GREASE: Generate Factual and Counterfactual Explanations for GNN-based Recommendations

Ziheng Chen[1][0000−0002−2585−637X], Fabrizio Silvestri[2][0000−0001−7669−9055], Jia Wang[3][0000−0002−3165−7051], Yongfeng Zhang[4][0000−0002−1243−1145], Zhenhua Huang[5][0000−0001−8659−4062], Hongshik Ahn[1][0000−0002−8924−6159], and Gabriele Tolomei[2][0000−0001−7471−6659]

[1] Stony Brook University, Stony Brook, NY 11794, USA
`ziheng.chen@stonybrook.edu`
[2] Sapienza University of Rome, Rome 00185, Italy
`{fabrizio.silvestri,gabriele.tolomei}@uniroma1.it`
[3] The Xi'an Jiaotong-Liverpool University, Suzhou 215000, China
`jia.wang02@xjtlu.edu.cn`
[4] Rutgers University, Piscataway, NJ 08854, USA
`yongfeng.zhang@rutgers.edu`
[5] South China Normal University, Guangzhou City 510631, China
`huangzhenhua@scnu.edu.cn`

Abstract. Recently, graph neural networks (GNNs) have been widely used to develop successful recommender systems. Although powerful, it is very difficult for a GNN-based recommender system to attach tangible explanations of why a specific item ends up in the list of suggestions for a given user. Indeed, explaining GNN-based recommendations is unique, and existing GNN explanation methods are inappropriate for two reasons. First, traditional GNN explanation methods are designed for node, edge, or graph classification tasks rather than ranking, as in recommender systems. Second, standard machine learning explanations are usually intended to support skilled decision-makers. Instead, recommendations are designed for *any* end-user, and thus their explanations should be provided in user-understandable ways.

In this work, we propose GREASE, a novel method for explaining the suggestions provided by *any* black-box GNN-based recommender system. Specifically, GREASE first trains a surrogate model on a target user-item pair and its *l*-hop neighborhood. Then, it generates both factual and counterfactual explanations by finding optimal adjacency matrix perturbations to capture the *sufficient* and *necessary* conditions for an item to be recommended, respectively. Experimental results conducted on real-world datasets demonstrate that GREASE can generate concise and effective explanations for popular GNN-based recommender models.

Keywords: Explainable Recommender Systems · Explainable Graph Neural Networks · Factual Explanations · Counterfactual Explanations.

1   Introduction

To alleviate the information overload caused by the tremendous amount of data that existing online services expose to end-users, many digital platforms rely on *recommender systems* to help their customers choose the best "content" via personalized suggestions. Indeed, answering questions like "*What smartphone should I buy from this e-commerce website?*" or "*What TV series should I watch on this streaming platform?*" has become unfeasible for the user without machine-based support.

A broad spectrum of techniques for developing successful recommender systems have been proposed in the literature: most notably, *content-based* [29], *collaborative filtering* [34,33], *collaborative reasoning* [7,6], and mixtures of those [5,55]. Regardless of the approach, recommender systems have benefited from the advancement of machine learning (ML): from classical ML models [19,31] to deep learning (DL) models based on more complex neural network (NN) architectures [11,14,16]. Amongst DL models, recommender systems based on graph neural networks (GNNs) have recently gained momentum [10,15,47]. As a matter of fact, GNNs have become a powerful technique to tackle several predictive modeling tasks in many cutting-edge domains [4,37,47].

One of the reasons why GNN-based recommender systems stand out from competitors is that they can intrinsically establish correlations amongst users, items, and associated features through information propagation on graphs derived from user-item interactions. Specifically, GNNs learn hidden representations (i.e., embeddings) of each node (user or item) and edge (user-item preference) by repeatedly aggregating neighborhood embeddings. This way, high-order information and multi-hop relevance among graph entities (nodes or edges) are encoded in the corresponding learned embeddings. Thanks to such a powerful modeling ability, GNNs have achieved outstanding performance and become state-of-the-art models for recommender systems [15,17,47,48].

Despite their effectiveness, GNN-based recommender systems lack inherent explainability, like many other DL models. In fact, the new challenge for any powerful recommender system would be to attach an *explanation* to each suggested item to clarify why *that* specific item has been recommended to a given user.

In the recent years, a large body of work on *eXplainable Artificial Intelligence* (XAI) have been proposed in the literature [45,44,28,18,20,25,38]. Some of these studies also attempt to make GNNs more interpretable, e.g., via subgraph explorations [54] or counterfactual explanations [24]. However, existing explanation methods for GNNs do not easily adapt to interpret GNN-based recommendations, mostly due to two reasons. First, traditional GNN explanation methods are designed for node, edge, or graph classification tasks. Instead, explaining recommendations would require to figuring out the rationale behind a whole ranking of items. Second, standard ML explanations are usually intended to support skilled decision-makers (e.g., a banker communicating why a customer has their mortgage refused by the automatic credit-risk classifier). On the other hand, recommendations are designed for *any* end-user, and thus their explanations should be provided in user-understandable ways. These challenges make

explanation methods for GNN-based recommender systems unique, and therefore we need specific approaches to address them.

In response to this demand, we propose a **GNN-based REcommendAtion System Explainer** (GREASE). GREASE operates as follows. First, it trains a surrogate model of a fully trained GNN on a target user-item pair and its l-hop neighborhood. Then, it generates both factual and counterfactual explanations by finding optimal adjacency matrix perturbations to capture the *sufficient* and *necessary* conditions for an item to be recommended, respectively. Intuitively, the sufficient (i.e., factual) conditions for an item to be recommended to a user are the set of user-item interactions in the l-hop neighborhood that allows that item to be included in the final list of top-k recommended items. An example of a factual explanation could be the following: "*You have been recommended item* X *because you have interacted with* Y". On the other hand, the necessary (i.e., counterfactual) conditions are the set of user-item interactions in the l-hop neighborhood without which the item will *not* be included in the final list of top-k recommended items. An example of a counterfactual explanation could be the following: "*If you had not interacted with* Y *then you would not have recommended item* X".

Overall, the main contributions of our work are as follows:

(i) We develop GREASE, a method for generating explanations for recommendations provided by *any* GNN-based recommender system.
(ii) GREASE operates in a "black-box" setting, i.e., it needs only to access the output predictions of the GNN-based recommender system to explain.
(iii) GREASE can generate both factual and counterfactual explanations.
(iv) We validate GREASE on two datasets (*LastFM* and *Yelp*) in combination with two famous GNN-based recommender systems (LightGCN and NGCF); results demonstrate the superiority of our method against baseline approaches.

The remainder of this paper is organized as follows. In Section 2, we review relevant contributions to our work proposed in the literature. We provide some background concepts in Section 3 and we formulate our problem in Section 4. In Section 5, we describe our proposed method GREASE, which we validate through experiments in Section 6. Finally, we conclude our work in Section 7.

2   Related Work

The set of relevant work to this paper cover two main areas: *(i) GNN-based recommender systems* and *(ii) explanation methods for GNNs*. Below, we review the most significant contributions in these areas.

GNN-based recommender systems. In the past few years, many works on GNN-based recommendation have been proposed. Most of these contributions apply GNN to the original user-item bipartite graph directly, with a significant impact on the effectiveness and efficiency [8,15,21,39,47,49]. Multi-GCCF [40] and DGCF [23] propose to address the first issue by adding artificial edges between two-hop neighbors on the original graph to obtain the user-user and item-item

graph. In this way, the proximity information among users and items can be explicitly incorporated into user-item interactions.

Existing GNN-based recommender systems differentiate from each other also for the node aggregation and update strategies used. For example, NGCF [47] employs element-wise product to augment the items' features the user cares about or the users' preferences for item features. NIA-GCN [39] proposes the pairwise neighborhood aggregation approach to explicitly capture the interactions among neighbors. Inspired by the GraphSAGE [12], some works [21,40,52] adopt concatenation function with nonlinear transformation to update node and neighborhood representations. LightGCN [15] and LR-GCCF [8] simplify the update operation by removing the non-linearities, thereby retaining or even improving performance and increasing computational efficiency.

The aggregation and update operations are applied layer by layer to generate the representations of nodes for each depth of a GNN. Thus, the overall representations of users and items are required for the downstream prediction task. A mainstream approach is to use the node vector in the last layer as the final representation [21,52]. Recent studies, however, employ different methods to integrate messages from different layers.

A complete survey on GNN-based recommender systems can be found in [50].
Explanation methods for GNNs. Several GNN XAI approaches have been proposed, and a recent survey of the most relevant work is presented in [53].

The majority of existing methods provide a *factual* explanation in the form of a subgraph of the original graph that is deemed to be important for the prediction [3,9,22,27,30,36,46,51,54]. Such methods are analogous to popular XAI methods like LIME [32] or SHAP [26], which identify relevant features for a particular prediction for tabular, image, or text data. All of these methods require the user to specify the size of the explanation in advance, i.e., the number of features (or edges) to keep. More recently, Lucic et al. [24] have introduced CF-GNNExplainer. This is the first method to extract *counterfactual* explanations from GNN model predictions. It does so by automatically finding the optimal perturbation of the input graph that results in a prediction switch.

Regardless of the approach, however, existing explanation methods for GNNs are designed for node, edge, or graph classification tasks. Instead, explaining GNN-based recommendations would require to figuring out the rationale behind a whole ranking of items, rather than a single prediction. Combining the work of Tan et al. [42,41], our method (GREASE) is different from other existing GNN XAI solutions as it provides a general framework to extract *both* factual and conterfactual explanations for GNN-based recommender systems.

3   Background and Notation

We consider the standard collaborative filtering recommendation task. Specifically, let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ be a set of m users, and $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of n items. We represent the interactions between users and items with a binary user-item matrix $Y \in \{0, 1\}^{m \times n}$, where $Y_{u,i} = y_{u,i} = 1$ indicates that user

u has interacted with item i, or 0 otherwise. Notice that this resembles the case where user's preference on items is expressed via implicit feedback (e.g., clicks/likes) rather than explicit relevance judgements (e.g., ratings). Either way, the recommendation problem can be formulated as estimating the value $\hat{y}_{u,i}$ for each $u \in \mathcal{U}$ and each $i \in \mathcal{I}$ as follows:

$$\hat{y}_{u,i} = f(\boldsymbol{h}_u, \boldsymbol{h}_i),$$

where $\boldsymbol{h}_u, \boldsymbol{h}_i \in \mathbb{R}^d$ are suitable user and item representations, respectively, and $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto [0,1]$ is a function that measures the preference score for user u on item i.[6] The predicted score computed with f is in turn used to rank items to populate a list $\mathcal{I}_u^k \subseteq \mathcal{I}$ of the top-k recommended items for user u, i.e., the list of k items most likely relevant to user u according to f.

   In this work, we focus on GNN-based recommender systems, namely we assume that both user and item latent representations (i.e., the embeddings $\boldsymbol{h}_u$ and $\boldsymbol{h}_i$) are learned through a GNN g. Concretely, we consider the bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathcal{E})$ obtained from user-item interactions represented by Y, where $\mathcal{U}$ and $\mathcal{I}$ correspond to two disjoint and independent sets of nodes and $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{I}$ is the set of edges. There exists an edge connecting a node $u \in \mathcal{U}$ to another node $i \in \mathcal{I}$ iff user u interacts with item i. Thus, the topology of $\mathcal{G}$ is encoded by its adjacency matrix $A \in \{0,1\}^{m \times n}$, where $A_{u,i} = y_{u,i}$, i.e., $A_{u,i} = 1$ iff $(u,i) \in \mathcal{E}$. We refer to the overall set of nodes of $\mathcal{G}$ as $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$. Moreover, each node $v \in \mathcal{V}$ can be also associated with a p-dimensional real-valued feature vector, which represents one row of the feature matrix $X \in \mathbb{R}^{(m+n) \times p}$.

   The embedding of each node is learned through g by iteratively updating the node's features based on the neighbors' features. Formally, let $\boldsymbol{h}_v^l$ denote the embedding of the generic node $v \in \mathcal{V}$ at the l-th layer of g. Hence,

$$\boldsymbol{h}_v^l = g(A_v^l, X_v^l; \boldsymbol{\theta}_g) = \phi(\boldsymbol{h}_v^{l-1}, \psi(\{\boldsymbol{h}_w^{l-1} \mid w \in \mathcal{N}(v)\})),$$

where: $\mathcal{N}(v) = \{w \in \mathcal{V} \mid (v,w) \in \mathcal{E}\}$ is the 1-hop neighbors of v, i.e., the set of all nodes that are adjacent to v; A_v^l and X_v^l are the adjacency matrix and the feature matrix of nodes in the subgraph $\mathcal{G}_v^l$ of $\mathcal{G}$, induced by v and its l-hop neighbors; ϕ and ψ are arbitrary differentiable functions (i.e., neural networks): ψ is a permutation-invariant operator that *aggregates* the information from the l-hop neighborhood of v; ϕ *updates* the node embedding of v by combining information from the previous layer; $\boldsymbol{\theta}_g$ are the trainable parameters of g. As it turns out, the number of hidden layers in g determines the set of neighbors included while learning each node's embedding. Finally, let $\boldsymbol{h}_v^l = g(A_v^l, X_v^l; \boldsymbol{\theta}_g)$ be the embedding of the generic node $v \in \mathcal{V}$. Thus, we assume that the predicted rating score of user u for the item i is approximated with f, defined as follows:

$$f(\boldsymbol{h}_u^l, \boldsymbol{h}_i^l; \boldsymbol{\theta}_f) = f(g(A_{u,i}^l, X_u^l; \boldsymbol{\theta}_g), g(A_{u,i}^l, X_i^l; \boldsymbol{\theta}_g); \boldsymbol{\theta}_f),$$

where we denote by $A_{u,i}^l$ the adjacency matrix of the subgraph $\mathcal{G}_{u,i}^l = \mathcal{G}_u^l + \mathcal{G}_i^l$ induced by the union of the l-hop neighborhood of both user u and item i. From

---

[6] A similar reasoning would apply if we instead considered explicit ratings, i.e., $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$.

now on, however, we will consider l fixed and omit the corresponding superscript, unless otherwise needed. Notice that f can be any function parameterized by $\boldsymbol{\theta}_f$ that takes as input two node embeddings learned by g and outputs the corresponding preference score, e.g., f can be itself another neural network like a multilayer perceptron. Otherwise, f can simply compute the dot product or the cosine similarity between the two input node embeddings.

4   Problem Formulation

Without loss of generality, we consider a well-trained GNN-based recommender system f, which computes the preference score of a user u on an item i as the dot product of the node embeddings learned through an l-layer GNN g, namely:

$$\hat{y}_{u,i} = f(g(A_{u,i}, X_u), g(A_{u,i}, X_i)) = f(\boldsymbol{h}_u, \boldsymbol{h}_i) = \boldsymbol{h}_u^T \boldsymbol{h}_i.$$

Let $\mathcal{I}_u^k$ be the list of top-k recommended items for the user u. Informally, our goal is to attach an *explanation* to every item $i \in \mathcal{I}_u^k$ that clarifies why i has been recommended to u.

To achieve this goal, we envision two methods that derive from two opposite yet complementary approaches: *factual* (FA) and *counterfactual* (CF) reasoning. Specifically, factual reasoning seeks a subgraph $\mathcal{G}^{FA}$ from the user-item interaction graph $\mathcal{G}_{u,i}$ whose information is *sufficient* to include item i into user u's top-k list $\mathcal{I}_u^k$. On the other hand, counterfactual reasoning targets on a subgraph $\mathcal{G}^{CF}$ whose information is *necessary*, i.e., without which item i would *not* be in the top-k list of recommendations $\mathcal{I}_u^k$.

Extracting two subgraphs $\mathcal{G}^{FA}$ and $\mathcal{G}^{CF}$ from $\mathcal{G}_{u,i}$ is equivalent to find two adjacency matrices A^{FA} and A^{CF} in place of the original $A_{u,i}$. Hence, the goal of factual and counterfactual reasoning boils down to learning two edge mask matrices P^{FA} and P^{CF}, such that $A^{FA} = A_{u,i} \odot P^{FA}$ and $A^{CF} = A_{u,i} \odot P^{CF}$, where $\odot$ indicates the Hadamard (i.e., element-wise) matrix product.

Formally, for factual explanations we perturb the l-hop neighborhood of both u and i, so that $\boldsymbol{h}_u \rightsquigarrow \boldsymbol{h}_u^{FA}$ and $\boldsymbol{h}_i \rightsquigarrow \boldsymbol{h}_i^{FA}$, where:

$$\boldsymbol{h}_u^{FA} = g(A^{FA}, X_u), \quad \boldsymbol{h}_i^{FA} = g(A^{FA}, X_i),$$

and $A^{FA} = A_{u,i} \odot P^{FA}$. Eventually, this will lead to a new prediction score $\hat{y}_{u,i}^{FA} = f(\boldsymbol{h}_u^{FA}, \boldsymbol{h}_i^{FA})$. In fact, notice that, because of the neighborhood aggregation operations performed by the GNN g, all the embeddings of the nodes in $\mathcal{G}_{u,i}$ will change if $A_{u,i}$ is modified. Therefore, in general, we may obtain a new prediction score $\hat{y}_{u,j}^{FA}$ for every $j \in \mathcal{I}_u^k$ that was also in the l-hop neighborhood.

However, instead of directly optimizing the rank of the target item i we want to explain, factual reasoning seeks to maximize its predicted rating score $\hat{y}_{u,i}^{FA}$ until i is recommended, i.e., until i is pushed in the list of top-k recommended items $\mathcal{I}_u^k$, regardless of its rank. Thus, we define the following factual explanation loss function for a fixed user-item pair (u,i):

$$\mathcal{L}_{FA}(P^{FA}) = -\hat{y}_{u,i}^{FA} \cdot \mathbb{1}_{\mathcal{I}_u^k}(i), \tag{1}$$

where $\mathbb{1}$ is the well-known 0-1 indicator function: $\mathbb{1}_{\mathcal{I}_u^k}(i)$ will be 1 if $\mathcal{I}_u^k$ contains item i, 0 otherwise. $\mathcal{L}_{FA}$ is minimized when item i appears in the list of top-k recommendations $\mathcal{I}_u^k$. Similarly, for counterfactual explanations, we perturb the l-hop neighborhood of both u and i, so that $\boldsymbol{h}_u \rightsquigarrow \boldsymbol{h}_u^{CF}$ and $\boldsymbol{h}_i \rightsquigarrow \boldsymbol{h}_i^{CF}$, where:

$$\boldsymbol{h}_u^{CF} = g(A^{CF}, X_u), \quad \boldsymbol{h}_i^{CF} = g(A^{CF}, X_i),$$

and $A^{CF} = A_{u,i} \odot P^{CF}$. As opposed to factual reasoning, though, the goal of counterfactual explanations is to directly minimize the new prediction score $\hat{y}_{u,i}^{CF} = f(\boldsymbol{h}_u^{CF}, \boldsymbol{h}_i^{CF})$ until i is *no longer* recommended, i.e., to evict item i from the list of top-k recommended items $\mathcal{I}_u^k$. Formally, we define the counterfactual loss function for a fixed user-item pair (u, i) as follows:

$$\mathcal{L}_{CF}(P^{CF}) = \hat{y}_{u,i}^{CF} \cdot \mathbb{1}_{I_u^k}(i), \tag{2}$$

where $\mathbb{1}$ is the well-known 0-1 indicator function, and $\mathcal{L}_{CF}$ is minimized when item i is removed from the list of top-k recommendations $\mathcal{I}_u^k$.

Notice that both factual and counterfactual loss incorporate a 0-1 indicator function term, which is non-convex and non-smooth, thus hard to optimize using gradient-based methods. In practice, we replace $\mathbb{1}_{I_u^k}(i)$ with the following term:

$$\text{ReLU}\Big(\hat{y}_{u,i} - \min\big\{\bigcup_{j \in I_u^k} \hat{y}_{u,j}\big\} + \varepsilon\Big) = \max\Big(0, \hat{y}_{u,i} - \min\big\{\bigcup_{j \in I_u^k} \hat{y}_{u,j}\big\} + \varepsilon\Big),$$

which evaluates to 0 for any item i whose predicted preference score is smaller than that of the last ranked item in the top-k list, or is greater than 0 otherwise. The small tolerance $\varepsilon > 0$ guarantees that the input of the ReLU function is positive until considering the last k-th item of the list, negative otherwise. In this work, we set $\varepsilon = 0.05$.

Furthermore, we want the selected subgraphs $\mathcal{G}^{FA}$ and $\mathcal{G}^{CF}$ to be as sparse as possible. Hence, we generate factual and counterfactual explanations by minimizing a joint loss function of the form:

$$\mathcal{L}(P) = \mathcal{L}_{\exp}(P) + \beta \mathcal{L}_{\text{dist}}(P), \tag{3}$$

where P is a generic perturbation matrix (i.e., either P^{FA} or P^{CF}) and $\mathcal{L}_{\exp}$ works as a placeholder for explanation loss, which could be substituted by $\mathcal{L}_{FA}$ or $\mathcal{L}_{CF}$ for generating factual and counterfactual explanations, respectively. Also, $\mathcal{L}_{\text{dist}}$ measures the distance between the perturbed matrix $A_{u,i} \odot P$ and the original adjacency matrix A. For example, $\mathcal{L}_{\text{dist}}(P) = ||A_{u,i} \odot P - A_{u,i}||_p$ computes the L^p-norm of the difference between the two matrices. In this work, we set $\mathcal{L}_{\text{dist}}(P) = ||A_{u,i} \odot P - A_{u,i}||_1$, i.e., the L^1-norm.

Our problem can be thus formulated as the following optimization task:

$$P^* = \underset{P}{\arg\min} \, \mathcal{L}(P). \tag{4}$$

Eventually, the best factual and counterfactual explanation is found by computing $A^{FA^*} = A_{u,i} \odot P^{FA^*}$ and $A^{CF^*} = A_{u,i} \odot P^{CF^*}$, respectively.

5   Proposed Method: GREASE

We propose GREASE to solve the objective defined in Equation 4. Theoretically, to find the optimal perturbation matrix P^*, we should take the gradient w.r.t. P of the joint loss function $\mathcal{L}(P)$ defined in Equation 3. However, since GREASE operates in a black-box setting, the gradient of the model g (and therefore f) is unknown. This assumption is restrictive yet pretty standard, especially in a commercial scenario, where the recommender system's internals (e.g., the model's architecture, parameters, or gradient) are rarely disclosed. Therefore, we first learn a local approximator $\widetilde{g}$. This surrogate model tries to resemble g in learning the embeddings of each node in the subgraph $\mathcal{G}_{u,i}^l = \mathcal{G}_u^l + \mathcal{G}_i^l$ induced by the union of the l-hop neighborhood of both user u and item i. The intuition is that the aggregation mechanism behind node embeddings is complex over the complete network. But in the local neighborhood of a target user-item pair, such aggregation process can be relatively simple, hence amenable to be extracted by a simpler model. Then, in our objective function, we replace g with a surrogate approximation of it $\widetilde{g}$, and find the optimal P via gradient-based optimization.

5.1   Train a Surrogate GNN Model

Given a trained black-box GNN model g, a user-item pair (u, i), and the subgraph $\mathcal{G}_{u,i}^l$ induced by the l-hop neighborhood of u and i, we first learn a surrogate model $\widetilde{g} \approx g$ to approximate the embedding of each node in $\mathcal{G}_{u,i}^l$. We can measure the surrogate loss function via standard mean squared error:

$$\mathcal{L}_{\text{sur}}(g, g') = \sum_{k=1}^{|\mathcal{V}_{u,i}^l|} ||g(A_{u,i}^l, X_k^l) - g'(A_{u,i}^l, X_k^l)||^2,$$

where $\mathcal{V}_{u,i}^l$, X_k^l, $A_{u,i}^l$ are the set of nodes, feature and adjacency matrix of $\mathcal{G}_{u,i}^l$. Both g and g' output a d-dimensional embedding vector, and the optimal approximate model $\widetilde{g}$ is found by minimizing the surrogate loss function above:

$$\widetilde{g} = \arg\min_{g'} \mathcal{L}_{\text{sur}}(g, g').$$

Considering we aim to extract the linkage information from a heterogeneous network with two types of nodes (i.e., users and items), we adopt a Relational Graph Convolutional Network (R-GCN) [43][35] to learn the surrogate model. R-GCN extends GCN to heterogeneous graphs by considering the direction and type of edges, separately. To allow R-GCN to pass messages through undirected edges linking users and items, we transform our original bipartite graph $\mathcal{G}$ by substituting each edge pair $(u, i) \in \mathcal{E}$ with two pairs of symmetric edges (u, i) and (i, u), respectively. Thus, the user embedding $\boldsymbol{h}_u$ and the item embedding $\boldsymbol{h}_i$ will be propagated with different weights in the generic l-th layer:

$$\boldsymbol{h}_u^l = \sigma \left(\sum_{i \in \mathcal{N}(u)} \frac{1}{\mathcal{N}(u)} W_{(i,u)}^{(l-1)} \boldsymbol{h}_i^{(l-1)} + W_0^{(l-1)} \boldsymbol{h}_u^{(l-1)} \right)$$

$$\boldsymbol{h}_i^l = \sigma \left(\sum_{u \in \mathcal{N}(i)} \frac{1}{\mathcal{N}(i)} W_{(u,i)}^{(l-1)} \boldsymbol{h}_u^{(l-1)} + W_0^{(l-1)} \boldsymbol{h}_i^{(l)} \right)$$

In this way, the weight matrices $W_{(i,u)}^{(l-1)}$ and $W_{(u,i)}^{(l-1)}$ are leveraged to update the users' and items' information, separately.

5.2   Generate Explanations

After we find the best surrogate GNN model $\widetilde{g}$, we can solve the optimization task defined in Equation 4. We let $\widetilde{\mathcal{L}}(P)$ denote the same loss function as the one defined in Equation 4 yet depending on $\widetilde{g}$ rather than the original GNN g.

In Algorithm 1, we describe how GREASE generates both factual *or* counterfactual explanations for a given recommended item i to a user u. The algorithm returns the adjacency matrix (i.e., the subgraph) corresponding to the factual or counterfactual explanations (i.e., A^{FA} or A^{CF}, respectively), or $\perp$ if no valid explanation is found. To achieve that, we consider the two types of explanations, separately. First, we create two $(m+n) \times (m+n)$ matrices of all ones: J and $\hat{J}$, respectively (line 4). Then, depending on the value of the *exp* flag, we perform a different sequence of initialization steps. Specifically, if we want to extract *factual* explanations (i.e., $exp = \texttt{FA}$), we need to start by evicting all the existing edges associated with user u and item i. Concretely, we nullify the u-th and i-th column of the matrix $\hat{J}$, and we assign it to the initial perturbation matrix P. Also, we instantiate the explanation loss to the factual explanation loss defined in Equation 1 (lines 5–8). Instead, if we want to extract *counterfactual* explanations (i.e., $exp = \texttt{CF}$), we simply initialize the perturbation matrix P with the matrix of all ones J, so that at the beginning we to retain all the edges of the original graph. In addition, we set the explanation loss to the counterfactual explanation loss of Equation 2 (lines 9–11). Still, as part of the common initialization procedure, we set the current explanation adjacency matrix A^0 to the original adjacency matrix associated with $\mathcal{G}_{u,i}^l$ (line 12). Furthermore, to transform our optimization problem from discrete to continuous, we use a smoothed, real-valued perturbation matrix $\hat{P}^1$, initially set to P. At the beginning, the optimal perturbation matrix is undefined ($\perp$) and the minimal distance between the original and the perturbed adjacency matrices is set to infinity (lines 12–14).

The core iterative procedure of GREASE is encapsulated in the for loop between lines 15 and 29, which is repeated M times. At each iteration j, the first step is to binarize the current real-valued perturbation matrix. To do so, every element of $\hat{P}^j$ is passed through a sigmoid filter σ that squashes them into a real value in the range $[0, 1]$. Then, we further discretize the matrix entries using a threshold equal to 0.5, i.e., if the entry is greater than or equal to 0.5 it is mapped to 1, or 0 otherwise (line 16). Once the perturbation matrix is binarized, with all its entries being 0 or 1, we compute the Hadamard product between the original adjacency matrix and the current perturbation, thereby originating a new explanation matrix A^j for this iteration (line 17). We thus compute and store the updated L^1-norm of the distance between the original and the current

---

Algorithm 1 GREASE

1: **Input:** Surrogate GNN model $\widetilde{g}$; user u; item i; feature matrix X; adjacency matrix $A_{u,i}^l$ of the subgraph $\mathcal{G}_{u,i}^l$ induced by the union of the l-hop neighborhoods of u and i; number of iterations M; trade-off β; learning rate η; size k of the list of recommended items $\mathcal{I}_u^k$; explanation flag exp (`FA`=factual; `CF`=counterfactual)

2: **Output:** The factual/counterfactual explanation matrix A^* associated with the optimal perturbation matrix P^*, or $\perp$ if no valid explanation is found.

3: **procedure** GREASE($\widetilde{g}, u, i, X, A_{u,i}^l, M, \beta, \eta, k, exp$):
4:      $\hat{J} = J$;
5:      **if** exp == `FA` **then**
6:          $\hat{J}[:, u] = \mathbf{0}$; $\hat{J}[:, i] = \mathbf{0}$;
7:          $P = \hat{J}$; $\mathcal{L}_{\exp}(P) = \mathcal{L}_{FA}(P)$;
8:      **end if**
9:      **if** exp == `CF` **then**
10:          $P = J$; $\mathcal{L}_{\exp}(P) = \mathcal{L}_{CF}(P)$;
11:      **end if**
12:      $A^0 = A_{u,i}^l \odot J$;
13:      $\hat{P}^1 = P$; $P^* = \perp$;
14:      $min\_loss\_dist = \infty$;
15:      **for** $j \in [1 \dots M]$ **do**
16:          $P^j = \sigma(\hat{P}^j, 0.5)$;
17:          $A^j = A_{u,i}^l \odot P^j$;
18:          $\mathcal{L}_{\text{dist}}(\hat{P}^j) = ||A^j - A^{j-1}||_1$;
19:          $curr\_loss\_dist = \mathcal{L}_{\text{dist}}(\hat{P}^j)$;
20:          $r = f(\widetilde{g}(A^j, X_u), \widetilde{g}(A^j, X_i))$;
21:          **if** (exp == `FA` **and** $r \leq k$) **or** (exp == `CF` **and** $r > k$) **then**
22:              **if** $curr\_loss\_dist < min\_loss\_dist$ **then**
23:                  $P^* = P^j$;
24:                  $min\_loss\_dist = curr\_loss\_dist$;
25:              **end if**
26:          **end if**
27:          $\widetilde{\mathcal{L}}(\hat{P}^j) = \mathcal{L}_{\exp}(\hat{P}^j) + \beta \mathcal{L}_{\text{dist}}(\hat{P}^j)$;
28:          $\hat{P}^{j+1} = \hat{P}^j - \eta \nabla_{\hat{P}^j} \widetilde{\mathcal{L}}(\hat{P}^j)$;
29:      **end for**
30:      **if** $P^* \neq \perp$ **then**
31:          **return** $A^* = A_{u,i}^l \odot P^*$;
32:      **else**
33:          **return** $\perp$;
34:      **end if**
35: **end procedure**

---

explanation matrix (lines 18–19). Afterwards, we compute the rank r of item i for user u according to our recommender system f based on the surrogate GNN model $\widetilde{g}$ and the currently perturbed adjacency matrix A^j (line 20). We therefore check if the explanation goal is met. Specifically, for factual explanations, the computed rank must be high enough to let item i be added to the top-k

recommendations. Instead, for counterfactual explanations, the computed rank must be low enough to let item i be removed from the top-k recommendations (line 21). In the case the explanation goal is met, we must check whether the current perturbation matrix is closer to the original adjacency matrix than the optimal perturbation computed so far. If so, we must then update the optimal perturbation accordingly and keep track of the new best distance (lines 21–26). Nevertheless, we update the overall loss function as defined in Equation 3 and take one step of gradient descent to update the perturbation matrix $\hat{P}^{j+1}$, which will be considered at the next iteration $j+1$ (lines 27–28).

Eventually, if the optimal perturbation matrix is changed during the M iterations, GREASE will return the final explanation matrix, i.e., either A^{FA^*} or A^{CF^*} (lines 30–31). Otherwise, no valid explanation is found for the given input user-item pair (line 33).

6   Experiments

In this section, we describe the experiments conducted to validate the ability of GREASE to explain suggestions provided by GNN-based recommender systems. **Datasets.** Before we can apply GREASE, we train our GNN-based recommender systems on the following two datasets: *LastFM* [1] and *Yelp* [2]. An overview of the main properties of these data collections is shown in Table 1.

Dataset	N. of Users	N. of Items	N. of Interactions
LastFM	1,801	7,432	76,693
Yelp	31,668	38,048	1,561,406

Table 1: Main characteristics of the public datasets used.

GNN-based Recommender Systems. For both datasets, we randomly select 80% of historical interactions of each user to build the training set, and treat the remaining as the test set. Thus, we train two different GNN-based recommender systems on each dataset, which in turn we will explain with GREASE: LightGCN [15] and NGCF [47]. We set the number of layers of the GNN to 3, namely the embeddings are learned from the 3-hop neighborhood of each node. In this way, each user's embedding will be influenced also by the items rated by other users, and each item's embedding will be influenced also by the users who have rated other items. Both LightGCN and NGCF are trained by minimizing binary cross-entropy loss via Adam optimizer with learning rate equal to 0.001. Eventually, both GNN-based recommender systems use the dot product between the node embeddings learned either by LightGCN or NGCF to compute the preference score of each user-item pair.

Parameter Settings. For each user u in every dataset, we first generate the top-k list or recommended items $\mathcal{I}_u^k$ according to the GNN-based recommender systems considered (i.e., LightGCN and NGCF), where k is selected from $\{10, 15, 20\}$. Then, we consider every item $i \in \mathcal{I}_u^k$ and let GREASE generate a factual and a counterfactual explanation, separately. It is worth remarking that GREASE

operates on the subgraph $\mathcal{G}^l_{u,i}$ induced by the l-hop neighborhood of u and i to generate such explanations. In this work, we set $l = 2$ for all settings. Besides, a two-layer R-GCN is used for training the surrogate models $\widetilde{g}$ associated with the original GNN models, i.e., LightGCN and NGCF. The hidden dimensions are set to 32 for *LastFM* and 64 for *Yelp* datasets, respectively. In addition, we set $M = 200$ and $\eta = 0.01$ for the number of iterations and the learning rate of GREASE (see Algorithm 1). The trade-off β between explanation ($\mathcal{L}_{\text{exp}}$) and distance loss ($\mathcal{L}_{\text{dist}}$) is set to $1/200$ both for *LastFM* and *Yelp*.

Evaluation Metrics. We evaluate the quality of generated explanations according to the following three metrics: *Probability of Sufficiency* (PS), *Probability of Necessity* (PN) and *Explanation Cost* (EC). PS measures the percentage of factual explanations that successfully push the item i into the u's top-k list. Similarly, PN measures the percentage of counterfactual explanations that evict i from u's top-k list. For both PS and PN, the higher the better. EC, instead, indicates the number of edges that needs to be added or deleted according to the factual explanation and counterfactual explanation, respectively. To deal with the huge number of user-item pairs to be explained, we randomly sample 10% users in each setting, and explain all the items in their top-k list of recommendations ($k \in \{10, 15, 20\}$). Moreover, experiments were repeated five times (i.e., five different samples of users are randomly selected) and results are expressed as the mean $\pm$ standard deviation.

Baselines. To evaluate our algorithm, we compare it against two baselines: RANDOM and PERSONALRANK. The former is meant as a sanity check, as it randomly adds/deletes edges from $\mathcal{G}^l_{u,i}$ to generate factual/counterfactual explanations. In PERSONALRANK, instead, we start by modifying all the existing edges associated with nodes that have the largest PersonalRank [13] score until we possibly get either a factual or a counterfactual explanation.[7]

To control the complexity (i.e., EC), we limit the maximum cost for constructing factual and counterfactual explanations. Specifically, after testing with several combination of values, we allow at most 6 and 10 edges to be added/removed, respectively, as these lead GREASE to the best results.

Results. In Table 2, 3, and 4, we compare the quality of explanations generated by our method (GREASE) with those of the two baselines (RANDOM and PERSONALRANK), when $k = 10$, 15, and 20, respectively. We can observe that GREASE consistently outperforms other competitors on all the three metrics. Specifically, compared with PERSONALRANK, GREASE improves the validity of factual explanations for the recommendations generated on *LastFM* by up to 15% (+12% on *Yelp*). Moreover, the validity of counterfactual explanations on *LastFM* is 21% higher than those provided by PERSONALRANK (+20% on *Yelp*).

7   Conclusion and Future Work

In this work, we presented GREASE, an explanation method for GNN-based recommendations. GREASE operates in a fully "black-box" setting, i.e., it needs

---

[7] PersonalRank is the personalized version of the well-known PageRank algorithm.

			Top-10 recommendations	
Dataset	Recommendation Model	Explanation Method	**PN(↑)/EC(↓)**	**PS(↑)/EC(↓)**
LastFM	LightGCN	GREASE	**0.89**(±3.00)/**3.81**(±0.27)	**0.69**(±3.00)/**6.21**(±0.76)
		PERSONALRANK	0.79(±2.00)/4.63(±2.50)	0.58(±2.00)/6.68(±0.36)
		RANDOM	0.19(±3.00)/5.00(±1.20)	0.11(±4.00)/8.50(±0.41)
	NGCF	GREASE	**0.90**(±6.00)/**3.23**(±0.33)	**0.70**(±6.00)/**6.30**(±0.82)
		PERSONALRANK	0.79(±2.00)/4.63(±2.50)	0.58(±2.00)/6.68(±0.36)
		RANDOM	0.19(±3.00)/5.00(±1.20)	0.11(±4.00)/8.50(±0.41)
Yelp	LightGCN	GREASE	**0.91**(±4.00)/**3.26**(±0.31)	**0.70**(±3.37)/**6.37**(±0.38)
		PERSONALRANK	0.82(±3.50)/4.79(±0.36)	0.61(±2.95)/6.93(±0.41)
		RANDOM	0.21(±5.30)/5.39(±1.32)	0.16(±2.80)/7.35(±0.13)
	NGCF	GREASE	**0.93**(±6.00)/**5.70**(±0.45)	**0.71**(±5.00)/**6.21**(±0.82)
		PERSONALRANK	0.83(±3.00)/6.22(±0.41)	0.59(±3.00)/7.21(±0.38)
		RANDOM	0.22(±3.00)/8.20(±1.47)	0.15(±4.00)/8.60(±0.46)

Table 2: PN, PS, and EC of factual and counterfactual explanations generated by GREASE, PERSONALRANK, and RANDOM for top-10 recommended items.

only to access the predicted ratings output by the GNN-based recommender system. Using a common framework, it can generate factual and counterfactual explanations to capture the *sufficient* and *necessary* conditions for an item to be recommended to a user, respectively. Experiments conducted on two real-world datasets combined with two well-known GNN-based recommender systems showed that GREASE outperforms baseline approaches.

Despite a lot of recent effort into explainable AI, we argue that explaining recommendations provided by AI/ML models (and, in particular, GNNs) is still underexplored. In future work, we plan to: *(i)* unify the framework introduced by GREASE to combine simultaneously factual and counterfactual explanations; *(ii)* extend our experiments to more datasets; and, possibly, *(iii)* adapt existing GNN explanation methods to the recommendation setting.

References

1. LastFM Dataset. [Online]. Available from: https://github.com/gusye1234/LightGCN-PyTorch/blob/master/data/lastfm/
2. Yelp Dataset. [Online]. Available from: https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018/
3. arre, F., Azizpour, H.: Explainability techniques for graph convolutional networks (2019), https://arxiv.org/abs/1905.13686
4. Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., Sima'an, K.: Graph convolutional encoders for syntax-aware neural machine translation. In: Proc. of EMNLP. pp. 1957–1967. ACL (2017)
5. Burke, R.: Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction **12**(4), 331–370 (2002)
6. Chen, H., Li, Y., Shi, S., Liu, S., Zhu, H., Zhang, Y.: Graph collaborative reasoning. In: Proc. of WSDM. pp. 75–84. ACM (2022)
7. Chen, H., Shi, S., Li, Y., Zhang, Y.: Neural collaborative reasoning. In: Proc. of WWW. pp. 1516–1527. ACM (2021)

Dataset	Recommendation Model	Explanation Method	Top-15 recommendations	
			PN($\uparrow$)/EC($\downarrow$)	PS($\uparrow$)/EC($\downarrow$)
LastFM	LightGCN	GREASE	**0.87**($\pm$6.00)/**3.92**($\pm$0.35)	**0.66**($\pm$5.00)/**6.33**($\pm$0.85)
		PERSONALRANK	0.76($\pm$4.00)/5.22($\pm$0.29)	0.56($\pm$2.00)/7.06($\pm$0.43)
		RANDOM	0.13($\pm$3.00)/5.23($\pm$1.60)	0.09($\pm$5.00)/8.70($\pm$0.43)
	NGCF	GREASE	**0.86**($\pm$7.00)/**3.76**($\pm$0.51)	**0.68**($\pm$9.00)/**6.30**($\pm$0.87)
		PERSONALRANK	0.80($\pm$5.00)/5.10($\pm$0.42)	0.58($\pm$3.00)/7.53($\pm$0.42)
		RANDOM	0.19($\pm$5.00)/5.77($\pm$1.82)	0.15($\pm$4.00)/8.93($\pm$0.47)
Yelp	LightGCN	GREASE	**0.86**($\pm$5.00)/**3.76**($\pm$0.51)	**0.63**($\pm$3.65)/**7.25**($\pm$0.62)
		PERSONALRANK	0.80($\pm$7.00)/5.10($\pm$0.42)	0.56($\pm$3.38)/7.36($\pm$0.63)
		RANDOM	0.18($\pm$6.00)/5.77($\pm$1.82)	0.09($\pm$2.60)/7.67($\pm$0.17)
	NGCF	GREASE	**0.85**($\pm$7.00)/**6.30**($\pm$0.53)	**0.68**($\pm$9.00)/**6.30**($\pm$0.87)
		PERSONALRANK	0.79($\pm$5.00)/6.79($\pm$0.47)	0.58($\pm$3.00)/7.52($\pm$0.42)
		RANDOM	0.18($\pm$2.00)/8.10($\pm$0.52)	0.15($\pm$5.00)/8.93($\pm$0.47)

Table 3: PN, PS, and EC of factual and counterfactual explanations generated by GREASE, PERSONALRANK, and RANDOM for top-15 recommended items.

8. Chen, L., Wu, L., Hong, R., Zhang, K., Wang, M.: Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In: Proc. of AAAI. pp. 27–34. AAAI Press (2020)
9. Duval, A., Malliaros, F.D.: Graphsvx: Shapley value explanations for graph neural networks (2021)
10. Gao, C., Wang, X., He, X., Li, Y.: Graph neural networks for recommender system. In: Proc. of WSDM. pp. 1623–1625. ACM (2022)
11. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: A factorization-machine based neural network for ctr prediction. In: Proc. of IJCAI. pp. 1725–1731. AAAI Press (2017)
12. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proc. of NeurIPS. pp. 1025–1035. Curran Associates Inc. (2017)
13. Haveliwala, T.: Topic-sensitive PageRank: A Context-sensitive Ranking Algorithm for Web Search. IEEE TKDE **15**(4), 784–796 (2003)
14. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proc. of SIGIR. pp. 355–364. ACM (2017)
15. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, pp. 639–648. ACM (2020)
16. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proc. of WWW. pp. 173–182. IW3C2 (2017)
17. Huang, C., Xu, H., Xu, Y., Dai, P., Xia, L., Lu, M., Bo, L., Xing, H., Lai, X., Ye, Y.: Knowledge-aware coupled graph neural network for social recommendation. Proc. of AAAI **35**(5), 4115–4122 (May 2021)
18. Karimi, A., Barthe, G., Balle, B., Valera, I.: Model-agnostic counterfactual explanations for consequential decisions. In: Proc. of AISTATS. vol. 108, pp. 895–905. PMLR (2020)
19. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8), 30–37 (2009)
20. Le, T., Wang, S., Lee, D.: GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction. In: Proc. of KDD. pp. 238–248. ACM (2020)

Dataset	Recommendation Model	Explanation Method	Top-20 recommendations	
			PN(↑)/EC(↓)	**PS(↑)/EC(↓)**
LastFM	LightGCN	GREASE	**0.87**(±7.00)/**4.36**(±0.46)	**0.65**(±9.00)/**6.35**(±0.87)
		PersonalRank	0.76(±5.00)/5.79(±0.32)	0.54(±5.00)/7.31(±0.52)
		Random	0.09(±4.20)/5.82(±1.90)	0.06(±7.00)/9.10(±0.49)
	NGCF	GREASE	**0.84**(±11.00)/**3.98**(±0.67)	**0.64**(±10.00)/**6.35**(±0.89)
		PersonalRank	0.77(±11.00)/5.32(±0.59)	0.53(±5.00)/7.93(±0.58)
		Random	0.17(±6.00)/5.92(±1.98)	0.14(±5.00)/9.60(±0.52)
Yelp	LightGCN	GREASE	**0.83**(±5.11)/**4.93**(±0.37)	**0.61**(±3.92)/**7.38**(±0.75)
		PersonalRank	0.77(±4.20)/5.38(±0.42)	0.53(±3.42)/7.52(±0.73)
		Random	0.16(±5.60)/5.42(±1.65)	0.07(±1.90)/8.02(±0.18)
	NGCF	GREASE	**0.75**(±8.00)/**6.70**(±0.56)	**0.63**(±10.00)/**6.37**(±0.86)
		PersonalRank	0.71(±6.12)/7.12(±0.52)	0.53(±5.00)/7.93(±0.58)
		Random	0.12(±2.00)/8.60(±0.55)	0.14(±6.00)/9.50(±0.52)

Table 4: PN, PS, and EC of factual and counterfactual explanations generated by GREASE, PersonalRank, and Random for top-20 recommended items.

21. Li, C., Jia, K., Shen, D., Shi, C.J.R., Yang, H.: Hierarchical representation learning for bipartite graphs. In: Proc. of IJCAI. pp. 2873–2879. AAAI Press (2019)
22. Lin, W., Lan, H., Li, B.: Generative causal explanations for graph neural networks (2021), https://arxiv.org/abs/2104.06643
23. Liu, Z., Meng, L., Jiang, F., Zhang, J., Yu, P.S.: Deoscillated graph collaborative filtering (2020), https://arxiv.org/abs/2011.02100
24. Lucic, A., ter Hoeve, M.A., Tolomei, G., de Rijke, M., Silvestri, F.: CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks. In: Proc. of AISTATS. vol. 151, pp. 4499–4511. PMLR (2022)
25. Lucic, A., Oosterhuis, H., Haned, H., de Rijke, M.: Focus: Flexible optimizable counterfactual explanations for tree ensembles. In: Proc. of AAAI. vol. 36, pp. 5313–5322. AAAI Press (2022)
26. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proc. of NeurIPS. pp. 4765–4774. Curran Associates, Inc. (2017)
27. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network (2020), https://arxiv.org/abs/2011.04573
28. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proc. of FAT*. pp. 607–617. ACM (2020)
29. Pazzani, M.J., Billsus, D.: Content-Based Recommendation Systems, pp. 325–341. Springer (2007)
30. Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H.: Explainability methods for graph convolutional neural networks. In: Proc. of CVPR. pp. 10764–10773. IEEE (2019)
31. Rendle, S.: Factorization machines. In: Proc. of ICDM. pp. 995–1000. IEEE (2010)
32. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proc. of KDD. pp. 1135–1144. ACM (2016)
33. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of WWW. pp. 285–295. IW3C2 (2001)
34. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems, pp. 291–324. Springer (2007)

35. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Proc. of ESWC. pp. 593–607. Springer (2018)
36. Schlichtkrull, M.S., De Cao, N., Titov, I.: Interpreting graph neural networks for NLP with differentiable edge masking (2020), https://arxiv.org/abs/2010.00577
37. Shi, W., Rajkumar, R.R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proc. of CVPR. pp. 1711–1719 (2020)
38. Siciliano, F., Bucarelli, M.S., Tolomei, G., Silvestri, F.: Newron: A new generalization of the artificial neuron to enhance the interpretability of neural networks. Proc. of IJCNN (2022), https://arxiv.org/abs/2110.02775
39. Sun, J., Zhang, Y., Guo, W., Guo, H., Tang, R., He, X., Ma, C., Coates, M.: Neighbor interaction aware graph convolution networks for recommendation. In: Proc. of SIGIR. pp. 1289–1298. ACM (2020)
40. Sun, J., Zhang, Y., Ma, C., Coates, M., Guo, H., Tang, R., He, X.: Multi-graph convolution collaborative filtering. In: Proc. of ICDM. pp. 1306–1311 (2019)
41. Tan, J., Geng, S., Fu, Z., Ge, Y., Xu, S., Li, Y., Zhang, Y.: Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In: Proc. of WWW. pp. 1018–1027. ACM (2022)
42. Tan, J., Xu, S., Ge, Y., Li, Y., Chen, X., Zhang, Y.: Counterfactual explainable recommendation. In: Proc. of CIKM. pp. 1784–1793. ACM (2021)
43. Thanapalasingam, T., van Berkel, L., Bloem, P., Groth, P.: Relational graph convolutional networks: A closer look (2021), https://arxiv.org/abs/2107.10015
44. Tolomei, G., Silvestri, F.: Generating actionable interpretations from ensembles of decision trees. IEEE TKDE **33**(4), 1540–1553 (2021)
45. Tolomei, G., Silvestri, F., Haines, A., Lalmas, M.: Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: Proc. of KDD. pp. 465–474. ACM (2017)
46. Vu, M.N., Thai, M.T.: PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. In: Proc. of the NeurIPS. Curran Associates Inc. (2020)
47. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proc. of SIGIR. pp. 165–174. ACM (2019)
48. Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., Wang, M.: Diffnet++: A neural influence and interest diffusion network for social recommendation. IEEE TKDE (2020)
49. Wu, L., Yang, Y., Zhang, K., Hong, R., Fu, Y., Wang, M.: Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In: Proc. of SIGIR. pp. 679–688. ACM (2020)
50. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: A survey. ACM Comput. Surv. (Mar 2022)
51. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: Generating explanations for graph neural networks (2019), https://arxiv.org/abs/1903.03894
52. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proc. of KDD. pp. 974–983. ACM (2018)
53. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey (2020), https://arxiv.org/abs/2012.15445
54. Yuan, H., Yu, H., Wang, J., Li, K., Ji, S.: On explainability of graph neural networks via subgraph explorations. In: Proc. of ICML. pp. 12241–12252. PMLR (2021)
55. Zhang, W., Yan, J., Wang, Z., Wang, J.: Neuro-symbolic interpretable collaborative filtering for attribute-based recommendation. In: Proc. of WWW. pp. 3229–3238. ACM (2022)