Revisiting XRec: How Collaborative Signals Influence LLM-Based Recommendation Explanations

Anonymous authors
Paper under double-blind review

Abstract

Recommender systems help users navigate large volumes of online content by offering personalized recommendations. However, the increasing reliance on deep learning-based techniques has made these systems opaque and difficult to interpret. To address this, XRec (Ma et al., 2024) was introduced as a novel framework that integrates collaborative signals and textual descriptions of past interactions into Large Language Models (LLMs) to generate natural language explanations for recommendations. In this work, we reproduce and expand upon the findings of Ma et al. (2024). While our results validate most of the original authors' claims, we were unable to fully replicate the reported performance improvements from injecting collaborative information into every LLM attention layer, nor the claimed effects of data sparsity. Beyond replication, our contributions provide evidence that the Graph Neural Network (GNN) component does not enhance explainability. Instead, the observed performance improvement is attributed to the Collaborative Information Adapter, which can act as a form of soft prompting, efficiently encoding task-specific information. This finding aligns with prior research suggesting that lightweight adaptation mechanisms can condition frozen LLMs for specific downstream tasks. Our implementation is open-source[1].

1 Introduction

Recommender systems (Resnick & Varian, 1997) help users filter through vast amounts of online content by providing personalized suggestions tailored to their preferences and interests (Cheng et al., 2021; 2022). Explainable recommendation models (Zhang et al., 2020) have emerged to enhance trust, decision-making, and satisfaction by providing explanations for their suggestions (Zhang, 2019).

Recent advancements in Large Language Models (LLMs) (e.g., Touvron et al. 2023; Grattafiori et al. 2024) have revolutionized the field of explainable recommendations. Earlier work, such as PETER (Li et al., 2021), used transformers to generate explanations, while more recent works, like PEPLER (Li et al., 2023), integrated GPT-2 (Radford et al., 2019) with user-item IDs as prompts to produce contextually rich and coherent justifications. Building on these foundations, LLM2ER (Yang et al., 2024b) introduced a personalized prompt module, overcoming the limitations of relying solely on user-item IDs for explanation generation.

A more recent study, *XRec: Large Language Models for Explainable Recommendation* (Ma et al., 2024), introduces a novel model-agnostic approach to address the limited availability of explanation data and the poor generalization of ID-based methods in zero-shot scenarios. This framework integrates collaborative signals—patterns of user interactions, such as reviews—into LLMs by injecting them into both the input prompt and the model layers, allowing XRec to extract meaningful insights from user preferences.

This paper attempts to reproduce and extend the main findings of Ma et al. (2024) through the following:

- **Reproducing XRec results:** We use the codebase of the original paper[2] to replicate its experiments, verify the authors' claims, and analyze the required computational resources.

[1] `https://anonymous.4open.science/r/ReXRec/`
[2] `https://github.com/HKUDS/XRec`

- **Extending the ablation study:** We investigate the role of collaborative signals by evaluating different GNN-based recommender systems that capture such signals from graph structures, including LightGCN (He et al., 2020) and NGCF (Wang et al., 2019). Additionally, we test three ablations that entirely remove graph-based collaborative information. We extend the evaluation by using STS (Thakur et al., 2020) and introducing LLMScore.

- **Addressing data leakage:** During our analysis, we identified two instances of data leakage caused by the original data generation process. The first involves using the same review to generate both the ground truth and the input to the framework, while the second is caused by the inclusion of test set information within the training data. We propose solutions to address these issues and analyze their impact.

- **Code adaptation and improvement:** We modify and improve the original codebase to support open-source LLMs for dataset generation and evaluation, facilitating future extensions and reproducibility. Additionally, we add scripts for tasks like dataset creation and data-sparsity testing.

After performing the above steps, we verified that XRec (Ma et al., 2024) outperforms baseline models. Our findings also confirm that incorporating user and item profile information improves performance and that both the profile and the injection component can contribute to improved personalization. However, we encountered challenges in reproducing the claims that injecting collaborative information across all LLM attention layers improves performance and that XRec is more effective under increased data sparsity.

Furthermore, our in-depth analysis of the impact of collaborative information reveals that the performance improvement in XRec (Ma et al., 2024) can be attributed to the Collaborative Information Adapter. We hypothesize this occurs because (1) collaborative signals may be redundant when unique identification features are available, even if randomly assigned, since the adapter can learn these signals as effectively as GNN-based models and (2) the adapter functions as a form of soft prompting, efficiently encoding task-specific information (Lester et al., 2021; Li et al., 2021).

While reproducing the datasets proposed by Ma et al. (2024), we identified two instances of data leakage, which we subsequently addressed in our attempt to reproduce the datasets. Despite the presence of these data leakage cases, our observations indicate that they did not significantly affect the overall claims or results.

Overall, our replication efforts yielded mixed results. While some performance trends aligned with the original study, some inconsistencies emerged. Moreover, performance gains were mainly driven by the Collaborative Information Adapter. This suggests that XRec's effectiveness depends on lightweight adaptation –whether through implicit learning of collaborative relationships or by soft prompting – rather than on explicit integration of collaborative information.

2 Scope of reproducibility

Ma et al. (2024) introduced XRec, a post-hoc technique that improves recommender system explainability by providing user-level personalization. Ma et al. (2024) make five key claims, which we investigate:

- **Claim 1: XRec improves over baselines in explainability and stability.** The model improves explanations in recommender systems by integrating collaborative information and textual descriptions into LLMs, outperforming state-of-the-art baselines in both explainability and stability.

- **Claim 2: Collaborative information injection improves explainability and stability.** XRec achieves higher accuracy by injecting collaborative information across all attention layers of the LLM, rather than simply passing it as input.

- **Claim 3: User and item profiles improve explainability and stability.** Adding user and item profiles to the input prompt improves explainability and stability by providing contextual information.

- **Claim 4: XRec has superior performance with increasing sparsity.** The model demonstrates robust performance across varying sparsity levels, showing improved results as user frequency decreases. Moreover, in zero-shot scenarios, its performance is comparable to other sparsity levels.
- **Claim 5: The model generates personalized explanations.** XRec generates customized explanations for each distinct user-item interaction, offering users meaningful and unique insights.

Moreover, we extend the work by answering the following research questions:

- **RQ 1:** How do Graph Neural Network (GNN)-based collaborative signals and the Collaborative Information Adapter influence the explainability and stability of XRec?
- **RQ 2:** What are the effects of dataset reproduction in XRec on explainability and stability, especially when addressing sources of data leakage?

3 Methodology

This section covers XRec's architecture (Section 3.1) and the metrics used to evaluate it (Section 3.2).

3.1 Model description

XRec (Ma et al., 2024) is a framework that uses a Large Language Model (LLM) to generate natural language explanations of why a recommender system suggested an item to a user. Figure 1 shows the framework with its four main components: a *Graph Neural Network (GNN)-based recommender system*, a *Collaborative Information Adapter*, *User and Item Text-Based profiles*, and an *Attention Injection Mechanism*.

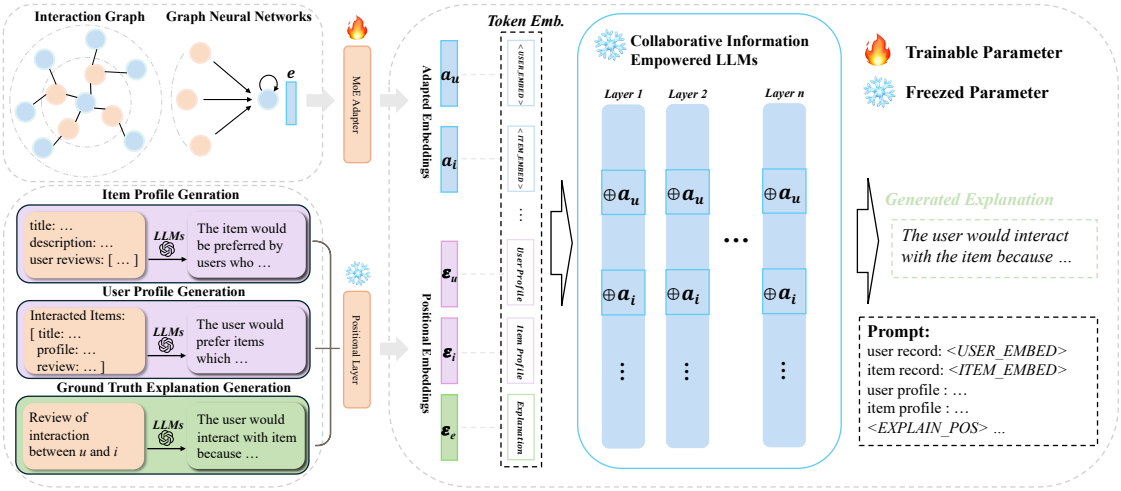

Figure 1: Overall XRec framework. XRec consists of 4 components: (i) **GNN-based recommender system** (top left): embeds user-item interactions. (ii) **Collaborative Information Adapter** (MoE Adapter): transforms user and item embeddings to LLM input space. (iii) **User and Item Text-Based Profiles** (bottom left): text profile of users and items generated by a LLM (iv) **Attention Injection Mechanism**: injects collaborative information directly into each LLM attention layer. Reprinted from Ma et al. (2024).

GNN-based recommender system The GNN-based recommender system models user-item interactions as a graph, with users and items as nodes and interactions as edges. GNNs embed users and items into fixed-size vectors using a message-passing mechanism (Gilmer et al., 2017), which aggregates information from neighboring nodes. XRec employs LightGCN (He et al., 2020), a model with a simple message-passing

mechanism which lacks linear transformations and nonlinear activations during aggregation. This is shown by Eq. 1, where $\mathbf{e}_x^{(l)}$ is the embedding of node x at layer l, and $\mathcal{N}_x$ is the set of neighbors of node x.

$$\mathbf{e}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(l)} \tag{1}$$

To investigate how the GNN affects XRec's performance, we experiment by replacing LightGCN with NGCF (Wang et al., 2019). NGCF has a more complex message-passing mechanism that uses nonlinear activations and learnable weight matrices ($\mathbf{W}_1, \mathbf{W}_2$), as shown in Eq. 2.

$$\mathbf{e}_u^{(k+1)} = \sigma\left(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \left(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2\left(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}\right)\right)\right) \tag{2}$$

Both GNNs are trained with the Bayesian Personal Ranking loss (Rendle et al., 2009) (Eq. 3), which favors higher scores for observed interactions. ℓ_2 regularization is also applied. Here, L denotes the loss function, M is the number of users, $\hat{y}_{ui}$ refers to the predicted rating of user u for item i, σ represents the sigmoid function, $\mathbf{E}^{(0)}$ denotes the embedding matrix, and λ is the ℓ_2 regularization parameter.

$$L = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\mathbf{E}^{(0)}\|^2 \tag{3}$$

Collaborative Information Adapter To effectively incorporate the collaborative information learned by the recommender system into the LLM, XRec uses an adapter which transforms the user and item embeddings into the input space of the LLM. This adapter uses a Mixture-of-Experts (MoE) architecture which effectively captures diverse aspects of the collaborative information.

User and Item Text-Based Profiles In addition to the adapted GNN embeddings, XRec takes as input textual profiles of users and items. Item profiles are created by prompting an LLM to summarize each item's description, while user profiles are generated by summarizing descriptions of items the user has interacted with. XRec's input is constructed as shown in Figure 1, where <USER_EMBED> and <ITEM_EMBED> are the embeddings generated by the GNN.

Attention Injection Mechanism A potential issue with prompt construction is that collaborative information, which is crucial for capturing user-item relationships, is limited to a single token. This could reduce its effectiveness in longer prompts and deeper layers. To address this, an attention injection mechanism is used, where collaborative information is directly integrated into the key, query, and value of all self-attention layers. This is shown in Eq. 4, where $\mathbf{W}$ is the projection matrix, $\mathbf{a}_i$ represents the adapted user or item embedding, and $f_{\{q,k,v\}}^{\text{original}}$ and $f_{\{q,k,v\}}^{\text{new}}$ denote the query, key, and value before and after injection, respectively.

$$f_{\{q,k,v\}}^{\text{new}}(\mathbf{x}_i) = f_{\{q,k,v\}}^{\text{original}}(\mathbf{x}_i) + \mathbf{W}_{\{q,k,v\}} \cdot \mathbf{a}_i \tag{4}$$

3.2 Evaluation metrics

We evaluate the model using LLM-based metrics, as they better reflect human judgment and capture context-aware semantic similarity, rather than relying solely on token-level analysis. First, we use GPTScore (Wang et al., 2023) to ensure a fair comparison with the baseline model performance reported by Ma et al. (2024). GPTScore prompts the gpt-3.5-turbo model from OpenAI to assess the semantic similarity between ground truth and generated explanations. GPTScore's use of a closed-source model limits accessibility and reproducibility and introduces model-specific bias. To address these limitations, we propose LLMScore, which averages the scores of multiple open-source LLMs, to mitigate model-specific biases and improve accessibility. Scores are obtained over five runs to ensure stability. The models used are *Llama 3.1 8b-instruct*, *Llama 3.2 3b-instruct* (Grattafiori et al., 2024), *Gemma2 9b-it* (Team et al., 2024), and *Qwen2.5 7b-instruct* (Yang et al., 2024a). The prompt used for scoring is provided in Appendix A.

In addition to LLMScore, we employ several established evaluation measures to capture different aspects of explanation quality. **BERTScore** (Zhang et al., 2019) measures the cosine similarity of BERT (Kenton & Toutanova, 2019) embeddings, allowing for a fine-grained token-level comparison that accounts for synonymy and paraphrasing. Complementing this, **BARTScore** (Yuan et al., 2021) measures the likelihood of regenerating ground-truth text using pre-trained BART (Lewis, 2019), making it effective for assessing fluency and coherence. To further align with human judgment, we include **BLEURT** (Sellam et al., 2020), a fine-tuned BERT model trained on human-annotated text pairs, which enhances sensitivity to subtle differences in meaning.

Beyond semantic quality, we assess personalization by measuring the diversity of generated explanations. We use **USR (Unique Sentence Ratio)** (Li et al., 2020) - the ratio of unique to total explanations, providing insight into the variability of generated content. To build on this, we introduce **STS (Sentence Transformer Similarity)** (Thakur et al., 2020), which uses SBERT (Reimers, 2019) embeddings and cosine similarity to evaluate the semantic similarity of generated explanations.

Finally, to compare the performance of LightGCN (He et al., 2020) and NGCF (Wang et al., 2019), we employ four metrics: **Precision@K** measures the proportion of relevant items among the top K recommendations, indicating accuracy; **Recall@K** measures the proportion of relevant items retrieved, indicating coverage; **NDCG@K (Normalized Discounted Cumulative Gain)** measures ranking quality by weighing relevance and position, prioritizing relevant items appearing earlier and **MRR@K (Mean Reciprocal Rank)** focuses on the rank of the first relevant item, rewarding higher-ranked relevant items. We choose $\mathbf{K} = 20$ by default.

4 Results

This section outlines our experimental set-up and results. We begin by describing the datasets used (Section 4.1), followed by an explanation of our training configurations (Section 4.2). We proceed with a detailed reproduction of the experiments from Ma et al. (2024) (Section 4.3) and conclude with an extended analysis that uncovers new insights beyond the original study (Section 4.4).

4.1 Datasets

Dataset processing To build XRec, Ma et al. (2024) use three datasets, each containing user reviews for various items: **Amazon Review Data**[3] (Ni et al., 2019), **Google Local Data**[4] (Li et al., 2022; Yan et al., 2023), and **Yelp Open Dataset**[5]. These were processed into what we refer to as **[Og]Amazon**, **[Og]Google**, and **[Og]Yelp** datasets. As we were unable to find the exact procedure used to create the datasets, we independently reproduce them and refer to them as **[Re]Amazon**, **[Re]Google**, and **[Re]Yelp**. Detailed processing steps for these datasets, along with information about data splitting, can be found in Appendix B. Table 1 presents a statistical comparison between the original and reproduced datasets.

Table 1: Dataset statistics from Ma et al. (2024) ([Og]) and our reproduced version ([Re]).

Dataset	[Og]Amazon	[Re]Amazon	[Og]Google	[Re]Google	[Og]Yelp	[Re]Yelp
#Users	15,349	15,069	22,582	19,503	15,942	15,962
#Items	15,247	15,028	16,557	18,998	14,085	14,085
#Interactions	360,839	350,644	411,840	400,038	393,680	393,680

Train-test overlap XRec is trained in two phases: (1) training the recommender system using the full interaction dataset, and (2) training the Collaborative Information Adapter using the explanation dataset (a subset of the interaction dataset). We observed overlap between the train, validation, and test splits of

[3] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

[4] https://jiachengli1995.github.io/google/index.html

[5] https://business.yelp.com/data/resources/open-dataset/

the original explanation and interaction datasets proposed in Ma et al. (2024), with overlap percentages provided in Appendix D. This issue is addressed in the reproduced datasets.

Ground-truth and profile generation For the original datasets, ground truth explanations are generated based on user reviews in order to make user preferences explicit. For item profiles, the Google and Amazon datasets use item names and descriptions, while the Yelp dataset also includes all user reviews of that item. User profiles are generated based on information from each item a user has interacted with, including the item's name, its associated profile, and the review the user left for that item. To improve dataset reproducibility, we employ Llama3.1 8B Instruct (Grattafiori et al., 2024) for generation, replacing the closed-source GPT model used in Ma et al. (2024).

Input contamination The user generation process introduces dataset leakage across all original datasets, while the Yelp item profile generation process introduces additional leakage specific to Yelp. This occurs because user reviews are used to generate both the ground-truth explanations and the input (profiles). This raises two issues: (1) It creates an overlap between the input and ground truth, potentially causing the model to rely on this overlap instead of learning meaningful patterns from user-item interactions, which hinders generalization during testing as this shortcut is unavailable (2) It compromises the evaluation process by leaking ground-truth test information into the test input profile. To address this in our reproduced datasets, we exclude all review information from input prompts used to generate profiles.

4.2 Training Configurations

The training process consists of two phases: training the GNN model and training the Collaborative Information Adapter module. The training process followed the hyperparameter configurations specified in Ma et al. (2024). An early stopping criterion is applied to the GNN-based models based on the Recall@20 metric. The LLM model used is LLaMA2-7B Touvron et al. (2023), which remains frozen throughout the training process. Appendix C shows the exact hyperparameter values for each phase.

Computational requirements: Training and inference are conducted using a single NVIDIA A100 40GB GPU, while evaluation is performed on two NVIDIA T4 16GB GPUs. Training takes approximately 15 hours for the Amazon and Google datasets and 12 hours for the Yelp dataset, for both the original and reproduced versions. Inference requires about 4 hours per dataset, and evaluation takes around 1.5 hours per dataset.

We estimate the carbon emissions from the experiments using the $CO_2e = CI \times PUE \times P \times t$ formula, where the carbon intensity (CI) in the Netherlands is 0.370 CO_2e emissions per kWh. The power usage effectiveness (PUE) values for the different setups are 1.2 and 1.1. The power consumption is 0.25 kW for A100 GPUs and 0.07 kW for T4 GPUs, with runtime durations of 280 hours and 160 hours, respectively. Based on these calculations, the estimated carbon emissions for the experiments amount to approximately 31 kg CO_2e and 4.5 kg CO_2e. In total, the experiments resulted in approximately 35.5 kg CO_2e emissions.

4.3 Results reproducing original paper

This section details our reproduction of the findings of Ma et al. (2024), where we retrain the XRec model and its ablations.. Our goal is to validate key claims regarding the model's explainability and stability, ensuring consistency with the original results. For comparisons with the baselines, we use GPTScore to ensure fairness, while for the ablation studies and other analyses throughout this section, we rely on LLMScore.

Claim 1: *XRec improves over baselines in explainability and stability.* We reproduce the results of Ma et al. (2024) by retraining (1) XRec on the original datasets, as well as (2) XRec (w/o Profile), where user and item profiles were excluded from the input prompts to ensure a fair comparison with the baselines that did not have access to profile information. Table 2 presents the original XRec (Ma et al., 2024) results alongside the scores from our reproduced models. Our reproduction shows an improvement in GPTScore and a decrease in GPT_{std} across all datasets. While we cannot determine the cause of this change given that the metric

Table 2: Comparison of baseline models and our reproduced results. We obtained the values for all baseline models and XRec (denoted as [Orig]XRec) from (Ma et al., 2024). Our results are denoted as XRec and in *italics*. The subscripts P, R, and F1 represent Precision, Recall, and F1 Score, respectively. The values in **bold** show the highest score, whereas the underlined show the second highest.

Metrics	Explainability ↑							Stability ↓					
	GPTScore	$BERT^{P}$	$BERT^{R}$	$BERT^{F_1}$	BARTScore	BLEURT	USR	GPT_{std}	$BERT^{P}_{std}$	$BERT^{R}_{std}$	$BERT^{F_1}_{std}$	$BART_{std}$	$BLEURT_{std}$
[Og]Amazon													
Att2Seq†	76.08	0.3746	0.3624	0.3687	-3.9440	-0.3302	0.7757	12.56	0.1691	0.1051	0.1275	0.5080	0.299
NRT†	75.63	0.3444	0.3440	0.3443	-3.9806	-0.4073	0.5413	12.82	0.1804	0.1035	0.1321	0.5101	0.3104
PETER†	77.65	**0.4279**	0.3799	0.4043	-3.8968	-0.2937	0.8480	11.21	0.1334	0.1035	0.1098	0.5144	0.2667
PETER+†	76.07	0.4119	0.3626	0.3876	-3.9647	-0.3293	0.4493	11.99	0.1576	0.1077	0.1245	0.5131	0.2805
PEPLER†	78.77	0.3506	0.3569	0.3543	-3.9142	-0.2950	0.9563	11.38	0.1105	0.0935	0.0893	0.5064	0.2195
[Orig]XRec†	82.57	0.4193	**0.4038**	**0.4122**	-3.8035	**-0.1061**	**1.0000**	9.60	0.0836	0.0920	0.0800	0.4832	**0.1780**
[Orig]XRec (w/o profile)†	81.77	0.4194	0.4004	0.4106	-3.8218	-0.1294	**1.0000**	9.60	0.0819	0.0955	0.0786	**0.4799**	0.1803
XRec	***83.75***	*0.4139*	*0.4037*	*0.4095*	***-3.7773***	*-0.1793*	***1.0000***	*9.40*	*0.0863*	*0.0933*	*0.0817*	*0.4985*	*0.2256*
XRec (w/o profile)	*83.28*	*0.4207*	*0.3967*	*0.4094*	*-3.8328*	*-0.1839*	***1.0000***	***9.16***	***0.0802***	***0.0888***	***0.0772***	*0.4801*	*0.2178*
[Og]Yelp													
Att2Seq†	63.91	0.2099	0.2658	0.2379	-4.5316	-0.6707	0.7583	15.62	0.1583	0.1074	0.1147	0.5616	0.247
NRT†	61.94	0.0795	0.2225	0.1495	-4.6142	-0.7913	0.2677	16.81	0.2293	0.1134	0.1581	**0.5612**	0.2728
PETER†	67.00	0.2102	0.2983	0.2513	-4.4100	-0.5816	0.8750	15.57	0.3315	0.1298	0.2230	0.5800	0.3555
PETER+†	67.98	0.2594	0.3097	0.2833	-4.3973	-0.5355	0.8637	13.80	0.2522	0.1174	0.1701	0.5665	0.3421
PEPLER†	67.54	0.2920	0.3183	0.3052	-4.4563	-0.3354	0.9143	14.18	0.1476	**0.1044**	0.1050	0.5777	0.2524
[Orig]XRec†	74.53	**0.3946**	**0.3506**	**0.3730**	-4.3911	**-0.2287**	**1.0000**	11.45	**0.0969**	0.1048	**0.0852**	0.5770	0.2322
[Orig]XRec (w/o profile)†	71.81	0.3879	0.3427	0.3657	-4.4035	-0.2486	**1.0000**	12.71	0.1087	0.1072	0.0919	0.5717	**0.2272**
XRec	***79.72***	*0.3541*	*0.3504*	*0.3527*	***-4.2969***	*-0.3089*	***1.0000***	***10.65***	*0.1126*	*0.1108*	*0.0967*	*0.5758*	*0.2334*
XRec (w/o profile)	*75.68*	*0.3814*	*0.3451*	*0.3636*	*-4.3717*	*-0.3303*	***1.0000***	*11.13*	*0.1218*	*0.1110*	*0.1014*	*0.5676*	*0.2514*
[Og]Google													
Att2Seq†	61.31	0.3619	0.3653	0.3636	-4.2627	-0.4671	0.5070	17.47	0.1855	0.1247	0.1403	0.6663	0.3198
NRT†	58.27	0.3509	0.3495	0.3496	-4.2915	-0.4838	0.2533	19.16	0.2176	0.1267	0.1571	0.6620	0.3118
PETER†	65.16	0.3892	0.3905	0.3881	-4.1527	-0.3375	0.4757	17.00	0.2819	0.1356	0.2005	0.6701	0.3272
PETER+†	66.74	0.4125	0.3975	0.4047	-4.1273	-0.3467	0.4887	15.23	0.1893	0.1244	0.1411	0.6515	0.3095
PEPLER†	69.71	0.3806	0.4093	0.3987	-4.1542	-0.2047	0.8660	17.11	0.1602	0.1154	0.1353	0.6800	0.3114
[Orig]XRec†	69.12	0.4546	0.4069	0.4311	-4.1647	-0.2437	0.9993	14.24	**0.0972**	**0.1031**	0.0937	**0.5700**	**0.2114**
[Orig]XRec (w/o profile)†	69.71	0.4427	0.4187	0.4310	**-4.1142**	-0.2026	0.997	14.09	0.1180	0.1171	0.1034	0.6465	0.2439
XRec	*71.13*	***0.4632***	*0.4025*	*0.4331*	*-4.1686*	*-0.2366*	*0.9993*	*13.22*	*0.0986*	*0.1163*	*0.0935*	*0.6560*	*0.2509*
XRec (w/o profile)	***71.66***	*0.4594*	***0.4191***	***0.4395***	*-4.1237*	***-0.1829***	***1.000***	***13.05***	*0.1001*	*0.1150*	***0.0931***	*0.6529*	*0.2392*

† This score was taken from Ma et al. (2024).

relies on a closed-source model[6], we observe that the general performance trends remain consistent with the original results.

For BERT-based metrics, BARTScore, and BLEURT, the results remain consistent with the original trends, with minor negligible variations. These metrics confirm that XRec maintains high explainability and stability, as the best and second-best scores are consistently achieved by our reproduced models. Additionally, while XRec (w/o Profile) performs slightly worse than XRec, it still consistently outperforms the baselines. This suggests that even when restricted to the same level of information as the baselines, XRec has superior performance. Overall, these results support **Claim 1**, demonstrating that XRec outperforms the baselines in both explainability and stability across multiple evaluation metrics and datasets.

Table 3: Comparison of GPTScore and LLMScore when evaluating the full XRec model and XRec without profile on the original datasets. LLMScore results are averaged over five runs.

Dataset	Model	GPTScore	LLMScore
[Og]Amazon	XRec	83.75	67.53 ± 11.65
	XRec (w/o profile)	83.28	66.95 ± 11.65
[Og]Yelp	XRec	79.72	61.67 ± 12.61
	XRec (w/o profile)	75.68	59.82 ± 12.11
[Og]Google	XRec	71.13	55.81 ± 13.04
	XRec (w/o profile)	71.66	56.49 ± 12.98

To verify this claim, we use GPTScore for a fair comparison with the baselines from Ma et al. (2024). However, reproducing results is challenging since GPTScore is closed-source. Instead, we propose LLMScore, which employs multiple open-source language models for scoring. Table 3 shows that LLMScore mirrors GPTScore's

[6]This prevents us from verifying whether the same model version and internal system prompt were used in both cases.

patterns: best performance on Amazon, worst on Google, and reduced performance without profiles except on Google. Therefore, LLMScore is the primary evaluation metric for the remaining experiments.

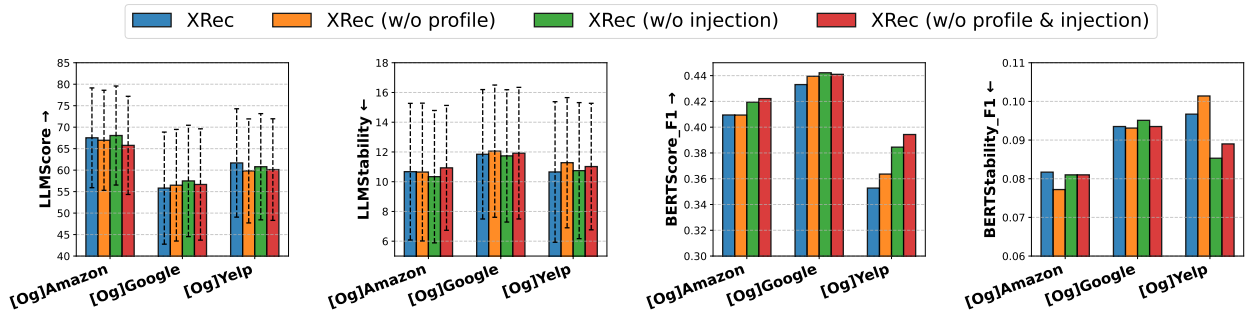

Figure 2: Ablation study results of the original datasets on the full model and three variants: without profile, without injection and without both.

Claim 2: *Collaborative information injection improves explainability and stability.* To evaluate this claim, we conducted ablation studies using four configurations: (1) XRec, (2) XRec (w/o profile), where user and item profiles are not given as input (3) XRec (w/o injection), where embeddings are not injected into the attention layers, and (4) XRec (w/o profile & injection). Following Ma et al. (2024), we evaluate these models using LLMScore and $BERT^{F_1}$.

Figure 2 shows that on LLMScore, XRec follows a similar trend as reported by Ma et al. (2024), outperforming the other configurations on Yelp and ranking second on Amazon. However, on the Google dataset, XRec performs the worst. Moreover, in terms of $BERT^{F_1}$, XRec consistently ranks last or ties for the worst performance across all three datasets. On the other hand, for LLMScore and LLMStability, XRec (w/o injection) outperforms all other ablations on two datasets (Amazon and Google), and ranks second on the other dataset (Yelp). These results contradict **Claim 2** and suggest that, on average, injection reduces performance rather than enhancing it.

One possible explanation for the lack of performance improvement from injecting collaborative information lies in the characteristics of the underlying graph neural network (GNN). According to Qin et al. (2024), injection is most effective when the model can disentangle the various latent factors that shape user preferences and item attributes. However, LightGCN (He et al., 2020) lacks this property, which may explain its negative impact on performance.

Claim 3: *User and item profiles improve explainability and stability.* We evaluate this claim with the ablation studies presented in Figure 2. XRec (w/o profile) performs worse than XRec in terms of LLMScore and LLMStability on Amazon and Yelp. Similarly, XRec (w/o injection & profile) underperforms across all three datasets compared to XRec (w/o injection). However, models without a profile exhibit a slight yet negligible improvement in $BERT^{F_1}$ compared to those that incorporate profiles. These findings support **Claim 3**, suggesting that integrating user and item profiles generally improves the explainability and stability of the system, although not significantly.

Claim 4: *XRec has superior performance with increasing sparsity.* We investigate the model's performance under varying levels of data sparsity. Specifically, as done in Ma et al. (2024), we divided the test data into six groups: one for zero-shot users (users absent from the training data) and five quantiles representing increasing ranges of user interactions. Appendix E provides interaction frequency ranges and statistics for each group. As shown in Figure 3, this grouping reveals that while BERTScore shows a slight improvement for users with sparser interactions—with stability remaining constant—the LLM-based metrics behave differently: explainability decreases and stability increases with growing sparsity. These conflicting trends suggest that **Claim 4** is not reproducible and warrants further investigation. Overall, however, the general trend indicates that performance improves with higher interaction frequency.

Claim 5: *The model generates personalized explanations.* Ma et al. (2024) quantify personalization using the Unique Sentence Ratio (USR) (Li et al., 2020). However, we argue that USR captures only lexical uniqueness

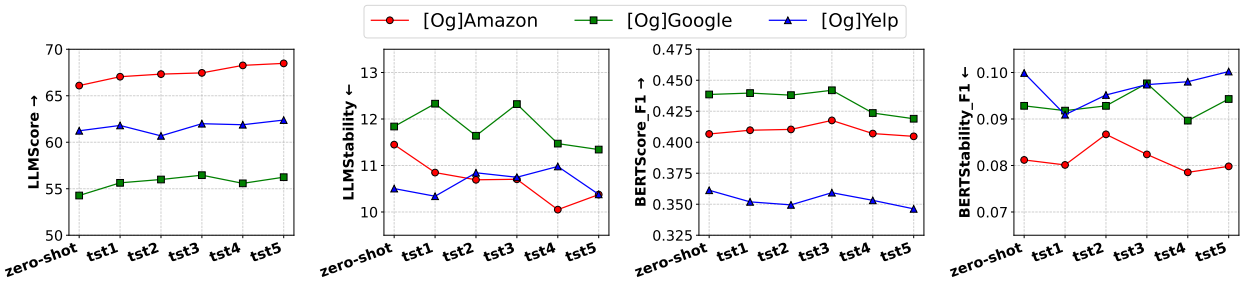

Figure 3: XRec's performance on six data splits with increasing sparsity level on the original datasets.

and fails to account for semantically equivalent sentences. For example, the sentences "I am going to the store" and "I am heading to the store" receive a USR score of 1.0, even though they are semantically identical. To test this hypothesis, we sampled user-item interactions from each dataset and generated 100 explanations per sample. In all cases, USR was 1.0 even though the explanations conveyed the same meaning. This suggests that USR is not an appropriate measure of personalization. To address this, we use Sentence Transformer Similarity (STS) (Thakur et al., 2020). We compute an aggregate STS score by averaging the pairwise STS values, providing an estimate of the overall personalization of generated explanations. On this dataset of 100 explanations, STS yields an average score of 0.78, indicating that the explanations are highly similar in meaning despite minor lexical differences. However, STS tends to assign higher similarity scores to sentences with similar structures, which is often the case with our generated explanations. Despite this limitation, STS provides a more meaningful measure of similarity than USR.

Table 4: Evaluation results of different XRec variations of the original datasets on the STS metric.

Dataset	XRec	XRec (w/o profile)	XRec (w/o injection)	XRec (w/o profile & injection)	XRec (w/o GNN)
[Og]Amazon	0.5667	0.6563	0.6614	0.6936	0.4139
[Og]Yelp	0.3802	0.3601	0.3671	0.4149	0.2759
[Og]Google	0.5891	0.5910	0.6006	0.5863	0.2807

Table 4 presents the average pairwise STS scores for each dataset. The [Og]Amazon and [Og]Google explanations exhibit overall high STS, suggesting a lower degree of personalization in comparison to the explanations for [Og]Yelp. This difference can be attributed to the higher level of detail in Yelp reviews compared to the other datasets. This shows that data quality affects the personalization capabilities of XRec. Additionally, we observe that for [Og]Amazon, incorporating user-profiles and injection improves personalization. However, for [Og]Google, these factors do not significantly impact the STS score. Finally, we notice that removing the GNN makes generated explanations more diverse. We hypothesize that this is due to the fact that when XRec is trained with the GNN component, the MoE adapter learns to prompt the model in such a way that the explanations follow a similar structure. As a result, removing this component allows the model to produce more diverse explanations, but at the cost of reduced explainability scores as shown in Table 2.

4.4 Results beyond original paper

This section explores additional aspects of XRec's performance. We analyze the impact of GNN collaborative signal, MoE adapter and assess the effect of data leakage mitigation. These extensions provide deeper insights into the model's capabilities and potential limitations.

4.4.1 Impact of GNN and Collaborative Information Adapter

RQ 1: *How do Graph Neural Network (GNN)-based collaborative signals and the Collaborative Information Adapter influence the explainability and stability of XRec?* To assess the impact of GNN-based collaborative signals on XRec, we replace its originally used LightGCN (He et al., 2020) GNN component with NGCF (Wang et al., 2019), referring to this variant as XRec w/ NGCF. For clarity, we refer to XRec with its

original GNN architecture as XRec w/ LightGCN in this section. Additionally, we evaluate three ablations that entirely exclude collaborative signals: one that replaces GNN outputs with fixed random embeddings (XRec w/o GNN, fixed), another that replaces them with variable random embeddings (XRec w/o GNN, random), and a third that removes both the GNN and the Collaborative Information Adapter (MoE) (XRec w/o GNN & MoE).

Table 5: Performance comparison of LightGCN and NGCF models. The best values are in **bold**.

Dataset	Model	Recall	NDCG	Precision	MRR
[Og]Amazon	LightGCN	**0.0719**	**0.0426**	**0.0175**	**0.0724**
	NGCF	0.0514	0.0319	0.0131	0.0587
[Og]Yelp	LightGCN	**0.0614**	**0.0382**	**0.0160**	**0.0677**
	NGCF	0.0571	0.0365	0.0156	0.0668
[Og]Google	LightGCN	**0.0791**	**0.0433**	**0.0129**	**0.0575**
	NGCF	0.0571	0.0327	0.0094	0.0455

First, we compare the performance of the two GNN-based recommender systems on the item recommendation task. Table 5 shows that LightGCN outperforms NGCF across all evaluation metrics and datasets.

We use the embeddings from these two models as input to the MoE Adapter. Table 6 shows that XRec (w/ NGCF) outperforms XRec (w/ LightGCN) on average, achieving higher BERT-based scores on two datasets and a higher LLMScore on one. This suggests that better performance in item recommendation does not necessarily translate to better explainability in XRec.

Table 6: Performance comparison of XRec with LightGCN and NGCF as collaborative information algorithms, as well as three variants where explicit collaborative information is removed. The subscripts P, R, and F1 represent Precision, Recall, and F1 Score, respectively. Values in **bold** indicate the highest score, while underlined values denote the second-best. LLMScore and LLMStability is averaged after five runs.

Metrics	**Explainability ↑**							**Stability ↓**					
	LLMScore	BERT ^{P}	BERT^{R}	BERT^{F_1}	BARTScore	BLEURT	STS	LLMStability	BERT$^{P}_{std}$	BERT$^{R}_{std}$	BERT$^{F_1}_{std}$	BART$_{std}$	BLEURT$_{std}$
[Og]Amazon													
XRec (w/ LightGCN)	67.53 ± 11.61	0.4139	0.4037	0.4095	**-3.7773**	-0.1783	0.57	10.67 ± 4.59	0.0863	0.0933	0.0817	0.4985	0.2256
XRec (w/ NGCF)	67.97 ± 11.66	0.4277	**0.4088**	**0.4164**	-3.7789	-0.1445	0.62	10.58 ± 4.57	0.0825	0.0934	0.0806	0.4913	0.2118
XRec (w/o GNN, fixed)	**68.05 ± 11.69**	0.4200	0.3963	0.4089	-3.8171	-0.1461	0.67	**10.19 ± 4.65**	**0.0077**	0.0899	**0.0764**	0.4766	0.2077
XRec (w/o GNN, random)	65.33 ± 11.81	**0.4361**	0.3984	0.4179	-3.8575	**-0.1264**	0.69	11.31 ± 4.38	0.0842	0.0940	0.0816	0.4861	**0.1990**
XRec (w/o GNN & MoE)	62.66 ± 12.14	0.2909	0.3102	0.3013	-4.0502	-0.4729	**0.42**	11.05 ± 4.29	0.1007	**0.0796**	0.0804	**0.4336**	0.2439
[Og]Yelp													
XRec (w/ LightGCN)	**61.67 ± 12.61**	0.3541	0.3504	0.3527	-4.2969	-0.3089	0.38	10.65 ± 4.73	0.1126	0.1108	0.0967	0.5758	0.2334
XRec (w/ NGCF)	60.95 ± 12.49	0.3828	**0.3635**	0.3736	**-4.2719**	-0.3053	0.33	10.67 ± 4.76	**0.0953**	**0.1047**	**0.0841**	0.5657	0.2128
XRec (w/o GNN, fixed)	61.20 ± 12.49	0.3737	0.3572	0.3658	-4.3070	-0.2971	0.36	**10.61 ± 4.78**	0.1103	0.1039	0.0897	0.5694	**0.2104**
XRec (w/o GNN, random)	59.34 ± 12.01	**0.4376**	0.3476	**0.3926**	-4.4425	**-0.2608**	0.45	11.00 ± 4.46	0.0967	0.1098	0.0877	0.5790	0.2112
XRec (w/o GNN & MoE)	58.68 ± 13.56	0.2796	0.2609	0.2710	-4.6124	-0.3615	**0.27**	11.51 ± 4.80	0.1045	0.1088	0.0953	**0.5392**	0.2174
[Og]Google													
XRec (w/ LightGCN)	55.81 ± 13.05	**0.4632**	0.4025	**0.4331**	-4.1686	-0.2366	0.59	**11.84 ± 4.35**	0.0986	0.1163	0.0935	0.6560	0.2509
XRec (w/ NGCF)	55.04 ± 13.29	0.4517	0.3943	0.4233	-4.2445	-0.2652	0.54	12.12 ± 4.30	0.1027	0.1136	0.0946	0.6558	0.2635
XRec (w/o GNN, fixed)	**57.25 ± 12.87**	0.4414	**0.4192**	0.4306	**-4.1158**	**-0.1776**	0.53	12.04 ± 4.44	0.1081	0.1141	0.0959	0.6488	0.2398
XRec (w/o GNN, random)	56.71 ± 12.89	0.4520	0.4094	0.4310	-4.1595	-0.1780	0.63	11.96 ± 4.45	**0.0958**	0.1187	0.0937	0.6542	**0.2390**
XRec (w/o GNN & MoE)	50.40 ± 14.29	0.2094	0.2806	0.2456	-4.5504	-0.5188	**0.28**	12.17 ± 4.58	0.1049	**0.1019**	**0.0918**	**0.5725**	0.2451

We observe that removing both the collaborative information and the adapter in XRec (w/o GNN & MoE) results in the lowest performance, highlighting the importance of these components. Additionally, we note that while explainability and stability decrease when the GNN and MoE components are removed, personalization, as measured by STS, improves.

However, when comparing with XRec (w/o GNN, fixed), which uses a random embedding vector instead of embeddings generated by GNNs, we do not observe any performance drop. Instead, we find that it achieves the best performance on two datasets and ranks second-best on the other. Meanwhile, XRec (w/o GNN, random) performs worse than models with the MoE adapter but better than those without it. This suggests that removing collaborative signals from the GNN has minimal impact on explainability and that the MoE effectively learns meaningful information, though it struggles when the input is random rather than fixed.

We hypothesize that this effect arises from two possible causes, though further research is needed to confirm the exact mechanism. First, the MoE adapter may interpret the embedding vector as a set of unique

identifiers, enabling it to implicitly learn collaborative signals even without GNN-based embeddings. The poorer performance of XRec (w/o GNN, random) supports this hypothesis, as the randomized inputs disrupt the model's ability to retain meaningful identification information. Second, both STS scores and manual inspection revealed structural differences in sentence composition between models with and without the MoE adapter. This suggests that incorporating the lightweight adapter, positioned almost as a prefix to the prompt, may unintentionally act as a form of soft prompting. This effect aligns with prior work in prompt tuning Lester et al. (2021); Li et al. (2021), which demonstrated that small, lightweight adapters can condition LLMs for downstream tasks without requiring full fine-tuning.

4.4.2 Effect of Data Leakage Mitigation

RQ 2: *What are the effects of dataset reproduction in XRec on explainability and stability, especially when addressing train-test overlap and the use of shared reviews for both profile generation and ground truth creation?* We identified two cases of data leakage in the original datasets of Ma et al. (2024): **Input Contamination**, where the same reviews are used to generate both ground-truth explanations and input profiles; and **Train-Test Leakage**, where training, validation, and test sets overlap. To assess the impact this leakage may have, we train XRec on both the original datasets and our reproduced versions, where these issues have been mitigated.

Table 7: XRec performance on (1) Original ([Og]) datasets with data leakage and (2) Reproduced ([Re]) datasets with mitigated leakage. Subscripts P, R, and F1 denote Precision, Recall, and F1 Score. The best values are in **bold**. LLMScore and LLMStability is averaged after five runs.

Metrics	Explainability ↑							Stability ↓					
	LLMScore	$BERT^{P}$	$BERT^{R}$	$BERT^{F_1}$	BARTScore	BLEURT	STS	LLMStability	$BERT^{P}_{std}$	$BERT^{R}_{std}$	$BERT^{F_1}_{std}$	$BART_{std}$	$BLEURT_{std}$
[Og]Amazon	**67.53 ± 11.61**	0.4139	0.4037	0.4095	**-3.7773**	-0.1793	**0.5667**	**10.67 ± 4.59**	**0.0863**	**0.0933**	**0.0817**	**0.4985**	**0.2256**
[Re]Amazon	64.80 ± 11.25	**0.4563**	**0.4218**	**0.4396**	-3.8135	**-0.1474**	0.7489	11.84 ± 3.99	0.0986	0.1049	0.0943	0.5583	0.2599
[Og]Yelp	61.67 ± 12.61	0.3541	0.3504	0.3527	-4.2969	-0.3089	**0.3802**	**10.65 ± 4.73**	0.1126	0.1108	0.0967	**0.5758**	0.2334
[Re]Yelp	**62.75 ± 11.89**	**0.4382**	**0.4207**	**0.4300**	**-4.0250**	**-0.1765**	0.5186	10.78 ± 4.52	**0.1002**	**0.1076**	**0.0937**	0.5813	**0.2176**
[Og]Google	55.81 ± 13.05	**0.4632**	0.4025	0.4331	-4.1686	-0.2366	**0.5891**	**11.84 ± 4.35**	**0.0986**	0.1163	**0.0935**	0.6560	0.2509
[Re]Google	**59.72 ± 12.45**	0.4450	**0.4238**	**0.4349**	**-3.9298**	**-0.1283**	0.6074	12.57 ± 4.43	0.1004	**0.1107**	0.0960	**0.6089**	**0.2451**

Table 7 demonstrates that mitigating data leakage generally improves the explainability of XRec, as reflected in higher BERTScore and BLEURT values. This result is counterintuitive since input contamination typically inflates performance by providing the model with direct access to ground-truth information. However, we cannot definitely attribute these improvements solely on data leakage mitigation, as other factors—such as variations in selected reviews, or differences in the models used to generate the ground truth and explanations—may have steered the results in this direction.

Figure 4 shows a similar data sparsity trend as Figure 3, indicating that data leakage does not impact the model's behaviour across sparsity levels. The consistency between these results indicates that eliminating data leakage does not alter the overall pattern observed in the model. In other words, mitigating leakage does not compromise the validity of the findings.

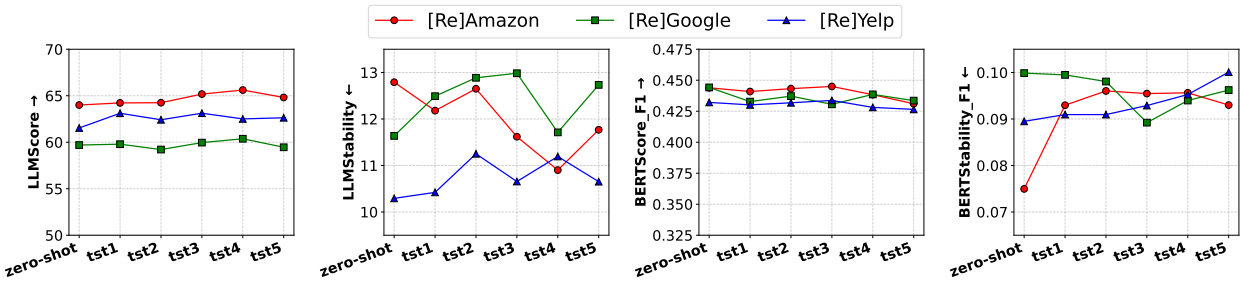

Figure 4: XRec's performance on six data splits with increasing sparsity level on the reproduced datasets.

5 Conclusion

Our reproduction study and analysis of XRec confirm that the model improves over baselines in both explainability and stability. While our results verify the impact of user and item profiles, they do not support the claim that collaborative information injection through the layers of the LLM improves explainability and stability, as XRec without injection performs better on two out of three datasets. Additionally, we find that explainability and stability improve with more information about the specific user, contradicting the claim that XRec performs better with increasing data sparsity. However, this suggests that the model learns user preferences. Lastly, our evaluation of personalization using both USR and STS yielded inconclusive results. We found out that while STS provides a more nuanced evaluation, it also favours similar writing styles, potentially misjudging semantically equivalent sentences written differently. We suggest future research into developing more reliable methods for evaluating personalization in explainable recommender systems.

Through additional experiments, we find that removing the GNN-based recommendation system has minimal impact on model explainability, as without it performance remains stable or slightly improves. We hypothesize that this occurs because (1) when distinct identification features of users and items are present—even if randomly assigned—the MoE can effectively capture collaborative patterns without relying on a GNN-based model, making those signals potentially redundant, and (2) the MoE adapter acts similarly to soft prompting, embedding task-specific knowledge in a way that enhances explainability (Lester et al., 2021; Li et al., 2021). The latter effect may be exacerbated by evaluation metrics that favour structurally similar sentences over semantic meaning. Future work should examine the MoE's influence on the structure and meaning of explanations, as well as explore more robust evaluation metrics.

Lastly, we attempted to reproduce the datasets while addressing the data leakage present in the original ones. Our experiments showed that removing data leakage led to improvements in model performance across most evaluation metrics. However, we cannot definitively attribute these improvements solely to data leakage mitigation, as other factors—such as variations in selected reviews, or differences in the models used to generate the ground truth and explanations—may have played a role. Based on our findings, we conclude that data leakage had no significant impact on model performance. However, due to limited resources, our experiments were not exhaustive. We recommend that future work conduct more comprehensive experiments to fully validate all claims from the original paper on the reproduced datasets.

What was easy Ma et al. (2024) provided most of the model implementation with documentation on running the training process. The model architecture description was detailed and easy to follow. Additionally, the authors provided evidence linked to specific claims, simplifying the reproduction process.

What was difficult Certain aspects of the reproduction study required more effort than anticipated. For instance, the provided code for dataset pre-processing was an incomplete example procedure for the Yelp dataset. Reproducing the full pipeline for all datasets required time and insight into the intended data requirements. The codebase also required modifications to run ablations and evaluations. Additionally, the documentation lacked details on evaluation metrics and dataset-specific configurations.

Communication with original authors They promptly responded to our questions and clarified details about the data generation process and the hyperparameters we could alter to replicate their datasets.

References

Min Cheng, Fei Yuan, Qingpeng Liu, Xia Xin, and Enhong Chen. Learning transferable user representations with sequential behaviors via contrastive pre-training. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 51–60. IEEE, 2021.

Min Cheng, Zhiyong Liu, Qingpeng Liu, Shuyuan Ge, and Enhong Chen. Towards automatic discovering of deep hybrid network architecture for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, pp. 1923–1932, 2022.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70, pp. 1263–1272, 2017.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, New York, NY, USA, 2020. ACM. doi: 10.1145/3397271.3401063.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1. Minneapolis, Minnesota, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Mike Lewis. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Jiacheng Li, Jingbo Shang, and Julian McAuley. Uctopic: Unsupervised contrastive learning for phrase representations and topic mining. *arXiv preprint arXiv:2202.13469*, 2022.

Lei Li, Yongfeng Zhang, and Li Chen. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 755–764, 2020.

Lei Li, Yongfeng Zhang, and Li Chen. Personalized transformer for explainable recommendation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4947–4957, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.383. URL https://aclanthology.org/2021.acl-long.383/.

Lei Li, Yongfeng Zhang, and Li Chen. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems (TOIS)*, 2023.

Qiyao Ma, Xubin Ren, and Chao Huang. Xrec: Large language models for explainable recommendation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 391–402, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.22. URL https://aclanthology.org/2024.findings-emnlp.22/.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Disentangled representation learning with large language models for text-attributed graphs, 2024. URL https://arxiv.org/abs/2310.18152.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pp. 452–461, 2009.

Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 2020.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL `https://arxiv.org/abs/2307.09288`.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*, 2023.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019.*, pp. 165–174, 2019.

An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, and Julian McAuley. Personalized showcases: Generating multi-modal explanations for recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2251–2255, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

Mengyuan Yang, Mengying Zhu, Yan Wang, Linxun Chen, Yilei Zhao, Xiuyuan Wang, Bing Han, Xiaolin Zheng, and Jianwei Yin. Fine-tuning large language model based explainable recommendation with explainable quality reward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8):9250–9259, 2024b. doi: 10.1609/aaai.v38i8.28777.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Yongfeng Zhang. Tutorial on explainable recommendation and search. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 255–256. ACM, 2019.

Yongfeng Zhang, Xu Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

A GPTScore Prompt

To acquire a GPTScore and LLMScore for each XRec explanation we use the following prompt proposed by Ma et al. (2024):

System Prompt:

```
Score the given explanation against the ground truth on a scale from 0 to 100,
focusing on the alignment of meanings rather than the formatting.
Provide your score as a number and do not provide any other text.
```

Prompt:

```
{
    "prediction" : <XREC explanation>,
    "reference"  : <ground truth explanation>
}
```

B Datasets

B.1 Dataset Processing

To evaluate XRec, Ma et al. (2024) use three datasets from distinct domains, which they subsequently process. The first dataset, **Amazon Review Data**[7], proposed by Ni et al. (2019), includes reviews and product metadata, where only the book subset is used for XRec. The second dataset, **Google Local Data**[8], introduced by Li et al. (2022); Yan et al. (2023), contains Google Maps reviews and business metadata, where only the California subset is used. Finally, the **Yelp Open Dataset**[9], includes reviews and business metadata, and the entire dataset serves as input.

Ma et al. (2024) provide example processing code for the **Yelp Open Dataset**. Their approach involves:

1. Filtering out items without a name, description, or categories;
2. Filtering out all reviews with a rating of 3 or less;
3. Randomly subsampling the remaining users;
4. Retaining only interactions within the 8-core interaction graph[10].

We follow these steps to reproduce the datasets, which we refer to as [Re]Amazon, [Re]Google, and [Re]Yelp to distinguish them from the original Amazon, Google, and Yelp datasets in XRec's repository. However, whereas Ma et al. (2024) use the full Google dataset and work with a manually extracted 8-core subgraph, we use the smaller 10-core version to reduce computational overhead. For consistency, we process the Amazon and Yelp datasets into 10-core versions.

The ground-truth explanations are generated from a subset of interactions, taken from the pre-processed dataset, following the original procedure in Ma et al. (2024). For the Amazon and Google datasets, we include reviews with more than 10 words in order to reduce noise.

[7] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

[8] https://jiachengli1995.github.io/google/index.html

[9] https://business.yelp.com/data/resources/open-dataset/

[10] A "k-core" of a graph is a subgraph in which every node has a degree of at least k, meaning each node is connected to at least k other nodes. For example, in an 8-core graph, every user or item must have at least 8 interactions to be included.

B.2 Data Splitting

We provide the Train, Validation, and Test splits for the reproduced interaction datasets—[Re]Amazon, [Re]Google, and [Re]Yelp—in Table 8. These interaction datasets are used to train the Recommender System. From each interaction dataset, we extract a subset containing interactions with ground truth explanations, which we refer to as the explanation dataset. This explanation dataset is used to train the XRec framework. To ensure consistency, the splits for the explanation dataset were designed to match the numbers of the corresponding original datasets (Ma et al., 2024), and the same split ratios were applied to the interaction datasets.

Table 8: Train, Validation, and Test splits for [Re]Amazon, [Re]Google, and [Re]Yelp datasets.

	[Re]Amazon		[Re]Google		[Re]Yelp	
Split	Explanations	Interactions	Explanations	Interactions	Explanations	Interactions
Train	95,841	303,246	94,663	345,846	74,212	337,797
Validation	11,980	37,905	11,833	43,231	9,277	42,227
Test	3,000	9,493	3,000	10,961	3,000	13,656
Sum	110,821	350,644	109,496	400,038	86,489	393,680

C Hyperparameters

The models were trained and evaluated using the parameters listed in 9:

Table 9: Hyperparameter configurations for the GNN-based recommender system and the Information Collaboration Adapter module (MoE), both optimized using Adam (Kingma & Ba, 2015).

Hyperparameter	**GNN**	**MoE**
Batch size	1024	1
Number of epochs	300	1
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Number of layers	4	-
Embedding size	64	-
Early Stopping Patience	10 epochs	-
Number of experts	-	8
Dropout rate	-	0.2
Gating Router Noise Factor	-	0.01

D Overlap Between Original Dataset Splits

Ma et al. (2024) train their system using two datasets: (1) the full interaction dataset (for training the recommendation system) and (2) the explanation dataset (a subset of the interaction dataset, used to train the Information Collaboration Adapter). The explanation dataset includes textual explanations of reviews corresponding to specific interactions. However, we observed that during the splitting of these datasets, overlaps occasionally occur, leading to instances where data from one dataset is exposed to the other. In this section, we present the percentage of overlap between the train, validation, and test splits of the explanation and interaction datasets. Tables 10, 11, and 12 summarize these overlaps for the Amazon, Google, and Yelp datasets introduced by Ma et al. (2024).

Table 10: Percentage of overlap between the splits of the Explanations and the Interactions Amazon datasets.

		Interactions		
		Train (%)	**Validation (%)**	**Test (%)**
Explanations	Train	79%	10%	10%
	Validation	79%	10%	9%
	Test	79%	9%	10%

Table 11: Percentage of overlap between the splits of the Explanations and the Interactions Google datasets.

		Interactions		
		Train (%)	**Validation (%)**	**Test (%)**
Explanations	Train	80%	9%	10%
	Validation	80%	9%	9%
	Test	78%	10%	10%

Table 12: Percentage of overlap between the splits of the Explanations and the Interactions Yelp datasets.

		Interactions		
		Train (%)	**Validation (%)**	**Test (%)**
Explanations	Train	79%	10%	10%
	Validation	80%	10%	9%
	Test	81%	9%	9%

E Data sparsity statistics

Table 13 provides the statistics of all sparsity sets used to reproduce **Claim 4**.

Table 13: Data sparsity set statistics including user interaction frequency intervals, and the percentage of data allocated to each sparsity set from the test set which includes 3000 interactions.

Sparsity Set	**zero-shot**	**tst1**	**tst2**	**tst3**	**tst4**	**tst5**
[Og]Amazon						
Interaction Frequency	$[0, 0]$	$[1, 9]$	$(9, 20]$	$(20, 42]$	$(42, 95]$	$(95, 548]$
Percentage of Test Set	12%	17%	18%	17%	18%	18%
[Og]Google						
Interaction Frequency	$[0, 0]$	$[1, 4]$	$(4, 7]$	$(7, 10]$	$(10, 17]$	$(17, 113]$
Percentage of Test Set	9%	20%	19%	16%	18%	18%
[Og]Yelp						
Interaction Frequency	$[0, 0]$	$[1, 4]$	$(4, 8]$	$(8, 14]$	$(14, 28]$	$(28, 262]$
Percentage of Test Set	10%	19%	20%	16%	17%	18%
[Re]Amazon						
Interaction Frequency	$[0, 0]$	$[1, 4]$	$(4, 6]$	$(6, 11]$	$(11, 28]$	$(28, 251]$
Percentage of Test Set	2%	27%	13%	19%	20%	19%
[Re]Google						
Interaction Frequency	$[0, 0]$	$[1, 3]$	$(3, 6]$	$(6, 9]$	$(9, 16]$	$(16, 110]$
Percentage of Test Set	2%	21%	26%	16%	17%	18%
[Re]Yelp						
Interaction Frequency	$[0, 0]$	$[1, 3]$	$(3, 5]$	$(5, 8]$	$(8, 15]$	$(15, 134]$
Percentage of Test Set	4%	28%	17%	15%	17%	19%