\). Formally,

\[
\min_{\Theta_E} - \ln \sigma(f_R(u, i|\Theta_R) - f_R(u, j|\Theta_R)), \forall j \sim \pi_E(\Theta_E) \tag{3}
\]

where \(f_R(\cdot)\) is the recommendation model parameterized with \(\Theta_R\) and \(\sigma(\cdot)\) is the sigmoid function. \((u, i)\) is the positive interaction that satisfies \(\exists r \in \mathcal{R}'\), s.t. \((u, r, i) \in G\). This formulation encourages positive items to receive higher prediction scores from the target user than counterfactual items. As such, the recommendation model not only produces recommendation results for counterfactual explanation model training, but is also co-trained with the counterfactual explanation model by interactively aggregating counterfactual signals.

### IV. Model Framework

We now introduce our counterfactual explainable recommendation (CERec) framework that generates attribute-based counterfactual explanations while providing improved recommendations. Our CERec is operated under a CKG [44] that unifies user-item interactions and a knowledge graph to capture complex user-item and item-attribute relations. CERec consists of three modules - one recommendation model, one graph learning module and the proposed counterfactual explanation model. The recommendation model generates ranking scores and is co-trained with the counterfactual explanation model by interactively aggregating the produced counterfactual items.
Fig. 2. CERec framework. graph learning module learns embeddings of users, items, and attributes entities from a CKG; recommendation model generates ranking scores of items for users; counterfactual path sampler uses entity embeddings to sample paths as actions for reinforcement learning agent; reinforcement learning agent learns the explanation policy by optimizing the cumulative counterfactual rewards of deployed actions from the sampler. The learned explanation policy outputs counterfactual items for original user-item interactions. Meanwhile, paths connecting original items and counterfactual items from path sampler are saved to retrieve counterfactual item attributes. Finally, counterfactual explanations are generated by abstracting real-world semantics of counterfactual items and counterfactual item attributes.

The graph learning module embeds users, items, and attribute entities among a given CKG as embedding vectors. The counterfactual explanation model conducts effective path sampling based on entity embeddings and ranking scores to discover high-quality counterfactual items. Two main parts are performed in our counterfactual explanation model: 1) counterfactual path sampler: uses entity embeddings to sample paths as actions for reinforcement learning agent; 2) reinforcement learning agent: learns explanation policy by optimizing the cumulative counterfactual rewards of deployed actions from the sampler. We depict the framework of CERec in Fig. 2. We introduce the recommendation model and the graph learning module below. Our counterfactual explanation model is detailed in the next section.

A. Recommendation Model

We now present the recommendation model $f_R$ that uses user and item latent factors for Top-$K$ recommendation. Here, we employ the pairwise learning-to-rank model CliMF [49] as the recommendation model. The CliMF initializes the IDs of users and items as latent factors, and updates latent factors by directly optimizing the Mean Reciprocal Rank (MRR) to predict ranking scores of items for users. It is worth noting that $f_R$ can be any model as long as it takes users’ and items’ embeddings as part of the input and produces ranking results, which makes our counterfactual explanation framework applicable to a broad scope of models, e.g., neural networks [33]. Specifically, we first map users and items into latent factors with the recommendation model,

$$f_R(u, i) = U_u^T V_i$$

where $U_u$ and $V_i$ denote $d$-dimensional latent factors for user $u$ and item $i$, respectively. We use the pairwise Mean Reciprocal Rank loss [49] to define our objective function to optimize recommendation model parameters $\Theta_R$ as below.

$$\mathcal{L}_R = \min_{\Theta_R} \sum_{(u,i) \in \mathcal{Y}} \left[ \ln \sigma(f_R(u, i)) + \sum_{j=1}^{\infty} \ln(1 - \sigma(f_R(u, j) - f_R(u, i))) \right]$$

(5)

where $\mathcal{Y} \in \mathbb{R}^{M \times N}$ is the historical user-item interaction matrix, $\sigma(\cdot)$ is the sigmoid function and $j \sim \pi_E(\Theta_E)$ is the counterfactual item generated from our counterfactual explanation model. By optimizing (5), we can get the ranking score for each item $i$ from user $u$. Formally,

$$P_u(i) = \frac{\exp(U_u^T V_i)}{\sum_{K=1}^{K} \exp(U_u^T V_k)}$$

(6)

where $K$ is the length of user $u$’s recommendation list $Q_u$, $P_u(i)$ indicates the ranking score of an item $i$ in $u$’s recommendation list.

B. Graph Learning Module

The graph learning module (GLM) learns users, items and attributes representations (i.e., embeddings) from the given collaborative knowledge graph $G$. The learned embeddings are deployed into our counterfactual explanation model to: (1) calculate the importance scores of user, item and attribute entities to form sampling paths as actions for counterfactual path sampler; (2) calculate the similarities among item entities to define the
counterfactual rewards of deployed actions for the reinforcement learning agent.

Inspired by recent advances in Graph Neural Networks (GNNs) [33], [34] for graph data representation, we employ GraphSAGE [32] in our GLM to learn representations for users, items and attributes entities. For an entity $e \in \mathcal{G}$, the GraphSAGE aggregates the information propagated from its neighbors $\mathcal{N}_e$ to learn $e$’s representation. As a user entity would connect with entities whose type is the item, i.e., $\mathcal{N}_e \subseteq \mathcal{I}$, the learned user embeddings capture the influence of historical user-item interactions. Analogously, item entities connect with item attribute entities such that the learned item embeddings absorb context information from item attributes. In particular, we first initialize entity representations at the 0th layer of GraphSAGE with Multi-OneHot [50] by mapping entity IDs into embeddings, where the embedding for an entity $e$ is denoted by $\mathbf{h}_e^{(0)}$. Then, at the $l$th graph convolutional layer, an entity $e$ receives the information propagated from its neighbors to update its representation,

$$\mathbf{h}_e^{(l)} = \delta \left( \mathbf{W}^{(l)} \left( \mathbf{h}_e^{(l-1)} \| \mathbf{h}_{\mathcal{N}_e}^{(l-1)} \right) \right)$$

where $\mathbf{h}_e^{(l)} \in \mathbb{R}^{d_l}$ is the embedding of an entity $e$ at layer $l$ and $d_l$ is the embedding size; $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times 2d_{l-1}}$ is the weight matrix, $\|$ is the concatenation operation and $\delta(\cdot)$ is a nonlinear activation function as LeakyReLU [51]. $\mathbf{h}_{\mathcal{N}_e}^{(l-1)}$ is the information from $e$’s neighbors $\mathcal{N}_e$ and is given by:

$$\mathbf{h}_{\mathcal{N}_e}^{(l-1)} = \sum_{e' \in \mathcal{N}_e} \frac{1}{\sqrt{\left| \mathcal{N}_e \right| \left| \mathcal{N}_{e'} \right|}} \mathbf{h}_{e'}^{(l-1)}$$

where $\mathcal{N}_e = \{ e' \mid (e, e') \in \mathcal{G} \}$ denotes $e$’s connected entities.

Having obtained the representations $\mathbf{h}_e^{(l)}$ at each graph convolutional layer $l \in \{1, \ldots, L\}$, we adopt layer aggregation mechanism [52] to concatenate embeddings at all layers into a single vector, as follows:

$$\mathbf{h}_e = \mathbf{h}_e^{(1)} + \cdots + \mathbf{h}_e^{(L)}$$

where $\mathbf{h}_e^{(i)}$ is the embedding for an entity $e$ at the $i$th layer. By performing layer aggregation, we can capture higher-order propagation of entity pairs across different graph convolutional layers. After stacking $L$ layers, we obtain the final representation for each entity among the CKG. Note that in the following, we use $\mathbf{h}_u$ to denote the embedding of user entity $u$, while $\mathbf{h}_{e_i}$ is the embedding for item entity $e_i$ and $\mathbf{h}_{e_i'}$ is the embedding for item attribute entity $e_i'$.

C. Counterfactual Explanation Model

Our counterfactual explanation model contains two main parts: the counterfactual path sampler and the reinforcement learning agent. The counterfactual path sampler performs two-step attention on users, items, and attribute embeddings from GLM to search for a high-quality path as action $a_t$ for each state $s_t$. Then, action $a_t$ and state $s_t$ are fed into our reinforcement learning agent to learn the explanation policy. Based on ranking scores produced by the recommendation model and item embeddings, the reinforcement learning agent learns the reward $r(s_t, a_t)$ at state $s_t$ and updates the explanation policy $\pi_E(\Theta_E)$ accordingly. Finally, with the learned explanation policy $\pi_E$ and path histories, our CERec generates attribute-based counterfactual explanations for recommendations. We detail our counterfactual explanation model in the next section.

V. REINFORCED LEARNING FOR COUNTERFACTUAL EXPLANATION MODEL

We now introduce our counterfactual explanation model assisted by GLM and recommendation model to generate explanation policy $\pi_E$ over reinforcement depth $T$. Our counterfactual explanation model contains two main parts: counterfactual path sampler (CPS) performs attention mechanisms on CKG entities to sample paths as actions for reinforcement learning agent; reinforcement learning agent learns the explanation policy $\pi_E$ by optimizing cumulative counterfactual rewards of the sampled actions from CPS. We introduce each part in our counterfactual explanation model as follows.

A. Counterfactual Path Sampler

The counterfactual path sampler (CPS) conducts path exploration over CKG to sample paths as actions for the latter explanation policy learning. The basic idea is to condition on the target user, start from the recommended item, learn to navigate to its item attribute, and then yield the potential counterfactual item along the sampling paths. In practice, to conduct such higher-order path sampling, large-scale CKGs are required since they encode rich relations, i.e., more than one-hop connectivity. As a result, counterfactual items are sampled from higher-hop neighbors of target user-item interactions to form the candidate action space. However, learning counterfactual explanations from the whole candidate action space is infeasible since the space would cover potentially enormous paths. Thus, our CPS is designed to reduce the candidate action space by filtering out irrelevant paths and selecting important paths for later policy learning. Here, we employ attention mechanisms to calculate the importance of the visited entity condition on the source entity to generate sampling paths. We now introduce our CPS that samples paths as actions $a_t$ at each state $s_t$ using attention mechanisms.

In particular, at each state $s_t$, our CPS produces a path as an action by a probability of $\mathbb{P}(a_t|s_t)$. Formally, $a_t \sim \mathbb{P}(a_t|s_t) = (e_t \rightarrow e_t' \rightarrow e_{t+1})$ is the path that roots at an item $e_t$ towards another item proposal $e_{t+1}$, where $(e_t, e_t'), (e_t', e_{t+1}) \in \mathcal{G}$ and $e_t, e_{t+1} \in \mathcal{I}$ are connected via the item attribute $e_t'$. The action $a_t = (e_t \rightarrow e_t' \rightarrow e_{t+1})$ is generated by the two-step path sampling: 1) choose an outgoing edge from $e_t$ to the internal item attribute entity $e_t'$; 2) determine the third entity $e_{t+1}$ conditioned on $e_t'$. We separately model the confidence of each exploration step into two attention mechanisms, i.e., $\mathbb{P}(e_t, e_t')|s_t$ and $\mathbb{P}(e_t', e_{t+1})|s_t$. The first attention mechanism $\mathbb{P}(e_t, e_t')|s_t$ specifies the importance of item attributes for $e_t$, which are sensitive to state $s_t = (u, e_t)$, i.e., user $u$ and item $e_t$. Formally, for each outgoing edge from $e_t$ to its item attribute $e_t' \in \mathcal{N}_{e_t}$, we first obtain the embeddings of item $e_t$, attribute $e_t'$ and user $u$ from (9), denoted by $\mathbf{h}_{e_t}, \mathbf{h}_{e_t'}$ and $\mathbf{h}_u$. Then, the importance score of item attribute attribute
\[ e'_t \text{ is:} \]
\[ \alpha_1(e_t, e'_t) = h^u_t \delta(h_u \odot h^e_t) \]  
(10)
where \( \alpha_1 \) is the attention score at the first attention mechanism. \( \odot \) is the element-wise product and \( \delta(\cdot) \) is LeakyReLU [51]. Thereafter, we normalize the scores of all neighbors of \( e_t \) as:
\[ \mathbb{P}((e_t, e'_t) \mid s_t) = \frac{\exp(\alpha_1(e_t, e'_t))}{\sum_{e'_t \in \mathcal{N}_{e_t}} \exp(\alpha_1(e_t, e'_t))} \]  
(11)
where \( \mathcal{N}_{e_t} \) is the neighbor item attributes set for \( e_t \).

Having selected item attribute \( e'_t \), we employ another attention mechanism \( \mathbb{P}((e'_t, e_{t+1}) \mid s_t) \) to decide which item from its neighbors \( \mathcal{N}_{e'_t} \) as the counterfactual item proposal. We first calculate the attention score of \( e_{t+1} \in \mathcal{N}_{e'_t} \) based on attribute embedding of \( e'_t \), item embedding of \( e_{t+1} \) and user embedding of \( u \), as:
\[ \alpha_2(e'_t, e_{t+1}) = h^u_{t+1} \delta(h^e_{t+1} \odot h^e_t) \]  
(12)
where \( \alpha_2 \) is the attention score at the second attention mechanism. \( h_u, h_{t+1}^e \) and \( h^e_t \) are embeddings of user \( u \), attribute \( e'_t \) and item \( e_{t+1} \). Then, we normalize attention scores for all item neighbors in \( \mathcal{N}_{e'_t} \) to generate the selection probability of item \( e_{t+1} \). Since we care for generating valid counterfactual items as proposals, we filter out irrelevant items that do not meet the counterfactual goal, i.e., those being recommended for \( u \) in the recommendation list \( Q_u \). Formally:
\[ \mathbb{P}((e'_t, e_{t+1}) \mid s_t) = \begin{cases} \frac{\exp(\alpha_2(e'_t, e_{t+1}))}{\sum_{e''_t \in \mathcal{N}_{e'_t}} \exp(\alpha_2(e''_t, e_{t+1}))}, & e_{t+1} \notin Q_u \\ 0, & e_{t+1} \in Q_u \end{cases} \]  
(13)

Finally, the two attention mechanisms are aggregated into the CPS \( \mathbb{P}(a_t \mid s_t) \) to yield the path \( a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1}) \) as an action for each state \( s_t \):
\[ \mathbb{P}(a_t \mid s_t) = \mathbb{P}((e_t, e'_t) \mid s_t) \cdot \mathbb{P}((e'_t, e_{t+1}) \mid s_t) \]  
(14)
where \( \mathbb{P}((e_t, e'_t) \mid s_t) \) is the probability of stepping from \( e_t \) to \( e'_t \) and is derived from (11); \( \mathbb{P}((e'_t, e_{t+1}) \mid s_t) \) derived from (13) is the probability of selecting \( e_{t+1} \) as the counterfactual item proposal. With \( \mathbb{P}(a_t \mid s_t) \), we can generate the action \( a_t \) for each state \( s_t \) for explanation policy learning.

### B. Reinforcement Learning Agent

We define the counterfactual explanation model as the path-based reinforcement learning (RL) agent to discover counterfactual items. Each action \( a_t \) is a path toward a candidate counterfactual item. The counterfactual reward \( r(s_t, a_t) \) measures whether the action \( a_t \) returns a valid counterfactual item for the current state \( s_t \).

1) Agent: Formally, the counterfactual explanation model is formulated as a Markov Decision Process (MDP) \( \mathcal{M} = \{S, A, \mathcal{P}, \mathcal{R}\} \), where \( s_t \in S \) is the state absorbing the current user and the visited entity, \( a_t \in A \) is the action deposited to the current state. \( \mathcal{P} \) is the transition of states, and \( \mathcal{R} \) is the reward function. In the policy learning, the explanation policy \( \pi_E(a_t \mid s_t) \) selects an action \( a_t \in A \) to take conditioning on the current state \( s_t \in S \), and the counterfactual explanation model receives counterfactual reward \( r(s_t, a_t) \in \mathcal{R} \) for this particular state-action pair. The final explanation policy is learned to maximize the expected cumulative counterfactual rewards. We introduce these key elements for RL as follows.

- **State \( S \):** A continuous state space describing a target user and the currently visited item entity among the CKG.

  Formally, for a user \( u \), at step \( t \), state \( s_t \) is defined as 
  \( s_t = (u, e_t) \), where \( u \in \mathcal{U} \) is a user and \( e_t \in \mathcal{E} \) is the item entity the agent visits currently. The initial state \( s_0 \) is \((u, i)\) and \( i \) is the positive item of \( u \), i.e., \( u, i \in \mathcal{Y} \).

- **Action \( A \):** A discrete space containing actions available for policy learning. The action \( a_t \in A \) is a path sampled from our CPS.

- **State Transition \( \mathcal{P} \):** A state transition containing transition probabilities of the current states to the next states. Given \( a_t \), the transition to the next state \( s_{t+1} \) is \( \mathcal{P}(s_{t+1} \mid s_t, a_t) \in \mathcal{P} = 1 \), where \( s_{t+1} = (u, e_{t+1}) \), \( s_t = (u, e_t) \) and \( a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1}) \).

- **Counterfactual Reward \( \mathcal{R} \):** The counterfactual reward measures whether the visited item \( e_{t+1} \) is a valid counterfactual item by deploying action \( a_t = (e_t \rightarrow e'_t \rightarrow e_{t+1}) \) at state \( s_t \), which is defined based on the two criteria: 1) **Rationality** [16]: \( e_{t+1} \) should receive high confidence of being removed from the current user’s recommendation list compared with the original item \( e_t \); 2) **Similarity** [53]: as a counterfactual explanation requires the minimal change of item attributes between counterfactual item and original item, \( e_{t+1} \) should be as similar as possible with the original item \( e_t \). The formal definition of the reward \( r(s_t, a_t) \) is given by:

\[ r(s_t, a_t) = \begin{cases} 1 + \cos(h^e_{t+1}, h^e_{t+1}), & \text{if } \mathbb{P}_u(e_t) - \mathbb{P}_u(e_{t+1}) \geq \epsilon \\ \cos(h^e_{t}, h^e_{t+1}), & \text{otherwise} \end{cases} \]  
(15)

where \( \epsilon \) is the recommendation threshold determines Rationality. \( \epsilon \) is defined as the margin between ranking scores of the original item \( e_t \) and the \( K \)th item (i.e., \( Q^K_u \)) in user’s recommendation list \( Q_u \), i.e., \( \epsilon = \mathbb{P}_u(e_t) - \mathbb{P}_u(Q^K_u) \).

\( \cos(h^e_{t}, h^e_{t+1}) \) is the cosine similarity between item embeddings \( h^e_{t} \) and \( h^e_{t+1} \) and is used to measure Similarity, i.e., \( \cos(h^e_{t}, h^e_{t+1}) = \frac{h^e_{t} \cdot h^e_{t+1}}{\|h^e_{t}\| \cdot \|h^e_{t+1}\|} \). Note that \( h^e_{t} \) and \( h^e_{t+1} \) are obtained by (9).

2) **Objective Function:** Using the trajectories \( \{S, A, \mathcal{P}, \mathcal{R}\} \) from the agent, our counterfactual explanation model seeks a counterfactual explanation policy \( \pi_E \) that maximizes the cumulative rewards \( R(\pi_E) \) over reinforcement depth \( T \):

\[ R(\pi_E) = \mathbb{E}_{s_0 \sim S, a_0 \sim \mathbb{P}_0(a_0 \mid s_0)} \mathbb{E}_{s_{t+1} \sim \mathbb{P}(s_{t+1} \mid s_t, a_t)} \left[ \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \]  
(16)

where \( T \) is the terminal step determines reinforcement depth, \( \gamma^t \) is a decay factor at current step \( t \in T \). \( \pi_E \) is the explanation policy that produces counterfactual items for users’ recommended actions.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
items. The final counterfactual explanations are formed by distilling the attributes of counterfactual items and their real-world semantics.

VI. MODEL OPTIMIZATION

We adopt iteration optimization [54] to optimize the recommendation model and the counterfactual explanation model. The algorithm for the model optimization is presented in Algorithm 1. The recommendation model is initialized to provide ranking scores for counterfactual explanation model training (line 1); then, the counterfactual explanation model is optimized to output high-quality counterfactual items for positive user-item interactions (lines 4–13). Thereafter, the recommendation model is updated by pairing one positive user-item interaction with one counterfactual item generated through the counterfactual explanation model (lines 14–19). In the following, we detail the recommendation model optimization and the explanation policy optimization.

A. Recommendation Model Optimization

We first initialize the recommendation model by pairing one positive interaction $y_{ui} \in \mathcal{Y}$ with one unobserved item $v \in \mathcal{I}$ sampled from uniform sampling [55]. Then, the recommendation model is optimized by training together with the counterfactual explanation model. At each iteration, the counterfactual explanation model outputs a counterfactual item $j \sim \pi_E(\Theta_E)$, which is then fed into the recommendation model to update user and item latent factors $U$ and $V$ in (4). Finally, the loss $L_R$ in (5) w.r.t. model parameters $\Theta_R$ is optimized by using stochastic gradient descent (SGD) [56].

B. Explanation Policy Optimization

Our counterfactual explanation model involves discrete sampling steps, i.e., the counterfactual path sampler, which blocks gradients when performing differentiation. Conventional policy gradient methods such as stochastic gradient descent [56] fail to calculate such hybrid gradients. Thus, we solve the optimization problem through REINFORCE with baseline [57] for explanation policy optimization. Having obtained the policy optimization function from (16), and the sampling function from (14), the optimization goal is to combine the two functions and optimize them together with a REINFORCE gradient w.r.t. model parameters $\Theta_E$:

$$\mathcal{L}_E = \nabla_{\Theta_E} R(\pi_E)$$
$$= \nabla_{\Theta_E} \mathbb{E}_{s_0 \sim S, a_t \sim P(a_t | s_t), s_{t+1} \sim P(s_{t+1} | s_t, a_t)} \left[ \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]$$
$$\simeq \frac{1}{T} \sum_{t=0}^{T} \left[ \gamma^t r(s_t, a_t) \nabla_{\Theta_E} \log P(a_t | s_t) \right]$$

(17)

where $\Theta_E$ are learnable parameters for our counterfactual explanation model.

Algorithm 1: Recommendation Model and Explanation Policy Optimization.

```
Input:\ Users U, items I, user-item interactions \mathcal{Y}, embedding size d
Output:\ Optimized recommendation model parameters \Theta_R and counterfactual explanation model parameters \Theta_E
1. Initialize \Theta_R with uniform sampling;
2. Initialize \Theta_E with random weights;
3. for epoch \leftarrow 1 to N do
4. Initialize accumulate reward $R = 0$, reward $r(s_t, a_t) \leftarrow 0$, state $s_0 \in \mathbb{R}^d$ with random weights, action $a_0$ uniformly sampled from $I$, decay factor $\gamma^0 = 1$;
5. Fix \Theta_R: // counterfactual explanation model optimization
6. Receive initial observation state $s_1$;
7. for $t \leftarrow 1$ to $T$ do
8. Sample action $a_t$ according to Eq. (14) (CPS $P(a_t | s_t)$);
9. Execute action $a_t$ and observe the reward $r(s_t, a_t)$ according to Eq. (15);
10. Accumulate reward $R \leftarrow R + \gamma^t r(s_t, a_t)$;
11. Transit to next state $s_{t+1}$;
end
12. Update \Theta_R using REINFORCE gradient:
$\nabla_{\Theta_R} R(\pi_E) \\ \simeq \frac{1}{T} \sum_{t=0}^{T} \left[ \gamma^t r(s_t, a_t) \nabla_{\Theta_E} \log P(a_t | s_t) \right]$
13. Update \Theta_E using REINFORCE gradient:
$\nabla_{\Theta_E} R(\pi_E) \\ \simeq \frac{1}{T} \sum_{t=0}^{T} \left[ \gamma^t r(s_t, a_t) \nabla_{\Theta_E} \log P(a_t | s_t) \right]$
14. Fix \Theta_E: // recommendation model optimization
15. for iteration \leftarrow 1 to M do
16. Sample a counterfactual item $j \sim \pi_E(\Theta_E)$ for user $u \in U$;
17. Compute $L_R$ according to Eq. (5);
18. Update \Theta_R using SGD on $L_R$;
end
21. return Optimized \Theta_R and \Theta_E;
```

C. Time Complexity Analysis

Our recommendation model (cf. Section IV-A) performs matrix factorization with a complexity of $O(|O|)$. For the graph learning module (cf. Section IV-B), establishing node representations has a complexity of $O(\sum_{l=1}^{L} (|G| + |O^+|) d_l d_{l-1})$, where $L$ is the number of layers, $|G|$ is the size of the graph, $|O^+|$ is the size of the positive observation set, and $d_l$ and $d_{l-1}$ are the dimensions of the current and previous layers, respectively. For the counterfactual explanation model, the complexity is mainly determined by the attention score calculation in the counterfactual path sampler (cf. Section V-A). The complexity of the attention score calculation is $O(2T|O^+|d_N^2)$, where $T$ is the reinforcement depth, $|O^+|$ is the size of the positive observation set, $|N_c|$ is the size of the sampled counterfactual neighbors, and $d$ is the dimension of the embeddings. In total, the time complexity is given by: $O(|O| + \sum_{l=1}^{L} (|G| + |O^+|) d_l d_{l-1} + 2T|O^+|n_2 d^2)$.  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE II
STATISTICS OF THE DATASETS

| Dataset            | Last-FM | Amazon-book | Yelp2018 |
|--------------------|---------|-------------|----------|
| #Users             | 23,566  | 70,679      | 45,919   |
| #Items             | 48,123  | 24,915      | 45,538   |
| #Interactions      | 3,034,796| 847,233     | 1,185,068|
| #Density           | 0.268%  | 0.048%      | 0.057%   |
| #Entities          | 58,266  | 88,572      | 90,964   |
| #Relations         | 9       | 39          | 42       |
| #Triplets          | 464,567 | 2,557,746   | 1,853,704|

Density is computed by | Interactions | #Users - #Items |

VII. EXPERIMENTS

We thoroughly evaluate the proposed CERec for counterfactual explainable recommendations on three publicly available datasets. We evaluate CERec in terms of recommendation performance and explainability performance to answer the following research questions:

- **RQ1**: Whether CERec with a counterfactual explanation model could improve the recommendation performance compared with state-of-the-art recommendation models?
- **RQ2**: Are the counterfactual explanations generated by CERec appropriate to explain users’ preferences?
- **RQ3**: How do different components (i.e., graph learning module, counterfactual path sampler, reinforcement learning agent) affect CERec’s performance?
- **RQ4**: How do different parameters (i.e., reinforcement depth T, training epoch) impact CERec’s performance?
- **RQ5**: How is the computation cost of CERec?

A. Experimental Setup

1) Dataset: We use three publicly available datasets: Last-FM, Amazon-book and Yelp2018. The statistics of these datasets are presented in Table II, which depicts historical user-item interactions and the knowledge graphs. The Amazon-book \(^1\) [58] dataset is a widely used book recommendation dataset, and the Last-FM \(^2\) [59] is a music listening dataset. For Amazon-book and Last-FM, we first map their items into Freebase \([60]\) entities. Then, the knowledge graphs for Amazon-book and Last-FM are build by extracting knowledge-aware facts for each item from the Freebase. Yelp2018 \(^3\) [44] is a business recommendation dataset. For Yelp2018, we extract its item knowledge from the business information network to construct the knowledge graph. Each dataset is processed by the following settings: for user-item interactions, we adopt a 10-core setting, i.e., retaining users and items with at least ten interactions. Each knowledge-aware fact (i.e., <user, item>, <item, attribute>) is represented as an edge among the collaborative knowledge graph (CKG). Moreover, to ensure CKG quality, we filter out infrequent entities (i.e., lower than 10 in both datasets) and retain the relations appearing in at least 50 triplets.

2) Evaluation Metrics: To evaluate the recommendation performance, we use three popular Top-K recommendation metrics: Recall@K, Normalized Discounted Cumulative Gain (NDCG)@K and Hit Ratio (HR)@K. The K is set as 20 by default. The average results w.r.t. the metrics over all users are reported in Table III while a Wilcoxon signed-rank test \([61]\) is performed in Table III to evaluate the significance.

We evaluate the quality of explanations in terms of consistency. The consistency measures to what extent the attributes in counterfactual explanations are appropriate to explain users’ preferences. We adopt the explanation Precision, Recall and F1 protocols following CountER \([15]\). In particular, we first build the ground-truth attribute sets of evaluations for Precision, Recall and F1 protocols. As we aim to search for a minimal item attributes set that can flip the positive user-item interaction to the negative. The ground-truth set, therefore, absorbs item attributes that cause the user to dislike the item. Formally, the ground truth set is defined as \(O_{ui} = \{o_{ui}^1, \ldots, o_{ui}^p\}\), where \(o_{ui}^p = 1\) if user \(u\) has negative preference on the \(p\)th attribute of item \(i\); otherwise, \(o_{ui}^p = 0\). The implementation of negative item attributes extraction is detailed in Section VII-A4. Our model produces the attributes set \(\Delta_{ui} = \{a_{ui}^1, \ldots, a_{ui}^p\}\), which constitutes the counterfactual explanation for user-item pair \((u, i)\). Then, for each user-item pair, the Precision, Recall and \(F_1\) of \(\Delta_{ui}\) with regard to \(O_{ui}\) are calculated by:

\[
\text{Precision} = \frac{\sum_{p=1}^{r} o_{ui}^p \cdot I(a_{ui}^p)}{\sum_{p=1}^{r} o_{ui}^p}, \quad \text{Recall} = \frac{\sum_{p=1}^{r} o_{ui}^p \cdot I(a_{ui}^p)}{\sum_{p=1}^{r} o_{ui}^p}, \quad F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

where \(I(a_{ui}^p)\) is an identity function such that \(I(\cdot) = 1\) when \(a_{ui}^p \neq 0\) and otherwise, \(I(\cdot) = 0\).

3) Baselines: To evaluate the recommendation and explainability performance, we compare our CERec with five popular recommendation models as well as three attribute-aware baselines. Each baseline is aligned with different evaluation tasks either for (I) recommendation evaluation or (II) explainability evaluation. The baseline models are listed in the following:

- **NeuMF** \([36]\) (I): extends Matrix Factorization (MF) to Deep Neural Network (DNN) for modeling user-item interactions.
- **MCRec** \([2]\) (I): uses DNN to model context information from Heterogeneous Information Network (HIN) for recommendations.
- **KGpolicy** \([62]\) (I): uses an RL agent to explore negative items over a KG. It trains a Bayesian Personalized Ranking model with sampled negatives for recommendations.
- **KGRQ** \([63]\) (I): models KG information with a Graph Convolution Network (GCN). It learns a recommendation policy with KG-enhanced state representations.
- **KGAT** \([44]\) (I): introduces the CKG concept and learns node embeddings by aggregating CKG neighbors for recommendations.
- **NeuACF** \([64]\) (II): learns attribute-based latent factors of users and items based on their similarities with user preferences. Explanations are the top 10 similar attributes with user preferences.
TABLE III
RECOMMENDATION EVALUATION: BOLD NUMBERS ARE THE BEST RESULTS, BEST BASELINES ARE MARKED WITH UNDERLINES

| Model  | Dataset | Last-PM | Amazon-book | Yelp2018 |
|--------|---------|---------|-------------|----------|
|        | K       | Recall@K | NDCG@K | HR@K | Recall@K | NDCG@K | HR@K | Recall@K | NDCG@K | HR@K |
| NeuMF  | 20      | 0.0651 | 0.0767 | 0.2921 | 0.0799 | 0.0611 | 0.1010 | 0.0279 | 0.0380 | 0.1008 |
|        | 40      | 0.0807 | 0.0848 | 0.3409 | 0.1377 | 0.0769 | 0.2660 | 0.0357 | 0.0411 | 0.1241 |
|        | 60      | 0.0913 | 0.0955 | 0.3788 | 0.1581 | 0.0888 | 0.3007 | 0.0389 | 0.0448 | 0.1552 |
|        | 80      | 0.1005 | 0.1009 | 0.4267 | 0.1888 | 0.0974 | 0.3332 | 0.0411 | 0.0477 | 0.2008 |
| MRec   | 20      | 0.0770 | 0.0821 | 0.2811 | 0.0879 | 0.0666 | 0.1420 | 0.0327 | 0.0412 | 0.1234 |
|        | 40      | 0.0825 | 0.0845 | 0.3124 | 0.1041 | 0.0712 | 0.2177 | 0.0338 | 0.0424 | 0.1406 |
|        | 60      | 0.0919 | 0.0919 | 0.3888 | 0.1801 | 0.0844 | 0.2805 | 0.0429 | 0.0436 | 0.1721 |
|        | 80      | 0.1112 | 0.0945 | 0.4282 | 0.2257 | 0.0957 | 0.3672 | 0.0457 | 0.0437 | 0.2205 |
| KGpolicy | 20    | 0.0761 | 0.0689 | 0.3197 | 0.1242 | 0.0684 | 0.2211 | 0.0557 | 0.0435 | 0.1528 |
|        | 40      | 0.1018 | 0.0763 | 0.4091 | 0.1716 | 0.0805 | 0.2947 | 0.0649 | 0.0460 | 0.2588 |
|        | 60      | 0.1179 | 0.0815 | 0.4639 | 0.2048 | 0.0879 | 0.3417 | 0.0832 | 0.0547 | 0.3300 |
|        | 80      | 0.1302 | 0.0855 | 0.5006 | 0.2315 | 0.0935 | 0.3762 | 0.0883 | 0.0618 | 0.3841 |
| KGQQR  | 20      | 0.0821 | 0.0856 | 0.3162 | 0.1076 | 0.0787 | 0.2182 | 0.0417 | 0.0458 | 0.1621 |
|        | 40      | 0.0932 | 0.0923 | 0.4229 | 0.1652 | 0.0891 | 0.2358 | 0.0456 | 0.0499 | 0.1899 |
|        | 60      | 0.0999 | 0.0939 | 0.5228 | 0.1987 | 0.0912 | 0.3077 | 0.0512 | 0.0535 | 0.2172 |
|        | 80      | 0.1132 | 0.0947 | 0.5323 | 0.2212 | 0.0942 | 0.3531 | 0.0557 | 0.0587 | 0.2566 |
| KGAT   | 20      | 0.0870 | 0.0897 | 0.5292 | 0.0801 | 0.0791 | 0.2006 | 0.0444 | 0.0472 | 0.2341 |
|        | 40      | 0.0962 | 0.0918 | 0.3762 | 0.1435 | 0.0812 | 0.2634 | 0.0469 | 0.0511 | 0.2666 |
|        | 60      | 0.1083 | 0.0937 | 0.4515 | 0.1766 | 0.0922 | 0.3244 | 0.0501 | 0.0546 | 0.3015 |
|        | 80      | 0.1232 | 0.0939 | 0.5464 | 0.2378 | 0.0985 | 0.3921 | 0.0544 | 0.0577 | 0.3655 |
| CERec  | 20      | 0.1015 | 0.0900 | 0.3867 | 0.1406 | 0.0803 | 0.2451 | 0.0650 | 0.0495 | 0.2515 |
|        | 40      | 0.1304 | 0.0993 | 0.4891 | 0.1881 | 0.0926 | 0.3174 | 0.1025 | 0.0555 | 0.3549 |
|        | 60      | 0.1478 | 0.1052 | 0.5295 | 0.2203 | 0.0999 | 0.3624 | 0.1317 | 0.0639 | 0.4218 |
|        | 80      | 0.1603 | 0.1094 | 0.5628 | 0.2462 | 0.1054 | 0.3948 | 0.1572 | 0.0706 | 0.4739 |

4) Implementation Details: We train the recommendation model by the CliMF with the train/test/validate sets, which are split from user-item interactions with a proportion of 60%/20%/20% of the original dataset. We optimize the CliMF using stochastic gradient descent (SGD). The same data splitting and gradient descent methods are also applied in all baselines. The three datasets for training our counterfactual explanation model are prepossessed into collaborative knowledge graphs, and the REINFORCE policy gradient is calculated to update the parameters. The REINFORCE is also applied to KGpolicy and KGQQR, and the same collaborative knowledge graphs are used in KGpolicy and KGAT. The embedding size for all baselines and our CERec is fixed as d = 64. Two graph convolutional layers with {32, 64} output dimensions are performed for the graph learning in our model. All neural network-based (i.e., DNN, GCN) baselines also keep 2 layers. For MRec, NeuACR and CaDSI, we use meta-paths with the path scheme of user-item-attribute-item to ensure the model compatibility, e.g., user-book-author-book for Amazon-book dataset. For counterfactual explanation, we fix the parameters of the trained counterfactual explanation model to produce one counterfactual item for each positive user-item interaction. The final explanation comprises item attributes that connect with the counterfactual item in the CKG but not with the positive user-item interaction. For explanation consistency evaluation, we construct ground-truth attribute sets using dynamic negative sampling (DNS). We first train an attribute-aware MF model by feeding positive user-item interactions and random negative item attributes. The DNS then uniformly draws one item attribute from the attribute space and feeds its latent factors into the MF model to predict the preference score. Item attributes with the highest scores are selected as negative samples to train the MF model recursively. Among multiple rounds of sampling, the DNS generates negative item attributes that match users’ negative preferences. The hyper-parameters of all models are

• CaDSI [30] (II): disentangles user embeddings as separated user intent chunks. It learns attribute-aware intent embeddings by assigning item context trained from a HIN to each user intent chunk. Explanations are the top 10 attributes of user intent.
• RDEexp (II): We randomly select 10 attributes from the item attribute space for each user-item interaction and generate explanations based on the selected attributes.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
chosen by the grid search, including learning rate, $L_2$ norm regularization, discount factor $\gamma$, etc. The maximum epoch for all methods is set as 400, while an early stopping strategy is performed, i.e., if the loss stops increasing, then terminate the model training.

**B. Counterfactual Enhanced Recommendation (RQ1)**

In this section, we present the recommendation performance evaluation to answer the RQ1. At each iteration, the counterfactual explanation model in our $CERec$ produces one counterfactual item for each positive user-item interaction to recursively train the recommendation model. Thus, we are interested in knowing whether incorporating counterfactual items could boost our recommendation performance. We present the recommendation performance of our $CERec$ and baselines in Table III, and here are our main findings.

- Our proposed $CERec$ equipped with a counterfactual explanation model consistently outperforms all baselines across three datasets on both evaluation metrics. For example, $CERec$ obtains 16.6%, 0.3% and 17.4% improvements for Recall@20, NDCG@20 and HR@20 respectively over the best baseline on Last-FM dataset. By designing a counterfactual explanation model, $CERec$ is capable of exploring the high-order connectivity among the CKG to generate counterfactual items for recommendation model training. This verifies the effectiveness of our counterfactual explanation model in producing high-quality counterfactual items that can boost the recommendation. Counterfactual items provide high-quality negative signals of user preference. By pairing one counterfactual item with one positive user-item interaction for training, our recommendation model learns to distinguish between positive items and negative items to generate more precise recommendations.

- By jointly analyzing the results across the three datasets, our $CERec$ achieves the most significant improvements over baselines on the Yelp2018 dataset, e.g., 78.0%, 14.2%, and 23.3% improvements for Recall@80, NDCG@80, and HR@80, respectively. The Yelp2018 dataset records a large number of inactive users that have few interactions with items. It is well acknowledged that inferring user preference for inactive users is challenging and limits the performance of recommendation systems [66]. However, our $CERec$ can still achieve favorable recommendation performance on the Yelp2018. We attribute the improvements of our $CERec$ to the augmenting of counterfactual items that could infer negative user preferences for inactive users and thus boost the recommendation.

- Reinforcement learning (RL) endows recommendation models with improved recommendation performance. In particular, in Table III, the recommendation performance of RL-based baselines (i.e., KGPolicy and KGQR) beats non-RL-based baselines (i.e., NeuMF, MCREC and KGAT) in most cases. For instance, KGPolicy achieves 0.1018 Recall@40 while NeuMF only has Recall@40 of 0.0807. Unlike static baselines without using RL, RL-based baselines could handle dynamic user-item interactions with the recommendation environments and thus better capture users’ preference shifts. This observation suggests that incorporating reinforcement learning helps improve recommendation performance by considering the dynamic nature of user preferences. Our $CERec$ also harnesses the power of reinforcement learning to adapt to dynamic user preferences when inferring counterfactual items. Those counterfactual items identified by our $CERec$ are proper to explain users’ ever-changing preferences. Ultimately, $CERec$ enhances the recommendation model by augmenting these high-quality counterfactual items, resulting in superior recommendation performance compared to all other baselines.

- Among the knowledge-aware models, our $CERec$ consistently outperforms MCREC, KGPolicy, KGQR, and KGAT. This is mostly because these knowledge-aware baselines might not fully utilize the item knowledge by lacking the counterfactual reasoning ability as in our $CERec$. All knowledge-aware baselines employ various attention mechanisms to capture users’ preferences on item knowledge w.r.t. different item attributes. However, they fail to consider the underlying mechanism that really triggered users’ interactions. On the contrary, our $CERec$ learns to discover the item attributes that cause the change of users’ interactions, thus resulting in a promoted recommendation performance.

**C. Counterfactual Explanation Quality (RQ2)**

To answer RQ2, we evaluate the explainability of our framework by reporting the quantitative results of explanation evaluation metrics. Then, we discuss the interpretability of our generated counterfactual explanations by distilling their real-world semantics.

1) **Quantitative Results:** We report the Precision, Recall and $F_1$ (cf. (18)) of explanations generated by our $CERec$ and baseline models in Table IV to test the explanation consistency. The three evaluation metrics reflect what extent the item attributes in explanations are appropriate to explain users’ preferences. By analyzing Table IV, we have several observations.

First, the RDExp method performs very poorly on both Precision, Recall, and $F_1$. The RDExp generates explanations by randomly sampling item attributes from the knowledge graph provided. The poor performance of RDExp shows that randomly choosing item attributes as explanations can barely reveal the reasons for recommendations. This demonstrates that high-quality explanations require the appropriate choice of item attributes. Second, the counterfactual explanations generated by our $CERec$ consistently show superiority against all baselines in finding item attributes that are consistent with users’ preferences. This indicates that our $CERec$ achieves superior explainability and can generate more relevant attribute-based explanations that truly match user preference. Unlike NeuACF and CaDSI, which generate explanations by selecting a fixed number of item attributes indicated by their importance scores, our $CERec$ searches a minimal set of item attributes that would flip the recommendation result. We hence conclude that inferring item
proposed GLM. We present evaluation results for Amazon-book and Yelp2018 on both Top-K recommendation and explanation consistency evaluation tasks. Since the results on Last-FM and Amazon-book are similar, we include only the results of Amazon-book in this study.

2) Interpretability of Counterfactual Explanations: We present a case study on the interpretability of counterfactual paths that root at a positive user-item interaction and end at its counterfactual item. Each freebase entity among paths are mapped into real-world entities. We show the real-world cases in Fig. 3 and give our observations in the following.

The first example (Case 1) comes from the Last-FM dataset, where user $u_{17724}$ positively interacted with “Perfect 10” in his listening record. Our CERec picks the “Loving you is killing me” as the counterfactual item, which shares three overlapping item attributes with the “Perfect 10”. Conventional explainable recommendation models would pick these three common item attributes to generate explanations as “you like Perfect 10 is because you like pop music that was certificated for its single#2 version”. However, this explanation cannot answer why the “Loving you is killing me” as the counterfactual item, which shares three overlapping item attributes with the “Perfect 10”, then it will not be recommended anymore.” This counterfactual explanation provides us with deeper insights into users’ real preferences. For example, analyzing the attribute differences, we find that although “Perfect 10” shares the same type Pop as “Loving you is killing me”, it is actually the Rock music published by Mercury. On the contrary, the “Loving you is killing me” that holds the opposite music type Soul and different polisher would potentially be disliked by the same user.

Analogously, in Case 2 from the Amazon-book, our CERec picks the “Death of Kings” as the counterfactual item for the positive interaction between $u_{1853}$ and “The Art of Dreaming”, and uses item attributes Historical and United Kingdom as the counterfactual explanation to reflect $u_{1853}$’s preferences on book genre and country. These counterfactual explanations are more rational than conventional explanations, since they discard tedious associations of item attributes and expand explanations to counterfactual attributes.

D. Ablation Study on CERec (RQ3)

We have three contributing components in our CERec (cf. Fig. 2): graph learning module (GLM), counterfactual path sampler (CPS) and reinforcement learning agent (RLA). We explore how replacing key designs in GLM, CPS and RLA impact CERec’s performance. We present evaluation results for Amazon-book and Yelp2018 on both Top-$K$ recommendation and explanation consistency evaluation tasks. Since the results on Last-FM and Amazon-book are similar, we include only the results of Amazon-book in this study.

1) Impact of Graph Learning Methods: Our graph learning module (GLM) employs GraphSAGE [32] to quantify complex graph topology and node features of a given collaborative knowledge graph as graph embeddings. To investigate how different graph learning methods affect CERec’s performance, we replace GraphSAGE in our GLM with other graph learning methods, including Graph Convolutional Network (GCN) [67], Graph Neural Network (GNN) [68], Simple Graph Convolutional Network (SGC) [69], Light Graph Convolution Network (LGC) [70] and Graph Attentional Network (GAT) [71]. Those graph learning methods are both state-of-the-art benchmarks in graph representation tasks [72]. Table V shows the experimental results, wherein CERec-GCN indicates the CERec trained with Graph Convolutional Network (GCN); similar notations for other variants. Following the parameter settings in Section VII-A4, for fair comparisons, we keep the same embedding size (i.e., $d = 64$) for all variants and the original CERec in Table V. Besides, two graph layers with $\{32, 64\}$ output dimensions are performed for all variants and CERec. Upon analyzing Table V, we have several observations.

- Our CERec with a GraphSAGE-based GLM produces the best recommendation and explanation performance compared with CERec with other graph learning methods. This verifies the effectiveness of using GraphSAGE in our CERec. GraphSAGE exhibits inductive learning, in contrast to GCN and GNN, which employ transductive learning. The inductive learning nature of GraphSAGE
helps our CERec to generalize well to unseen nodes, as inductive learning uses partial graph Laplacian to allow better inferences of unseen nodes compared with transductive learning. Besides, compared with SGC and LGC, GraphSAGE captures higher-order neighbor dependencies and layer dependencies to model complex graph structures. In return, CERec with a GraphSAGE-based GLM could harness complex dependencies in graphs to enhance the quality of explanation learning. Because of these two advantages, GraphSAGE generates superior graph embeddings, which in turn improves counterfactual explanation learning of CERec.

• Graph learning methods largely impact model explainability, as evidenced by the severely degraded explanation consistency of both variants in Table V. This suggests that explanations heavily rely on the quality of embeddings learned by the graph learning methods. The reason is that graph embeddings provide essential semantics for explanation learning, while inaccurate or false semantics can lead to varying or contradictory explanations for recommendations. For example, on Last–FM dataset, we observe that our CERec with a GraphSAGE-based GLM explains user u17724’s preference by the music type “Soul”, which aligns with the actual user preference recorded in the dataset. In contrast, CERec with a GCN-based GLM uses the music singer “Aloe Blacc” as the explanation, which is contradictory with u17724’s preference. We thus conclude that ensuring the reliability of graph embeddings through appropriate graph learning methods is crucial for maintaining high explanation consistency. GraphSAGE, with its ability to capture accurate semantics, proves to be a suitable choice for enhancing the learning of counterfactual explanations.

2) Impact of Attention Scores: The attention mechanism is pivotal in the counterfactual path sampler (CPS) to guide the search for counterfactual paths within the two-step path sampling. Hence, we investigate the effectiveness of the attention mechanism, i.e., how attention scores facilitate searching for informative counterfactual paths. In particular, two attention scores α1 and α2 are calculated respectively through the attention mechanism on source and target node embeddings. We thus do the ablation study by fixing one attention score as uniform probability while keeping the other score unchanged. We present the experimental results in Table VI, wherein the notation Unif(α1) signifies that α1 is sampled using a uniform probability U(0, 1); the same applies for Unif(α2). Note that we do not consider setting both α1 ∼ U(0, 1) and α2 ∼ U(0, 1), as this would lead to the zero gradients of our CPS. Analyzing Table VI, we observe that both Unif(α1) and Unif(α2) downgrade the recommendation and explainability performance of CERec. We thus conclude that both attention scores are important for our CERec to find appropriate counterfactual paths to better serve the explanation learning. By leveraging these attention scores, our CERec can effectively guide the search for informative counterfactual paths toward candidate counterfactual items, ultimately improving the quality of counterfactual explanations. Moreover, Unif(α2) exhibits relatively larger decrease percentages compared with Unif(α1). For instance, Unif(α1) downgrades CERec’s performance by a Recall@20 of 7.9% while Unif(α2) downgrades 25.0% Recall@20 on Yelp2018. This is reasonable as Unif(α2) aligns uniformly distributed attention scores at the second sampling step, which ignores the influence of α1 calculated during the first sampling step. This observation further highlights the superiority of applying the attention mechanism at each step of the path sampling.

3) Impact of Reward Functions: The reward function is the core of the reinforcement learning agent as it guides policy explorations. To verify the influence of reward functions, we consider two variants of our proposed counterfactual reward in (15). In particular, (15) incorporates two criteria: Rationality [16] and Similarity [53]; we employ only Rationality to build the variant R-reward, while using Similarity to construct the variant S-reward. Formally, R-reward is given by,

$$ r(s_t, a_t) = \begin{cases} 1, & \text{if } P_u(\epsilon_t) - P_u(\epsilon_{t+1}) \geq \epsilon \\ 0, & \text{otherwise} \end{cases} $$  

where \( r(s_t, a_t) \) is the reward calculated for action \( a_t \) at state \( s_t \). S-reward is defined as,

$$ r(s_t, a_t) = \cos(h_{s_t}, h_{s_{t+1}}) $$

The value after “+” or “−” indicates the decrease (or increase) percentage of the variant’s performance compared with the original CERec.

| Variants       | Top-K Recommendation | Explanation Consistency |
|---------------|----------------------|------------------------|
|               | Recall@20 | NDCG@20 | Precision% | Recall% | Amazon-book     |
| CERec-GCN     | 0.1519 (0.010) | 0.0796 (0.085) | 21.7364 (0.906) | 2.8739 (0.713) |
| CERec-GNN     | 0.1233 (5.90) | 0.0794 (0.753) | 21.6332 (9.22) | 2.7793 (0.103) |
| CERec-SGC     | 0.1585 (-3.90) | 0.0779 (-2.86) | 19.9846 (0.093) | 2.3837 (0.167) |
| CERec-LGC     | 0.1301 (-7.60) | 0.0752 (-0.63) | 19.4328 (-18.6) | 2.5062 (-17.6) |
| CERec-GAT     | 0.1394 (-0.80) | 0.0791 (-2.15) | 21.8229 (-6.85) | 2.9078 (-6.56) |
| CERec         | 0.1406 | 0.0803 | 23.8969 | 3.1020 |

| Variants       | Top-K Recommendation | Explanation Consistency |
|---------------|----------------------|------------------------|
|               | Recall@20 | NDCG@20 | Precision% | Recall% | Yelp2018       |
| Unif(α1)      | 0.1390 (-1.10) | 0.0790 (-1.64) | 21.8847 (-8.64) | 2.7903 (-10.05) |
| Unif(α2)      | 0.1193 (-15.10) | 0.0674 (-16.03) | 19.0109 (-20.46) | 2.4360 (-21.46) |
| CERec         | 0.1406 | 0.0803 | 23.8969 | 3.1020 |

**TABLE V**

**TABLE VI**
TABLE VII

| Variants | Top-K Recommendation | Explanation Consistency |
|----------|----------------------|-------------------------|
|          | Recall@20 | NDCG@20 | Precision% | Recall% |
| Amazon-book |                      |                        |
| R-reward | 0.1417 (0.7%) | 0.0802 (0.9%) | 15.8732 (35.5%) | 1.8533 (40.1%) |
| S-reward | 0.1225 (14.7%) | 0.0733 (9.5%) | 10.6173 (55.5%) | 0.9231 (66.3%) |
| CERec    | 0.2660 (60.3%) | 0.0803 (8.3%) | 23.8969 (9.2%) | 3.1020 |
| Yelp2018 |                      |                        |
| R-reward | 0.0631 (-2.8%) | 0.0419 (-18.1%) | 31.0579 (31.6%) | 29.7834 (28.3%) |
| S-reward | 0.0556 (-16.8%) | 0.0367 (-34.9%) | 20.0731 (55.8%) | 25.6865 (-38.1%) |
| CERec    | 0.0650 (64.9%) | 0.0495 (45.5013) | 41.5931 |

The results are shown in Table VII. We observe that using R-reward and S-reward degrades CERec’s performance, especially for the explanation consistency performance. This suggests that both R-reward and S-reward misguide policy learning to learn irrelevant counterfactual explanations that are not consistent with users’ preferences. This is reasonable as excluding either Rationality or Similarity would cause the learning process to deviate from the desired objectives of counterfactual explanations, resulting in inferior performance. In contrast, our proposed counterfactual reward considers both Rationality and Similarity, aligning with the definition of counterfactual explanation to guide accurate policy learning. Moreover, using S-reward leads to relatively more severe model degradation compared with using R-reward. S-reward only encourages the candidate counterfactual node to be similar to the target node, which, however, does not verify the Rationality, i.e., whether the candidate node crosses the decision boundary. We thus conclude that ensuring Rationality is essential for learning counterfactual explanations.

E. Parameter Analysis and Computation Costs (RQ4, RQ5)

We first study how the reinforcement depth $T$ in (16) affects the recommendation performance. We then investigate how the training epoch impacts the stability of our recommendation model. We also give the computation costs of reinforcement learning-based baselines and our method.

1) Impact of Reinforcement Depth: The reinforcement depth $T$ in (16) determines the search space, with $T = 1$ denoting that at most 1-hop neighbors of starting entities are visited as proposals for policy learning. We search the number of $T$ in {1, 2, 3, 4, 5} and report the recommendation performance of our model on both datasets in Fig. 4(a), (b), and (c). We have the following observations.

First, our CERec with $T = 4$ yields the best performance on the Amazon-book and Yelp2018 dataset, while $T = 2$ gives the best results on the Last-FM. The Amazon-book and Yelp2018 are presented with 0.048% and 0.057% density and are much sparser compared with the Last-FM with 0.26% density. That means more inactive users with few item interactions are recorded in the Amazon-book and Yelp2018. To achieve the optimal recommendation performance, the Amazon-book and Yelp2018 require larger reinforcement depth (i.e., $T = 4$) compared with the Last-FM (i.e., $T = 2$). This is because diverse counterfactual items are retrieved by increasing the reinforcement depth to help the recommendation model achieve optimal performance on the two datasets. This finding indicates that augmenting diverse counterfactual items could provide precise recommendations for inactive users to enhance the recommendation. Counterfactual items offer high-quality negative signals for user preferences, and thus could help filter out negative items when providing recommendations for inactive users. Second, increasing the reinforcement depth enhances the recommendation performance before reaching the peaks on both datasets. We attribute such consistent improvements to the improved diversity of counterfactual items. This is because higher-hop item neighbors naturally cover more items beyond those unexposed but are actually counterfactual ones than lower-hop neighbors. Third, after peaks, increasing reinforcement depth leads to downgraded performance. This is because performing too many counterfactual item explorations introduces less relevant items that may bias the recommendation results.

2) Reward Gradient: To evaluate the stability and robustness of our model, we plot the cumulative rewards while training our model on the three datasets in Fig. 5. Here are our observations.

First, our counterfactual explanation model gains stable cumulative rewards on both datasets along with training, i.e., the cumulative rewards increase as the epoch increases and finally reach stable states without suffering any drastic fluctuations. This indicates that our counterfactual explanation model enjoys stable training without losing reward gradients (e.g., gradients vanishing). This further verifies the rationality and robustness of our proposed model. Second, the rates of reward convergence are different across different datasets. For example, the cumulative rewards on Amazon-book and Yelp2018 start to increase drastically at the beginning and converge at around epoch 40, while the counterpart on Last-FM first converges slowly and then becomes stable at around epoch 120. This indicates that our model on Amazon-book and Yelp2018 can quickly reach stable states using a small number of
iterations; even Amazon-book and Yelp2018 are much more sparse than Last-FM. This is because we use counterfactual items during training, which contain negative user preference signals to assist the decision-making for inactive users. Besides, we explore higher-order connectivity among the CKG as side information for training, thus enhancing model robustness facing sparse datasets.

3) Computation Costs: We evaluate the running time of reinforcement learning-based baselines, including KGPolicy, KGQR, and KGAT, on the largest Amazon-book dataset. For the average case, the corresponding results are 232 s, 379 s, and 279 s per epoch, respectively. Our CERec algorithm runs in 284 s per epoch on average. Thus, CERec has a comparable cost to other RL-based baselines. Besides, our CERec runs in 273 s per epoch in the best case and 291 s per epoch in the worst case, which are also comparable costs to other RL-based baselines. We also observed that the training time of CERec increases by 42.78 s per epoch on Amazon-book after using the counterfactual explanation model. Overall, as our counterfactual explanation model delivers explanations with superior increased explainability, e.g., 23.8969 $F_1$ score on Amazon-book (cf. Table IV), we consider the increased cost by 42.78 s as a reasonable trade-off.

VIII. CONCLUSION

We propose CERec, a reinforcement learning-based counterfactual explainable recommendation framework over a CKG. Our CERec can generate attribute-based counterfactual explanations while providing precise recommendations. We design a counterfactual explanation model as a reinforcement learning agent to discover high-quality counterfactual explanations. The counterfactual explanation model takes paths sampled from our counterfactual path sampler as actions to optimize an explanation policy. By maximizing the counterfactual rewards of the deployed actions, the explanation policy is learned to generate high-quality counterfactual items. In addition, we reduce the vast action space by utilizing attention mechanisms in our path sampler to yield effective paths from the CKG. Finally, the learned explanation policy generates attribute-based counterfactual explanations for recommendations. We deploy the explanation policy to a recommendation model to enhance the recommendation. Extensive explainability and recommendation evaluations on three large-scale datasets demonstrate CERec’s abilities to improve the recommendation and provide counterfactual explanations consistent with user preferences.

REFERENCES

[1] Q. Li, X. Wang, Z. Wang, and G. Xu, “Be causal: De-biasing social network confounding in recommendation,” ACM Trans. Knowl. Discov. Data, vol. 17, pp. 1–23, 2022.
[2] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, “Leveraging meta-path based context for top-N recommendation with a neural co-attention model,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1531–1540.
[3] Y. Zhang et al., “Explainable recommendation: A survey and new perspectives,” Found. Trends Inf. Retrieval, vol. 14, no. 1, pp. 1–101, 2020.
[4] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, “Explainable reasoning over knowledge graphs for recommendation,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 5329–5336.
[5] L. Xie, Z. Hu, X. Cai, W. Zhang, and J. Chen, “Explainable recommendation based on knowledge graph and multi-objective optimization,” Complex Intell. Syst., vol. 7, no. 3, pp. 1241–1252, 2021.
[6] N. Froestl and G. E. Hinton, “Distilling a neural network into a soft decision tree,” in Proc. 1st Int. Workshop Comprehensibility Explanation AI ML Co-located 16th Int. Conf. Italian Assoc. Artif. Intell., 2017. [Online]. Available: http://ceur-ws.org/Vol-2071/CECIAIA_2017_paper_3.pdf
[7] E. Shulman and L. Wolf, “Meta decision trees for explainable recommendation systems,” in Proc. AAAI/ACM Conf. AI, Ethics, Soc., 2020, pp. 365–371.
[8] W. Lin, S. A. Alvarez, and C. Ruiz, “Collaborative recommendation via adaptive association rule mining,” Data Mining Knowl. Discov., vol. 6, no. 1, pp. 83–105, 2000.
[9] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, “Explicit factor models for explainable recommendation based on phrase-level sentiment analysis,” in Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2014, pp. 83–92.
[10] A. Ghazimatin, R. Saha Roy, and G. Weikum, “Fairy: A framework for understanding relationships between users’ actions and their social feeds,” in Proc. 12th ACM Int. Conf. Web Search Data Mining, 2019, pp. 240–248.
[11] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, and X. Xie, “A reinforcement-learning framework for explainable recommendation,” in Proc. IEEE Int. Conf. Data Mining, 2018, pp. 587–596.
[12] G. Peake and J. Wang, “Explanation mining: Post hoc interpretability of latent factor models for recommendation systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 2060–2069.
[13] Q. Xu and L. Narayanan R, “Model-agnostic versus model-intrinsic interpretability for explainable product search,” in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., 2021, pp. 5–15.
[14] J. Singh and A. Anand, “Posthoc interpretability of learning to rank models using secondary training data,” 2018, arXiv:1806.11330.
[15] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, “Counterfactual explainable recommendation,” in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., 2021, pp. 1784–1793.
[16] S. Verma, V. Boomsangong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah, “Counterfactual explanations and algorithmic recourses for machine learning: A review,” 2020, arXiv:2010.10596.
[17] J. Pearl, “Causal inference in statistics: An overview,” Statist. Surv., vol. 3, pp. 86–146, 2009.
[18] V. Kaffes, D. Sacharidis, and G. Giannopoulos, “Model-agnostic counterfactual explanations of recommendations,” in Proc. 29th ACM Conf. User Model. Adapt. Personalization, 2021, pp. 280–285.
[19] K. Xiong et al., “Counterfactual review-based recommendation,” in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., 2021, pp. 2231–2240.
[20] W. Cheng, Y. Shen, L. Huang, and Y. Zhu, “Incorporating interpretability into latent factor models via fast influence analysis,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 885–893.
[21] A. Ghazimatin, O. Babalou, R. Saha Roy, and G. Weikum, “Prince: Provider-side interpretability with counterfactual explanations in recommender systems,” in Proc. 13th Int. Conf. Web Search Data Mining, 2020, pp. 196–204.
Xiangmeng Wang received the MSc degree in computer application technology from Shanghai University. She is currently working toward the PhD degree with the School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). Her general research interests lie primarily in explainable artificial intelligence, data analysis, and causal machine learning. Her papers have been published in top-tier conferences and journals in the field of machine learning.

Qian Li is a lecturer with the School of Engineering, Computing and Mathematical Sciences (EECMS), Curtin University, Perth, Australia. Her general research interests lie primarily in optimization algorithms and causal machine learning. Her papers have been published in top-tier conferences and journals in the field of machine learning and computer vision.

Dianer Yu (Graduate Student Member, IEEE) received BSc and MSc degrees in computer science from UTS. He is currently working toward the PhD degree with the School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). His general research interests lie primarily in data mining, causal inference and explainable machine learning.

Qing Li (Fellow, IEEE) is a chair professor (Data Science) and the Head of the Department of Computing, the Hong Kong Polytechnic University. He has been actively involved in the research community by serving as an associate editor and reviewer for technical journals, and as an organizer/co-organizer of numerous international conferences. He is currently a IET/IEEE, a member of ACM-SIGMOD and IEEE Technical Committee on Data Engineering. He is the chairperson of the Hong Kong Web Society, and also serves/is serving as an executive committee (EXCO) member of IEEE-Hong Kong Computer Chapter and ACM Hong Kong Chapter. In addition, he serves as a councilor of the Database Society of Chinese Computer Federation (CCF), a member of the Big Data Expert Committee of CCF, and is a Steering Committee member of DASFAA, ER, ICWL, UMEDIA, and WISE Society.

Guandong Xu (Member, IEEE) received BSc and MSc degrees in computer science and engineering and the PhD degree in computer science. He is a professor in the School of Computer Science and Advanced Analytics Institute with the University of Technology Sydney. He currently heads the Data Science and Machine Intelligence Lab, which consists of more than 15 members of academics, research fellows and HDR students. From 2019, he directs the newly established Smart Future Research Centre, which is an across-disciplines industry engagement and innovation platform for AI and Data Science Application towards smart wealth management and investment, energy, food, water, living, and city.