Why This, Not That? Mining User Profiles for Pair-wise Counterfactuals

Meysam Varasteh[1,*], Veronika Bogina[2], Noam Koenigstein[2] and Robin Burke[3]

[1]Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA
[2]Tel Aviv University, Tel Aviv, Israel
[3]Department of Information Science, University of Colorado Boulder, Boulder, CO 80309, USA

Abstract

The topic of explanation in recommender systems has seen steady research attention since the earliest days of the field. With some exceptions, this work has focused on the explanation of single items in a recommendation list and, especially recently, has emphasized approaches that are decoupled from the logic of the recommendation algorithm itself. Based on findings in the psychology of interpersonal communication, we propose a new task, pairwise interpretation of item rankings, asking the comparative question "Why is item A ranked higher than item B?". An effective solution to this task, we argue, is inherently grounded in the operation of the recommendation algorithm. We propose a class of techniques based on counterfactual learning to uncover the items in a user's profile that have contributed to the relative ranking of items. Using multiple datasets, we show that it is possible to identify such items as potential basis for comparative explanation.

Keywords

recommender systems, explanation, comparative explanation, counterfactual explanation

1. Introduction

Why was this item recommended? This question has been central to recommender systems research for decades, with well-established methods for explaining the presence of a single item in a ranked list [1, 2]. In this work, we focus on a different and equally natural question: "Why is item T ranked higher than item C?". This problem is referred to as "Comparing Item Rankings" in [3].

Psychological studies show that people often seek contrastive explanations when reasoning about decisions [4]. Comparison is a fundamental component of human judgment [5], and preferences can be influenced simply by altering the set of alternatives [6]. In the context of recommender systems, comparative explanations have the potential to enhance clarity and relatability by focusing on the specific factors that distinguish two competing items, rather than enumerating all reasons for recommending T in isolation. This narrower focus can make the explanation both more intuitive for the user and potentially easier to generate algorithmically.

From an algorithmic perspective, the question "Why is T ranked above C?" is inherently counterfactual: the user implicitly asks "and not the other way around?". Addressing this requires identifying the minimal changes to the underlying data that would reverse the relative positions of the two items. Such counterfactual comparative explanations directly expose causal relationships between user data and ranking outcomes, aligning naturally with transparency goals in recommender systems [2]. They are valuable not only for end users but also for system designers and auditors, and they support regulatory demands for explanations of how personal data influences automated decisions, such as those in the GDPR[1].

IntRS'26: Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, September 28, 2026, Minneapolis.
*Corresponding author.
meysam.varasteh@colorado.edu (M. Varasteh); sveron@gmail.com (V. Bogina); noamk@tauex.tau.ac.il (N. Koenigstein); robin.burke@colorado.edu (R. Burke)
0009-0003-0346-4951 (M. Varasteh); 0000-0002-8005-7618 (V. Bogina); 0000-0001-8219-4512 (N. Koenigstein); 0000-0001-5766-6434 (R. Burke)

[1]https://www.consilium.europa.eu/en/policies/data-protection/data-protectionregulation/

As a terminological matter, we note that the terms "comparative" and "contrastive" are both used somewhat interchangeably in different publications in this area [7, 8, 9, 10]. The term "contrastive explanation" has a well-established meaning in the explainable AI literature meaning a counterfactual explanation (CE) for a classification: "Why was this instance classified in category A instead of category B?" Rather than trying to expand this established definition to cover our case, we use the term "comparative" instead [11, 12, 13]. We will use the terms *target* item to refer to the higher ranked item T and *comparative* item to refer to the lower ranked item C.

Following the work of Barkan et al. [14], we adopt a learning-oriented approach to this CE problem. Similar to their method, we develop an *explainer model* that learns how modifications in a user's profile influence the recommendation outcomes. Our findings indicate that this method allows us to identify items in a user's profile that, when removed, would reverse the ranking positions of the target item and the comparative item. Such a set of items we term a *perturbation*. Thus, our intended explanation is of the following form: "The reason item T is ranked higher than item C is because the user profile included items x_1, x_2, and x_3. If these items were not present, C would have been recommended higher than T."

We highlight two aspects relevant to the quality of our explanation output, drawing from Grice's maxims of communication [15]. One is *veracity*, the "maxim of quality". It should be the case that C will be ranked above T if the user's profile is edited by removing the identified items and the recommendations are re-generated. The second quality is that the explanation should be succinct, the "maxim of clarity". Ideally, the suggested perturbation should be the smallest set of items for which the reversal happens.

This study represents an initial foray into the development of comparative ranking explanations using this approach. We focus here on the feasibility of generating accurate and succinct counterfactual perturbations. We address the following research questions:

- **RQ1**: Can counterfactual learning be used to efficiently, in terms of perturbation size, approximate profile perturbations for generating contrastive ranking explanations?
- **RQ2**: How does a contrastive objective perform compared to one that decouples target and comparative items?
- **RQ3**: How do the results vary across different datasets and recommendation algorithms?

2. Related Work

As noted above, explanation is a core topic in recommender systems research, and there are a variety of well-known techniques, especially for explaining the presence of a single item in a recommendation list. Readers are referred to [1, 2] for details of the state of the art in these areas. Our research directly addresses a need identified in [3], which surveys the comparative explanation task and proposes several variations. However, this position paper focuses solely on defining the comparative explanation problem and does not present any implementations.

In the realm of XAI, model-agnostic methods like SHAP [16] and LIME [17] stand out for their versatility and flexibility. These techniques are not tied to any specific model architecture, allowing them to be applied across various ML models, including those used in recommendation systems. SHAP and LIME generate explanations by perturbing input vectors to assess the contribution of each feature to the model's predictions. However, when applied to recommender systems, this approach often requires perturbing the entire user dataset, which can be computationally intensive and impractical for real-time applications with many items.

Counterfactual explanations for recommendations are a more recent innovation, drawing on research from XAI. CEs generally seek to demonstrate how altering specific inputs would affect a model's prediction [18]. This approach is useful for revealing the causal links between user data and the recommendations provided. By pinpointing modifications in user data that would lead to different

recommendations. Recent examples of these methods include [19, 20, 21]. These techniques are challenging to employ in practical contexts because they require search over a large perturbation space for each user, a process that would be prohibitive in a real-world setting.

A complementary line of work moves beyond single-instance perturbations by learning a model to predict counterfactual explanations at scale. Rather than searching for perturbations individually, these approaches train a mask over user profiles or item features to generate explanations efficiently across many instances [14, 22, 23, 18, 24]. In this work, we adapt the LXR framework proposed by Barkan et al. [14], which learns to predict profile masks, for the comparative explanation task.

Several recent papers have addressed the comparative explanation task for ranking [10, 25, 26] and NLP task [8, 12]. In this line of research, the items are compared based on their attributes. For example, in [10], the authors provide an example explanation: *Characteristics in favor of Candidate 00079 include a higher score in HSC_P and a higher score in SSC_P. Characteristics in favor of Candidate 00188 include a higher score in DEGREE_P and having previous work experience.* However, this approach is not easily transferable to recommendation tasks, where ranking criteria are personalized. In contrast, our counterfactual explanation is more closely tied to the user's profile, focusing on what would need to change in the user's preferences for the recommendation to differ. Also, [8] proposed a method that produces contrastive explanations by projecting the latent space of inputs. However, since one of the main characteristics of our model is being model-agnostic, this approach is not used.

Contrastive explanation is also applied in Information Retrieval (IR) due to the ranking nature of the models, which are designed to order documents for each query. Several studies have investigated contrastive explanations with the aim of addressing ranking related questions, such as: "Why is this document ranked higher than the other?" For example, [27] proposed a variant of the LIME [17]method adapted specifically to the contrastive setting, while [28] approximates the original model with a simple ranker to explain "Why document d_i is more relevant than d_j. Similarly, [29] incorporated multiple simple rankers to provide listwise explanations for ranking models. However, these IR methods are not applied to cases where ranking is personalized as in recommendation.

There is also research that uses the term "comparative explanation" (for example, [7]) to describe explanations that compare a single recommended item against an item or items from the user's profile. This is also a different explanation task than ours, which poses a counterfactual question about two recommended items and requires very different explanation methods.

3. Explanation Models

Our approach to comparative explanation builds directly on the LXR framework [14, 30], a state-of-the-art method for generating counterfactual explanations of individual recommended items. We first outline the LXR method to establish the foundation of our work, then describe how we extend it to address the comparative setting. All notation used throughout the paper is summarized in Table 1.

3.1. LXR

LXR [14] formulates the task of explaining a single recommended item as identifying the specific items in a user's profile $\mathbf{x}$ whose removal would cause that item to disappear from the top slate of recommendations. The approach is learning-based: the explainer is a trainable function optimized via a counterfactual loss over the recommender's outputs.

Formally, the learnable explainer $\mathbf{e}^l : \{0,1\}^{|\mathcal{V}|} \times \{0,1\}^{|\mathcal{V}|} \to [0,1]^{|\mathcal{V}|}$ takes as input the user's interaction history $\mathbf{x} \in \{0,1\}^{|\mathcal{V}|}$ and a target item $\mathbf{y} \in \{0,1\}^{|\mathcal{V}|}$ represented as a one-hot vector, and produces an *explanation mask* $\mathbf{m}^l \in [0,1]^{|\mathcal{V}|}$, where each entry represents the estimated contribution of the corresponding profile item to the recommendation of $\mathbf{y}$.

We can order the profile items by their score in the mask and establish some threshold for removal. The set of items removed we call the profile *perturbation*, denoted by Π. The quality of Π is determined by its *veracity*: if the items in Π are removed from $\mathbf{x}$, the recommender should cease to recommend $\mathbf{y}$.

Symbol	Description
$\mathcal{U}$	The set of users.
$\mathcal{V}$	The set of items.
$\mathbf{x}$	The user profile, represented as a binary vector over items.
$\mathbf{y}_t, \mathbf{y}_c$	The target and comparative items, represented as a one-hot vectors.
$\mathbf{m}$	An explanation map / importance scores (superscript denotes type).
$\mathbf{e}$	An explainer (superscript denotes type).
$\mathbf{f}$	A recommender system.
$\mathbf{r_f}(\mathbf{x}, \mathbf{y})$	The rank of item $\mathbf{y}$ in the recommendations produced by $\mathbf{f}$ for user $\mathbf{x}$.
Π	A perturbation of a user profile.

Table 1
Table of notations

The conciseness of the explanation is measured by the size of Π: ideally it includes only a small subset of items whose removal leads to a sharp drop in $\mathbf{y}$'s rank.

3.2. Comparative Explanation

Next, we formulate a *comparative explainer* that tries to explain the recommender's preference for a target item over a comparative item. Let $\mathbf{e}^c : \{0,1\}^{|\mathcal{V}|} \times \{0,1\}^{|\mathcal{V}|} \times \{0,1\}^{|\mathcal{V}|} \to [0,1]^{|\mathcal{V}|}$ be a comparative explainer that receives a user data vector $\mathbf{x} \in \{0,1\}^{|\mathcal{V}|}$ and two one-hot vectors: one representing the target item $\mathbf{y}_t \in \{0,1\}^{\mathcal{V}}$, and another, the comparative item $\mathbf{y}_c \in \{0,1\}^{\mathcal{V}}$. Similar to LXR, the output of the explainer is an *explanation mask* $\mathbf{m}^c = \mathbf{e}(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c)$, where $\mathbf{m}^c[i]$ indicates the contribution of item i from the user's history to the recommender's preference of the target item $\mathbf{y}_t$ over the comparative item $\mathbf{y}_c$. In what follows, we present several approaches to implement such a comparative explainer.

3.2.1. CLXR-score:

A naive implementation of a comparative explainer can be based on a simple adaptation of a pre-trained LXR explainer $\mathbf{e}^l$ to the aforementioned comparative task. This can be achieved by considering the cases of the target item and the comparative item independently, using the explanation scores from the LXR explainer. We refer to this variant as *CLXR-score*.

Given a pre-trained LXR explainer $\mathbf{e}^l$, we employ it to produce two explanation maps $\mathbf{m}^l_t$ and $\mathbf{m}^l_c$ and combine them as follows:

$$\begin{aligned} \mathbf{m}^l_t &= \mathbf{e}^l(\mathbf{x}, \mathbf{y}_t) \\ \mathbf{m}^l_c &= \mathbf{e}^l(\mathbf{x}, \mathbf{y}_c) \\ \mathbf{m}^c &= \mathbf{m}^l_t \odot (\mathbf{1} - \mathbf{m}^l_c) \end{aligned} \tag{1}$$

where $\mathbf{m}^c$ is the comparative explanation mask, $\mathbf{1}$ is a vector of ones, and $\odot$ is the element-wise product.

The $\mathbf{m}^l_t$ mask identifies the items in the profile that are likely to reduce the rank of $\mathbf{y}_t$ if removed from the user profile. The inverse of the $\mathbf{m}^l_c$ mask identifies items that do not contribute to the $\mathbf{y}_c$'s rank. Therefore, the multiplicative combination of these values focuses $\mathbf{m}^c$ on items that will reduce the rank of $\mathbf{y}_t$ but not reduce the rank of $\mathbf{y}_c$ if removed from the profile.

3.3. Learning Comparative Explanations

The primary advantage of the CLXR-score implementation is its reliance on the pre-trained LXR model from Barkan et al. [14]. This allows a single LXR explainer to be used for both single-item explanations

and comparative explanations without additional optimization. Next, we explore solutions that directly optimize for comparative explanations.

Given the user's history vector $\mathbf{x}$, target item $\mathbf{y}_t$, and comparative item $\mathbf{y}_c$, our objective consists of four loss terms:

- $\mathcal{L}^{\text{cmp}}\big(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c\big)$: Encourages finding explanations that emphasize the target item while suppressing the comparative item.
- $\mathcal{L}^{\text{pred}}\big(\mathbf{x}, \mathbf{y}_t\big)$: Inspired by the LXR objective, this term prioritizes supporting the target item.
- $\mathcal{L}^{\text{inv}}\big(\mathbf{x}, \mathbf{y}_t\big)$: This term is designed to suppress the recommendation of the target item when the most important elements related to the target item are removed.
- $\mathcal{L}^{\text{reg}}\big(\mathbf{m}^c\big)$: Regularization term promoting mask sparsity.

The complete CLXR objective is:

$$\begin{aligned}\mathcal{L}^{\text{CLXR}}\big(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c\big) =& \lambda_{\text{cmp}}\mathcal{L}^{\text{cmp}}\big(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c\big) + \lambda_{\text{pred}}\mathcal{L}^{\text{pred}}\big(\mathbf{x}, \mathbf{y}_t\big) + \\ & \lambda_{\text{inv}}\mathcal{L}^{\text{inv}}\big(\mathbf{x}, \mathbf{y}_t\big) + \lambda_{\text{reg}}\mathcal{L}^{\text{reg}}(\mathbf{m}^c)\end{aligned} \tag{2}$$

This framework allows flexibility in defining the loss functions based on the explainer's architecture. Below, we present two implementation approaches.

3.3.1. CLXR-joint Implementation

Our first implementation, *CLXR-joint*, employs a Siamese network to jointly optimize all loss terms in Eq. 2. This approach builds on the CLXR-score concept but replaces the LXR explainer $\mathbf{e}^l$ in Eq. 1 with an intermediate Siamese network $\mathbf{e}^s$, defined as $\mathbf{e}^s : \{0,1\}^{|\mathcal{V}|} \times \{0,1\}^{|\mathcal{V}|} \to [0,1]^{|\mathcal{V}|}$. The comparative explanation mask $\mathbf{m}^c$ is then computed as:

$$\begin{aligned}\mathbf{m}_t^s &= \mathbf{e}^s(x, \mathbf{y}_t), \\ \mathbf{m}_c^s &= \mathbf{e}^s(x, \mathbf{y}_c), \\ \mathbf{m}^c &= \mathbf{m}_t^s \odot (\mathbf{1} - \mathbf{m}_c^s)\end{aligned} \tag{3}$$

Here, the Siamese network $\mathbf{e}^s$ generates intermediate masks $\mathbf{m}_t^s$ and $\mathbf{m}_c^s$, akin to $\mathbf{m}_t^l$ and $\mathbf{m}_c^l$ from Eq. 1. Thus, *CLXR-joint* learns a comparative explanation mask $\mathbf{m}^c$ using:

$$\mathbf{m}^c = \mathbf{e}^c(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c) = \mathbf{e}^s(\mathbf{x}, \mathbf{y}_t) \odot \big(\mathbf{1} - \mathbf{e}^s(\mathbf{x}, \mathbf{y}_c)\big), \tag{4}$$

where $\mathbf{e}^s$ is optimized using Eq. 2. In essence, this architecture is similar to that of CLXR-score, except that in CLXR-joint, $\mathbf{e}^s$ is optimized directly for a comparison explanation task, whereas in CLXR-score, $\mathbf{e}^c$ is the original LXR explainer, trained on explaining single items.

Specifically, training $\mathbf{e}^c$ is performed via optimizing the following loss terms:

$$\begin{aligned}\mathcal{L}^{\text{cmp}} &= -\log\big(\mathbf{y}_t\mathbf{f}(\mathbf{x} \odot \mathbf{m}^c) - \mathbf{y}_c\mathbf{f}(\mathbf{x} \odot \mathbf{m}^c)\big), \\ \mathcal{L}^{\text{pred}} &= -\log(\mathbf{y}_t\mathbf{f}(\mathbf{x} \odot \mathbf{m}_t^s)), \\ \mathcal{L}^{\text{inv}} &= \log(\mathbf{y}_t\mathbf{f}(\mathbf{x} \odot (1 - \mathbf{m}_t^s))) + \log(\mathbf{y}_t\mathbf{f}(\mathbf{x} \odot (1 - \mathbf{m}^c))), \\ \mathcal{L}^{\text{reg}} &= |\mathbf{m}^c|.\end{aligned} \tag{5}$$

The comparative loss $\mathcal{L}^{\text{cmp}}$ maximizes the score gap between $\mathbf{y}_t$ and $\mathbf{y}_c$ after applying the explanation mask $\mathbf{m}^c$ to $\mathbf{x}$. The prediction loss $\mathcal{L}^{\text{pred}}$ ensures $\mathbf{y}_t$ remains the top-ranked item by maximizing its score when masked by $\mathbf{m}_t^s$. The inverse loss $\mathcal{L}^{\text{inv}}$ is designed to suppress the recommendation of the target item when the most important items related to it are removed from the user's profile by inverting

$\mathbf{m}_t^s$ and $\mathbf{m}^c$. Finally, the regularization loss $\mathcal{L}^{\text{reg}}$ enforces sparsity in $\mathbf{m}^c$. This formulation enables a more targeted optimization for comparative explanations, improving fidelity and interpretability.

We note that employing $\mathcal{L}^{\text{cmp}}$, $\mathcal{L}^{\text{pred}}$ and $\mathcal{L}^{\text{inv}}$ requires an "untraditional" setup where the recommender model $\mathbf{f}$ is embedded within the learning objective and plays an active role in the optimization process. However, the recommender's weights remain frozen to preserve the integrity of the explanation target.

3.3.2. TDLR Implementation

Our second implementation adopts the Targeted Deviation - Logit Ranking (TDLR) loss, commonly used in adversarial attacks [31]. Hence we dub this implementation *CLXR-tdlr*. Different from the *CLXR-joint* implementation, CLXR-tdlr does not rely on an intermediate network $\mathbf{e}^s$. Instead, given both items ($\mathbf{y}_t$ and $\mathbf{y}_c$), the comparative explainer $\mathbf{e}^c$ directly learns the comparative mask $\mathbf{m}^c$. Additionally, we set $\lambda_{\text{pred}} = 0$ and $\lambda_{\text{inv}} = 0$, effectively removing $\mathcal{L}^{\text{pred}}$ and $\mathcal{L}^{\text{inv}}$ from the objective in Eq. 2, relying solely on the comparative loss $\mathcal{L}^{\text{cmp}}$ and regularization term $\mathcal{L}^{\text{reg}}$ defined as :

$$\begin{aligned} \mathcal{L}^{\text{cmp}} &= -\frac{\mathbf{y}_t\mathbf{f}(\mathbf{x} \odot \mathbf{m}^c) - \mathbf{y}_c\mathbf{f}(\mathbf{x} \odot \mathbf{m}^c)}{\mathbf{f}(\mathbf{x})[1] - \mathbf{f}(\mathbf{x})[K] + \epsilon}, \\ \mathcal{L}^{\text{reg}} &= |\mathbf{m}^c|, \end{aligned} \tag{6}$$

where $\mathbf{f}(\mathbf{x})[1] - \mathbf{f}(\mathbf{x})[K]$ represents the score difference between the top-ranked item and the item at position K. We set $K = 10$ and $\epsilon = 1 \times 10^{-8}$ as hyperparameters.

Similar to the *CLXR-joint* implementation, the comparative loss $\mathcal{L}^{\text{cmp}}$ in the *CLXR-tdlr* implementation utilizes the recommender model $\mathbf{f}$ as part of the learning objective (without changing $\mathbf{f}$). Minimizing $\mathcal{L}^{\text{cmp}}$ guides the explainer $\mathbf{e}^c$ to find a mask $\mathbf{m}^c$ that increases the score gap between $\mathbf{y}_t$ and $\mathbf{y}_c$. The denominator normalizes this effect, ensuring loss stability across varying score distributions. The regularization term $\mathcal{L}^{\text{reg}}$ remains an ℓ_1 sparsity constraint to encourage minimal yet effective explanations.

3.4. Optimization

To train the CLXR explainers, we generate recommendation slates for each training set user using the recommendation algorithm. We assign each user's target item $\mathbf{y}_t$ as the top-ranked recommendation from this slate. The comparative item is randomly sampled from the remaining items ranked 2 through 10.

In both implementations, the explainer networks are modeled as simple feed-forward multi-layer perceptrons. The explainer's parameters, denoted by θ, are optimized using stochastic gradient descent to minimize the following objective:

$$\theta^* = \arg\min_{\theta} \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} \mathcal{L}^{\text{CLXR}}(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c), \tag{7}$$

where $\mathcal{U}$ is the set of users, and $\mathcal{L}^{\text{CLXR}}$ as in Eq. 2.

4. Evaluation Metrics

Comparative explanation of recommendation outputs is a novel problem, and no established metrics exist for evaluating the quality of potential solutions. We therefore define two metrics to assess the effectiveness of our algorithms. Since evaluating all possible item pairs in a user's recommendation list is impractical, we focus on the top-ranked item for each user, denoted as $\mathbf{y}_t$ where $\mathbf{r}_{\mathbf{f}}(\mathbf{x}, \mathbf{y}_t) = 1$. The comparative item $\mathbf{y}_c$ is sampled from the other top-K recommendations, with $K = 10$.

The user profile $\mathbf{x}$ is a set of items x_i previously rated by the user. Our algorithms produce explanations in the form of an attribution mask $\mathbf{m}^c$ over these items, indicating their estimated influence in ranking

$\mathbf{y}_t$ above $\mathbf{y}_c$. As discussed earlier, our intended counterfactual explanation takes the form: "Item T is ranked higher than item C because you liked x_1, x_2, and x_3. If you had not liked these items, C would have been ranked higher than T."

A perturbation Π_σ is defined as the subset of items removed from the profile to induce rank reversal, where σ is the fraction of profile items with the highest attribution scores. For example, $\Pi_{0.2}$ denotes the top 20% of items in $\mathbf{x}$ according to $\mathbf{m}^c$.

We measure the size of the smallest perturbation that reverses the ranks of $\mathbf{y}_t$ and $\mathbf{y}_c$, with smaller values indicating a better explainer. This is the *Minimal Perturbation for Ranking Reversal* (MPRR):

$$\text{MPRR}_{\%}(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c) = \min_{\sigma \in [0,1]} \{\sigma : \mathbf{f}(\mathbf{x} \setminus \Pi_\sigma, \mathbf{y}_t) < \mathbf{f}(\mathbf{x} \setminus \Pi_\sigma, \mathbf{y}_c)\}. \tag{8}$$

Note that our search over σ considers progressively larger subsets of top-ranked items according to $\mathbf{m}^c$, rather than all possible item subsets. As a result, it is possible for a valid perturbation to exist but remain undiscovered if its items are not ranked highly by the explainer. Hence, if no perturbation in this sequence produces a rank reversal, we set $\text{MPRR}_{\%} = \infty$ and exclude that user from the MPRR calculation.

Differences in profile length make percentages more comparable across datasets, but we also report $\text{MPRR}_{\#}$, the corresponding perturbation size in absolute item counts.

Some rank reversals may be fundamentally impossible, for example due to strong popularity effects in the data. For example, consider two diverse movies: Abbas Kiarostami's *A Taste of Cherry* and Greta Gerwig's *Barbie*. While *A Taste of Cherry* is an acclaimed film, even winning the Palme d'Or in 1997, a broad-based recommender system trained on the preferences of the general American audience is unlikely to rank it above *Barbie*, a blockbuster with extensive marketing and widespread appeal. If a user receives *Barbie* as their top recommendation, no realistic perturbation may push *A Taste of Cherry* above it, given its niche appeal. In such cases, a system might instead need to fall back to a general, non-personalized explanation.

Because not all pairs can be explained, we also report *Coverage*:

$$\text{Coverage} = \frac{1}{|\mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{U}} \mathbb{I}\big(\text{MPRR}_{\%}(\mathbf{x}, \mathbf{y}_t, \mathbf{y}_c) < \infty\big), \tag{9}$$

where $\mathbb{I}$ is the indicator function.

High Coverage means explanations can be generated for more users, while low $\text{MPRR}_{\#}$ means explanations are more succinct. We expect a trade-off: more precise perturbations may be possible for fewer users, while broader coverage may require accepting larger perturbations. The appropriate balance depends on the intended application.

5. Methodology

5.1. Data sets

Our evaluations are based on the MovieLens 1M (ML-1M), Yahoo Music (Yahoo), and Pinterest datasets. To simulate implicit ratings for ML-1M, we kept only ratings of 3.5 or higher and included only users and items with at least two ratings. The resulting dataset comprised 575,128 ratings from 6,037 users across 3,381 items. For the Yahoo dataset, we included ratings of 70 or higher and retained users and items with at least two ratings. To keep the datasets similar in size, we sampled a subset of 13,797 users, yielding a dataset with 365,750 ratings across 4,604 items. Finally, for the Pinterest dataset [32], we randomly sampled 486,744 ratings from 19,155 users covering 9,362 items. Each dataset was split into 80% for training and 20% for testing. 10% of the training data was used for hyperparameter tuning. The code for data preparation, model training, and evaluation is available via anonymous GitHub[2].

[2]https://github.com/that-recsys-lab/CLXR-an

5.2. Recommendation models

Although the explainer model is designed to be algorithm-agnostic, its performance is a function of the choice of recommendation algorithm $\mathbf{f}$, especially concerning the well-known issue of popularity bias [33]. To explore this, we present the results of the experiments using two different recommendation algorithms, following the approach in [14]. The performance of two recommendation models across the three datasets is presented in Table 2.

Matrix factorization (MF): Matrix factorization is a widely used technique in recommender systems that models the interaction between users and items to predict preferences. The goal is to decompose the user-item interaction matrix into two lower-dimensional matrices: user latent factors and item latent factors. Each user and item is represented as a vector in a shared latent space[34]. The predicted interaction is computed as the dot product of these vectors. We developed a version of matrix factorization (MF) where the model takes a binary encoding of the user's historical interactions and dynamically computes the user's latent representation using a simple projection matrix.

Variational Autoencoder (VAE): A Variational Autoencoder (VAE) is a probabilistic generative model that has been effectively applied in recommender systems for learning user and item representations. It consists of two main components: an encoder and a decoder. The encoder maps the user's interaction history into a latent space, producing a probabilistic latent representation modeled as a Gaussian distribution [35, 36]. The decoder reconstructs the original interaction data from this latent representation. Our VAE recommender model uses a similar architecture to that in [37] and [14].

	ML1M		**Yahoo**		**Pinterest**	
	HR	**CC**	**HR**	**CC**	**HR**	**CC**
MF	0.08	0.13	0.23	0.20	0.04	0.13
VAE	0.19	0.44	0.40	0.23	0.1	0.72

Table 2
Hit rate (HR) and catalog coverage (CC) at 10 for the recommendation algorithms / dataset combinations.

5.3. Explanation models

In this study, we evaluate the following models for ranking items to generate counterfactual comparative explanations.

Popularity (POP): This is a simple baseline that ranks items according to their popularity score (i.e., number of associated actions).

SHAP: SHAP (SHapley Additive exPlanations) [38] is a popular model-agnostic method for explaining predictions of machine learning models, based on the concept of Shapley values. In the context of recommender systems, the number of perturbations grows exponentially with the number of items in a user's profile, making this approach computationally expensive. In [21], SHAP was applied to recommender systems using only 12 explainable features. In our case, we grouped each user's items into $K = 10$ clusters and computed Shapley values for the aggregated item clusters.

LIME: LIME (Local Interpretable Model-Agnostic Explanations) [39] is a local, model-agnostic explanation method that approximates the original model using linear surrogate models. In LIME, the original input is perturbed, weighted based on proximity, and used to fit a surrogate model such as linear regression. In [40], LIME-RS was introduced for recommender system applications.

ACCENT: ACCENT (Action-based Counterfactual Explanations for Neural Recommenders for Tangibility) [41] is a model-agnostic counterfactual explanation framework based on influence functions for neural networks. It employs the Fast Influence Analysis (FIA) [42] mechanism to approximate the contribution of each user's action, based on its estimated impact on the prediction generated by the neural recommender. ACCENT identifies counterfactuals whose removal from the training set would lead to a different recommendation.

LXR: Although LXR is not specifically designed for comparative explanation, it identifies items that support the recommendation of a single target item. A perturbation involving these items will lower the

target's rank, and may drop the target item below the comparator item. We use the LXR implementation from [14], and as the results will show, it is competitive with our other methods under some conditions.

CLXR-score: As described above, CLXR-score extends LXR to the contrastive task by independently handling the target and comparative items and constructing a mask that integrates their scores, as shown in Eq. 1.

CLXR-joint: The CLXR-joint model is trained by minimizing the objective function presented in Eq. 2. Similar to [14], the MLP in the explainer comprises $d = 20$ hidden layers with dimension d. The output dimensions of the first and second MLP layers are set to $2d$ and $3d$, respectively. The explainer is trained using the Adam optimizer [43] with a learning rate of 0.001 and a batch size that varies depending on the dataset and recommendation model.

CLXR-tdlr: The CLXR-tdlr model is trained to minimize the objective function described in Eq. 6. It has the same structure as CLXR-joint and was trained and tuned using the same procedures.

We performed a grid search to optimize all hyperparameters as well as batch size, learning rate, and latent factor dimensionality. Optimal hyperparameters can be found in the repository.

5.4. Experimental procedures

The experiments can be divided into two stages: the training of the explainer model and its evaluation. The first task is to assemble the training data for the explainer. After training the recommender system $\mathbf{f}$, we generate recommendations for all users. For each set of recommendations, we select $\mathbf{y}_t$ (item of rank 1) and $\mathbf{y}_c$ (randomly chosen from ranks 2-10). The explainer training data therefore consists of $< \mathbf{x}, \mathbf{y}_t, \mathbf{y}_c >$ tuples across the training set. The explainer $\mathbf{e}^c$ is trained over this data by optimizing the objective described in Section 3.4.

The evaluation stage proceeds similarly, using $\mathbf{f}$ to generate a recommendation list for each test user and identifying $\mathbf{y}_t$ and $\mathbf{y}_c$ items from these recommendations. Using the explainer model $\mathbf{e}^c$, we compute the mask $\mathbf{m}^c$, scoring each item in the user profile by its predicted contribution to the relative ranking of $\mathbf{y}_t$ and $\mathbf{y}_c$.

Given the mask $\mathbf{m}^c$, we generate a series of successively larger perturbations $\Pi_{0.1..1.0}$ in 0.1 increments and use these to generate new recommendations based on perturbed profiles $\mathbf{x} \backslash \Pi_\sigma$. We examine the predicted ratings assigned to $\mathbf{y}_t$ and $\mathbf{y}_c$ in these recommendations in order to compute $\text{MPRR}_\%$ and Coverage.

6. Results

Table 3 presents the Coverage and average MPRR scores for the ML-1M, Yahoo and Pinterest datasets using two recommendation models. Higher Coverage values indicate better performance, whereas lower MPRR values are preferable. The results show that CLXR-joint consistently outperforms all baselines in both Coverage and MPRR except for the VAE model on the Pinterest dataset. CLXR-tdlr is also competitive, especially for Matrix Factorization. Additionally, ACCENT, which is based on the Fast Influence Analysis (FIA) on the gap score between the target and comparative items, proves to be the best model for the VAE/Pinterest task and also has good performance in the MF/Yahoo condition. As noted earlier, the model's inherent contrastive structure makes it a suitable baseline.

Figure 1 illustrates cumulative plots for each explanation model tracking when the target and comparative items are reversed within a perturbation of at most that size. The Y-axis represents the cumulative fraction of users, while the X-axis indicates the number of masked items or MPRR. We focus on smaller perturbation sizes (< 50 for MovieLens, < 25 for Yahoo and Pinterest datasets) as these are more likely to be practically useful as explanations. The most desirable region in these plots is the upper-left quadrant, where an explainer achieves high Coverage (affecting a large number of users) with minimal perturbation to user profiles (MPRR, measuring explanation succinctness).

These figures provide a more detailed view of algorithm performance while reaffirming CLXR-joint and CLXR-tdlr as the strongest performers except for the VAE/Pinterest. For the matrix factorization model on MovieLens (Figure 1a), the curves for all algorithms are relatively close, but CLXR-joint

Recommender	Method	ML-1M		Yahoo		Pinterest	
		Coverage ↑	MPRR$_{\#}$ ↓	Coverage ↑	MPRR$_{\#}$ ↓	Coverage ↑	MPRR$_{\#}$ ↓
MF	Popularity	56%	50	66%	15	62%	16
	LIME	59%	37	67%	11	60%	12
	ACCENT	68%	28	79%	11	58%	14
	SHAP	32%	52	63%	19	57%	16
	LXR	72%	33	78%	11	74%	11
	CLXR-score	65%	31	63%	10	64%	11
	CLXR-joint (our)	**80%**	**24**	**81%**	**7**	**77%**	**9**
	CLXR-tdlr (our)	79%	26	80%	10	71%	9
VAE	Popularity	74%	64	65%	20	72%	15
	LIME	61%	38	56%	14	68%	14
	ACCENT	77%	34	67%	12	**86%**	**9**
	SHAP	57%	68	50%	24	50%	22
	LXR	82%	30	73%	14	74%	15
	CLXR-score	82%	30	73%	12	77%	13
	CLXR-joint (our)	**92%**	**21**	**83%**	**9**	78%	12
	CLXR-tdlr (our)	85%	27	70%	14	70%	13

Table 3
Coverage and MPRR scores for the MF and VAE recommenders on the three datasets. The best result is shown in **bold**.

and CLXR-tdlr are dominant throughout. With the VAE recommender on this dataset, CLXR-joint is even more dominant with CLXR-tdlr a bit lower. Note that in these figures, the perturbation sizes are limited to < 50 and < 25, whereas in Table 3 there are no such limitations. When applied to the Yahoo dataset Figure 1b, CLXR-joint and CLXR-tdlr are more clearly dominant. In most conditions, LXR and CLXR-score have similar performance, with LXR actually improving on CLXR-score in Fig 1a.

On the Pinterest dataset with the MF model, CLXR-joint and CLXR-tdlr perform similarly and are the dominant methods. However, for the VAE model, the ACCENT method outperforms all other models, with CLXR-joint as the second best baseline.

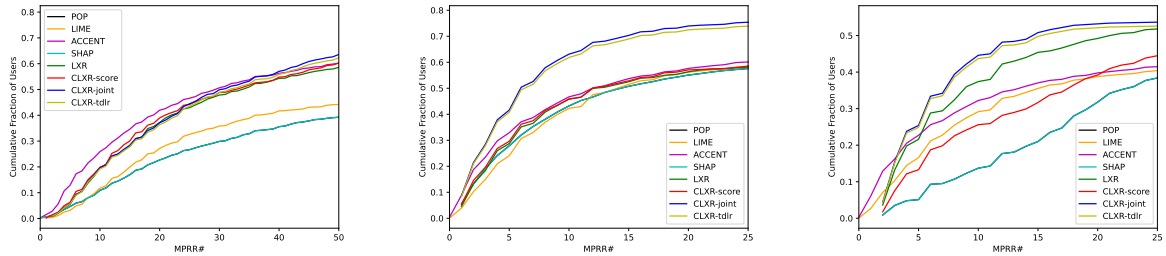
6.1. Explanation Example

An example counterfactual perturbation is shown in Figure 2. The user profile shown is synthetic, but the perturbation was generated using the CLXR-joint explainer trained on the VAE recommender and the ML1M dataset. The table in the upper left presents the user's profile, while the list on the right displays the corresponding recommendations. The task is to explain why the science fiction/horror classic *Alien* is ranked higher than Mel Gibson's historical epic *Braveheart*.

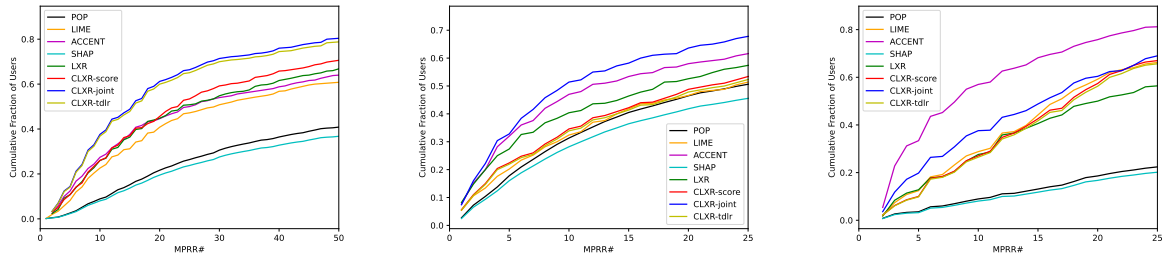
The lower figure presents the perturbed user profile, where *Halloween*, *Forbidden Planet*, and *Brazil* have been removed. The numerical values in the lower table are from the $\mathbf{m}^c$ mask, representing the importance of each item to this task as output by the $\mathbf{e}^c$ explainer. The revised recommendation list (shown in the lower right) results in a rank reversal between *Alien* and *Braveheart*. From this information, it would be possible to generate an explanation as follows: "If you had not liked *Halloween*, *Forbidden Planet*, and *Brazil*, then I would have recommended *Braveheart* higher than *Alien*." Interestingly, other science fiction films such as *Apollo 13*, *Mission to Mars*, and *War of the Worlds* have much lower $\mathbf{m}^c$ scores.

6.2. The Impact of Popularity

Recommendation and ranking systems are known to suffer from popularity bias, which arises because errors on popular items will be penalized more heavily in training as they affect more users than rare items [44, 45]. Given the performance of the POP algorithm in our experiments, we believe that popularity bias may also impact our explainer models as well and lower their effectiveness. We would



(a) Recommender: MF, Dataset: MovieLens (b) Recommender: MF, Dataset: Yahoo (c) Recommender: MF, Dataset: Pinterest



(d) Recommender: VAE, Dataset: MovieLens (e) Recommender: VAE, Dataset: Yahoo (f) Recommender: VAE, Dataset: Pinterest

Figure 1: Coverage versus minimum perturbation size for the different algorithms.

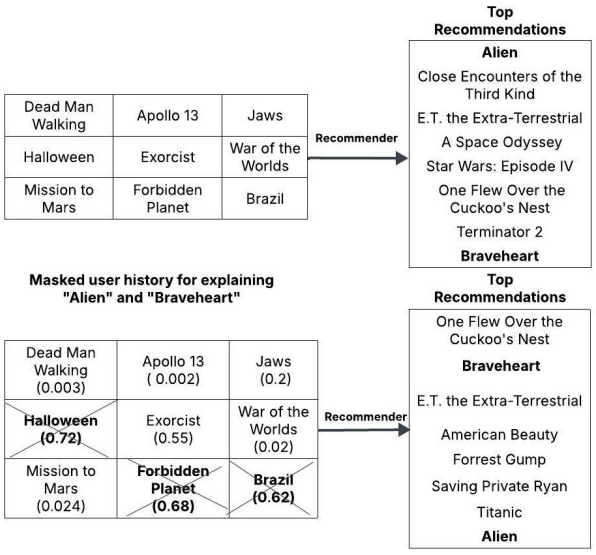


Figure 2: Example of comparative explanation generation. The original profile and recommendations appear at the top, with *Alien* as the target item and *Braveheart* as the comparative item. The perturbed profile and new recommendations are shown at the bottom.

expect that a user not liking an item which is generally very popular would have a big impact on their placement in the latent space used to generate recommendations. For example, consider a perturbation that calls for the removal of *Star Wars*, *Titanic*, and other extremely popular movies. We would expect this would have a big impact on the ranking of all items, so perhaps it is counterfactually correct. But at the same time, as an explanation, a list of such items might not make any sense to the user. How helpful is an explanation that tells the user that, if they had not watched these extremely popular items, the ranking of their recommendations might be different? They might find this counterfactual extremely

unlikely.

This line of reasoning suggests that popular items might not be the best items for counterfactual explanation. To study this phenomenon, we created a version of CLXR-joint in which the explainer is trained on filtered user profiles in which the top 15% most popular items are removed. We call this version *CLXR-filter*. The recommendation model is unchanged.

Table 4 shows the Coverage and MPRR results for both datasets and both recommenders. The most notable effect is the drop in coverage. This indicates that popular items are forming a large part of many of the perturbations CLXR-joint is producing. It is possible these would be considered low quality explanations in the eyes of users. There are small MPRR increases, except in the VAE / Yahoo condition, meaning that when explanations can be found, the system is finding similar size perturbations among the lower popularity items. These results suggest that item popularity does interact with our counterfactual explanation model but that the model can still function even without popular items that might be less convincing as explanations. Further studies, especially with human subjects, will be needed to understand how popularity interacts with the perceived quality of perturbation-based explanations.

Recommender	Explainer	ML-1M		Yahoo	
		Coverage ↑	MPRR$_{\%}$ ↓	Coverage ↑	MPRR$_{\%}$ ↓
MF	CLXR-joint	81%	25.3%	87.3%	25%
	CLXR-filter	56.3%	25.8%	24%	27%
VAE	CLXR-joint	93.3%	22.3%	80.5%	30%
	CLXR-filter	81.1%	22.5%	26%	24%

Table 4
Comparison of CLXR-joint (as above) and CLXR-filter

7. Conclusion and Future Work

In this paper, we tackle the recommendation explanation problem of "Comparing Item Rankings" from [3]. Starting from the LXR counterfactual explanation technique originally developed for explaining single items in recommendation list [14], we developed a family of techniques for the comparative explanation task. Of these, we show that a combined objective with a Siamese network has the best performance.

To address **RQ1**, we explored whether counterfactual learning can be leveraged to approximate profile perturbations that yield meaningful contrastive ranking explanations. Building upon the LXR framework originally designed for explaining single-item recommendations [14], we extend the method to the comparative setting introduced in [3]. Our results show that counterfactual profile perturbations are indeed effective in generating contrastive explanations, and that a model trained with a combined contrastive objective in a Siamese network architecture performs particularly well in approximating the necessary changes to reverse item rankings. An important open challenge in this context is the *optimality* of the generated perturbations. While our current method provides efficient approximations, future work should investigate the gap between these and truly minimal explanations, potentially via brute-force analysis on small-scale datasets.

We showed that explicitly optimizing for comparative explanations leads to improved performance, answering **RQ2**. By jointly optimizing for the preference between target and comparative items, CLXR-joint produces more focused and faithful explanation masks than the methods that are not optimized for a contrastive loss.

RQ3 focuses on the generalizability of our methods across different datasets and recommendation algorithms. Our experiments on multiple benchmark datasets revealed consistent overall trends, but also uncovered some dataset-specific behaviors. For example, popularity effects are more evident in cases where target items are frequently top-ranked, suggesting that fixed target positions may introduce certain biases. These observations highlight the need for further investigation into how the rank of the

target item and the characteristics of the underlying algorithm influence explanation behavior.

Finally, while our current study is focused on technical and empirical evaluation, understanding the quality and usability of comparative explanations from a user perspective remains a critical next step. The off-line metrics developed here help us understand the performance of our different explainer models in technical terms, but these need to be complemented with human-centered evaluation. We are planning a follow-up user study to assess whether the explanations produced by these methods are readily comprehensible, actionable, and aligned with the types of "why this, not that" questions real users might ask.

As we have seen, counterfactual perturbations must sometimes be large in order to achieve the desired reversal of rank — affecting dozens of items rather than just a few. We expect that techniques such as clustering will help generalize over larger perturbations. For instance, an explanation summarizing over a large set of items might state: “If you had not watched 25 Marvel action films, we would not have recommended *Aquaman and the Lost Kingdom* over *Batman Returns*.” Implementing such explanations would necessitate additional summarization mechanisms, which we plan to explore in future work.

8. Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (OpenAI) for grammar and spelling checks, paraphrasing and rewording, improving writing style, and programming assistance. After using this tool, the authors reviewed and edited the generated content and code as needed, validated the resulting implementation, and take full responsibility for the publication’s content and results.

References

[1] N. Tintarev, J. Masthoff, Explaining recommendations: Design and evaluation, in: Recommender systems handbook, Springer, 2015, pp. 353–382.

[2] N. Tintarev, J. Masthoff, Beyond explaining single item recommendations, in: Recommender Systems Handbook, Springer, 2022, pp. 711–756.

[3] M. Varasteh, E. McKinnie, A. Aird, D. Acuña, R. Burke, Comparative explanations for recommendation: Research directions, in: Proceedings of the 11th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS 2024), 2024, pp. 1–14. URL: https://ceur-ws.org/Vol-3815/paper1.pdf.

[4] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38.

[5] A. Tversky, S. Sattath, P. Slovic, Contingent weighting in judgment and choice., Psychological review 95 (1988) 371.

[6] I. Simonson, A. Tversky, Choice in context: Tradeoff contrast and extremeness aversion, Journal of marketing research 29 (1992) 281–295.

[7] A. Yang, N. Wang, R. Cai, H. Deng, H. Wang, Comparative explanations of recommendations, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 3113–3123.

[8] A. Jacovi, S. Swayamdipta, S. Ravfogel, Y. Elazar, Y. Choi, Y. Goldberg, Contrastive explanations for model interpretability, 2021. URL: https://arxiv.org/abs/2103.01378. arXiv:2103.01378.

[9] L. Malandri, F. Mercorio, M. Mezzanzanica, A. Seveso, Model-contrastive explanations through symbolic reasoning, Decision Support Systems 176 (2024) 114040.

[10] A. Castelnovo, R. Crupi, N. Mombelli, G. Nanino, D. Regoli, Evaluative item-contrastive explanations in rankings, Cognitive Computation 16 (2024) 3035–3050.

[11] Y. Wang, X. Wang, "why not other classes?": Towards class-contrastive back-propagation explanations, Advances in Neural Information Processing Systems 35 (2022) 9085–9097.

[12] R. Luss, E. Miehling, A. Dhurandhar, Cell your model: Contrastive explanations for large language models, 2025. URL: https://arxiv.org/abs/2406.11785. arXiv:2406.11785.

[13] J. v. d. Waa, M. Robeer, J. v. Diggelen, M. Brinkhuis, M. Neerincx, Contrastive explanations with local foil trees, in: Proceedings of the International Conference on Machine Learning (ICML) Workshop on Human Interpretability (WHI) in Machine Learning, 2018, pp. 41–47.

[14] O. Barkan, V. Bogina, L. Gurevitch, Y. Asher, N. Koenigstein, A counterfactual framework for learning and evaluating explanations for recommender systems, in: Proceedings of the ACM on Web Conference 2024, 2024, pp. 3723–3733.

[15] P. Grice, Studies in the Way of Words, Harvard University Press, 1991.

[16] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).

[17] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[18] S. Verma, V. Boonsanong, M. Hoang, K. Hines, J. Dickerson, C. Shah, Counterfactual explanations and algorithmic recourses for machine learning: A review, ACM Computing Surveys 56 (2024) 1–42.

[19] V. Kaffes, D. Sacharidis, G. Giannopoulos, Model-agnostic counterfactual explanations of recommendations, in: Proceedings of the 29th ACM conference on user modeling, adaptation and personalization, 2021, pp. 280–285.

[20] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, Y. Zhang, Counterfactual explainable recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1784–1793.

[21] J. Zhong, E. Negre, Shap-enhanced counterfactual explanations for recommendations, in: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 2022, pp. 1365–1372.

[22] M. Baklanov, V. Bogina, Y. Elisha, Y. Schein, L. Allerhand, O. Barkan, N. Koenigstein, Refining

fidelity metrics for explainable recommendations, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2025, pp. 2967–2971.
[23] A. R. Mohammadi, A. Peintner, M. Müller, E. Zangerle, Beyond top-1: Addressing inconsistencies in evaluating counterfactual explanations for recommender systems, in: Proceedings of the Nineteenth ACM Conference on Recommender Systems, 2025, pp. 515–520.
[24] O. Barkan, Y. Schein, Y. Elisha, V. Bogina, M. Baklanov, N. Koenigstein, Fidelity-aware recommendation explanations via stochastic path integration, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 40, 2026, pp. 14484–14492.
[25] M. Salimiparsa, Counterfactual explanations for rankings, in: Canadian AI, 2023.
[26] J. Singh, A. Anand, Posthoc interpretability of learning to rank models using secondary training data, arXiv preprint arXiv:1806.11330 (2018).
[27] J. Singh, A. Anand, Exs: Explainable search using local model agnostic interpretability, in: Proceedings of the twelfth ACM international conference on web search and data mining, 2019, pp. 770–773.
[28] J. Singh, A. Anand, Model agnostic interpretability of rankers via intent modelling, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020, pp. 618–628.
[29] D. Rennings, L. Lyu, A. Anand, Listwise explanations for ranking models using multiple explainers, in: Advances in Information Retrieval-45th European Conference on IR Research, ECIR, 2023.
[30] L. Gurevitch, V. Bogina, O. Barkan, Y. Schein, Y. Elisha, N. Koenigstein, Lxr: Learning to explain recommendations, ACM Transactions on Recommender Systems 4 (2025) 1–39.
[31] F. Croce, M. Hein, Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, in: International conference on machine learning, PMLR, 2020, pp. 2206–2216.
[32] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.
[33] D. Jannach, L. Lerche, I. Kamehkhosh, M. Jugovac, What recommenders recommend: an analysis of recommendation biases and possible countermeasures, User Modeling and User-Adapted Interaction 25 (2015) 427–491.
[34] M. Varasteh, M. S. Nejad, H. Moradi, M. A. Sadeghi, A. Kalhor, An improved hybrid recommender system: Integrating document context-based and behavior-based methods, in: 2023 31st International Conference on Electrical Engineering (ICEE), IEEE, 2023, pp. 881–887.
[35] S. Rendle, W. Krichene, L. Zhang, Y. Koren, Revisiting the performance of ials on item recommendation benchmarks, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 427–435.
[36] S. Rendle, W. Krichene, L. Zhang, J. Anderson, Neural collaborative filtering vs. matrix factorization revisited, in: Proceedings of the 14th ACM Conference on Recommender Systems, 2020, pp. 240–248.
[37] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 world wide web conference, 2018, pp. 689–698.
[38] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).
[39] M. T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.
[40] C. Nóbrega, L. Marinho, Towards explaining recommendations through local surrogate models, in: Proceedings of the 34th ACM/SIGAPP symposium on applied computing, 2019, pp. 1671–1678.
[41] K. H. Tran, A. Ghazimatin, R. Saha Roy, Counterfactual explanations for neural recommenders, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1627–1631.
[42] W. Cheng, Y. Shen, L. Huang, Y. Zhu, Incorporating interpretability into latent factor models via fast influence analysis, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 885–893.

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. URL: https://arxiv.org/abs/1412.6980. arXiv:1412.6980.

[44] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, The impact of popularity bias on fairness and calibration in recommendation, 2019. URL: https://arxiv.org/abs/1910.05755. arXiv:1910.05755.

[45] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, E. Malthouse, User-centered evaluation of popularity bias in recommender systems, in: Proceedings of the 29th ACM conference on user modeling, adaptation and personalization, 2021, pp. 119–129.