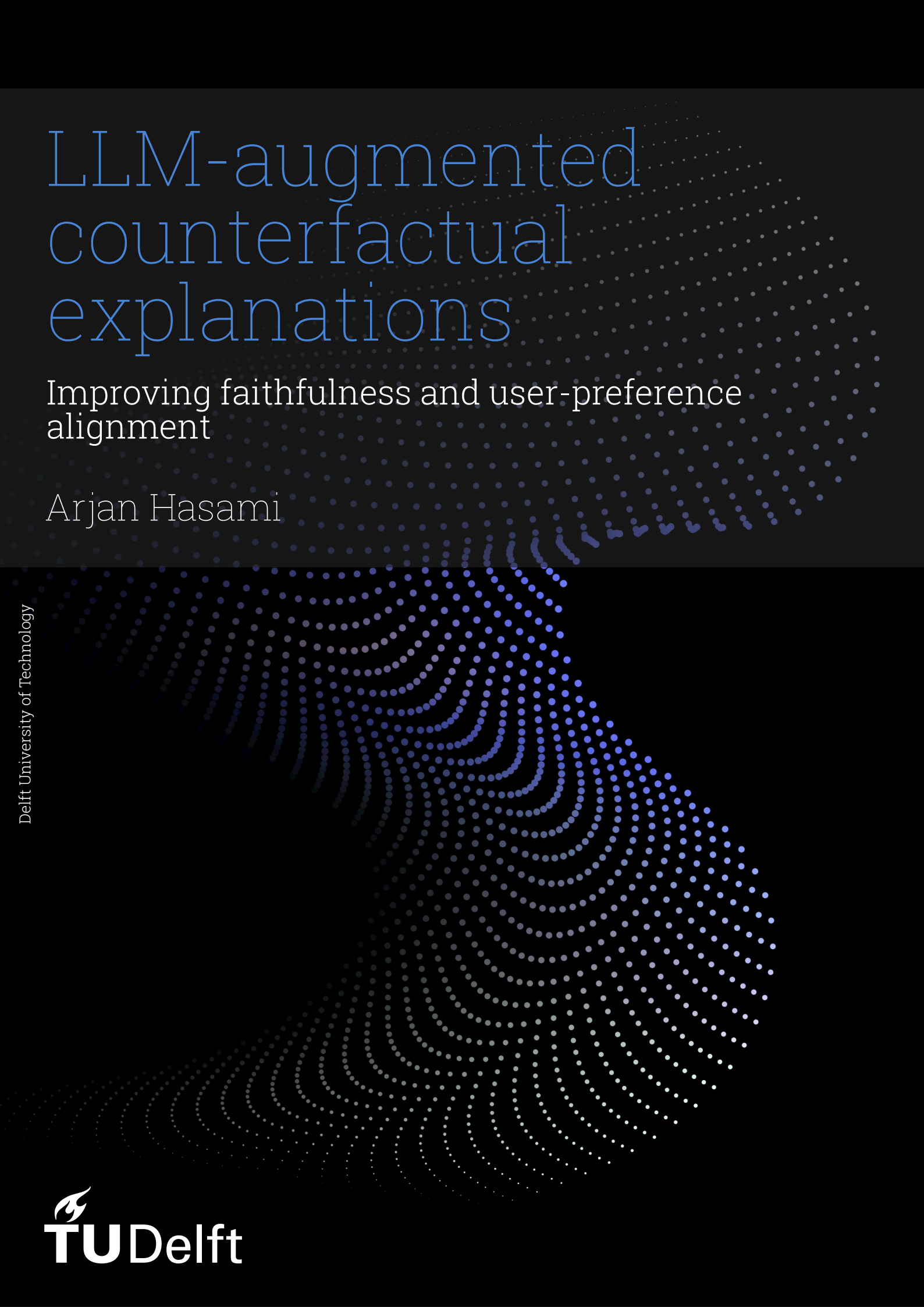
LLM-augmented counterfactual explanations

Improving faithfulness and user-preference alignment

Arjan Hasami

Delft University of Technology



TU Delft

LLM-augmented counterfactual explanations

Improving faithfulness and user-preference alignment

by

Arjan Hasami

5082099

Thesis Advisor:	Alan Hanjalic
Daily Supervisor:	Masoud Mansoury
Project Duration:	October, 2024 - November, 2025
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Part of the research and writing presented in this thesis has previously been published:

Arjan Hasami and Masoud Mansoury. 2025. Mitigating Popularity Bias in Counterfactual Explanations using Large Language Models. In Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3705328.3759330

TU Delft

Abstract

Counterfactual explanations (CFEs) offer a tangible and actionable way to explain recommendations by showing users a ”what-if” scenario that demonstrates how small changes in their history would alter the system’s output. However, existing CFE methods are susceptible to bias, generating explanations that might misalign with the user’s actual preferences. In this thesis, we study ACCENT, a neural CFE framework, and analyze its behavior through the lens of popularity bias. We introduce two alignment metrics, popularity distribution similarity (PDS) and expected popularity deviation (EPD), and evaluate 736 users with strongly niche- or blockbuster-oriented histories on MovieLens 1M and Amazon Video Games. Analysis shows that ACCENT’s explanations are systematically misaligned with historical user popularity preference. To address this, we propose a pre-processing step that leverages large language models to identify and filter out-of-character history items before generating explanations. Compared to simple heuristics and embedding-based filters, LLM-based filtering yields counterfactuals that are more closely aligned with each user’s popularity preferences, while preserving explanation conciseness and fidelity. A comparison between 4B and 8B parameter models further reveals that larger LLMs provide more stable, instruction-following behavior and stronger alignment, at the cost of increased computational overhead.

Preface

This thesis marks the end of my academic journey and the completion of my Master's degree in Computer Science at TU Delft.

I would like to sincerely thank my supervisor, Masoud Mansoury, for his guidance, patience, and constant support throughout the project. We developed this topic from scratch, and without his ideas, suggested literature, and detailed feedback, this work would not be where it is now. I am especially grateful for his help in turning this thesis into a publication, successfully submitting our paper to RecSys '25, and presenting it in Prague.

Finally, I want to thank the people around me who made these years enjoyable. From my family, to the friends I met at the University of Groningen and my housemates there, to everyone in Delft at the Delftsche Studenten Bond and Apollo, your company and support meant a lot. Two friends from my hometown, Simcha and Ruben, also deserve a special mention. Thank you all for your support, interest, and encouragement over the years.

Arjan Hasami
Delft, November 2025

Contents

1

Introduction

This chapter provides the foundation for the thesis. In § 1.1, the broader context in which explainable recommender systems and counterfactual explanations have emerged is first discussed, and then introduces how large language models can enhance these explanations by aligning them more closely with user expectations. § 1.2 then states the research questions used to approach the problem and goes over the main contributions of the thesis. Finally, § 1.3 goes over the structure of the rest of this document.

1.1. Context and motivation

Recommendation and Explainability Recommender systems (RS) have become a vital tool for guiding users to relevant content across various domains, including movies, music, and e-commerce [48, 11]. By learning from past user interactions, modern RS can personalize suggestions to each individual, greatly enhancing the user experience. As these systems become increasingly complex, leveraging deep learning and vast amounts of data, the decision-making process becomes more opaque. In the current age, where the content we see shapes our opinions and actions, this raises concerns about transparency and trust; users and stakeholders may question why certain items are recommended [12]. Explainability in recommender systems has thus emerged as a critical area of research. An explainable RS attempts to provide a clear and understandable reason behind its suggestions, which has been shown to increase user satisfaction and confidence in the system [69]. Over the years, various explanation approaches have been explored, but despite these advances, a key challenge remains: ensuring that the explanations truly resonate with the user's perspective and needs. It is not enough for an explanation to be logically correct; it must also be meaningful to the user. This observation sets the stage for more nuanced explanation strategies, such as counterfactual explanations, that can offer more profound insight into the recommendation process.

Counterfactuals Counterfactual explanations (CFE) [15, 62] do not just state why a prediction was made, but highlight how the prediction would change if the user behaviors or inputs were (minimally) different. In the recommendation setting, this typically means identifying a subset of the user's past actions that, if removed, would alter the top recommendation. A user might, for instance, be told the following:

> You were recommended 'John Wick' because you previously liked 'Taken' and 'The Equalizer'. Otherwise, you would have been recommended 'Mean Girls'.

This format provides a highly personalized and actionable explanation, allowing users to infer how their interests drive recommendations. The usefulness of this type of explanation is backed up by empirical research: participants in a user study on CFEs in recommendations [49] indicated they would like to see more of these detailed explanations instead of high-level transparency statements, which are often displayed in current applications. CFEs were especially

preferred by participants if they perceived the decision-making based on the recommendation as relatively consequential.

Rashomon and User-Alignment A well-known issue with CFEs is the Rashomon effect [38]: multiple distinct explanations can be valid for the same recommendation, each highlighting a different set of changes that might contradict each other. While this diversity of possible explanations can be seen as a strength, it also raises the question of which explanation should be shown to the user. Prior work tackles this in several, non-mutually exclusive ways, such as keeping only the nearest edit [16, 55, 59] or returning a diverse set of multiple counterfactuals [6, 39]. Instead, this work argues that the selected CFE should align with the user's expectations and interests. Highlighting an obscure or marginally relevant item may satisfy formal criteria, yet appear unintuitive or misleading to the user. Previous research also stresses that explanations should be tailored to the user and context, as different users value different criteria [8, 58].

The need for alignment becomes even more relevant when considering that explanation methods may reflect biases in the underlying recommender model. Because many CFE frameworks operate on internal signals such as gradients or influence measures, systemic biases can surface in the generated explanations. Figure 1.1 demonstrates how this might look with popularity bias [1], where users mainly interested in niche items may fail to resonate with explanations based on popular items.

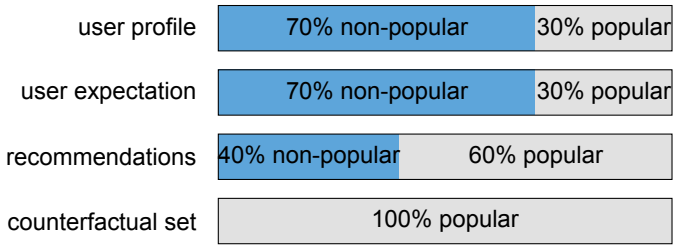


Figure 1.1: A user with both popular and non-popular items in their history may expect a similar balance in recommendations and counterfactual explanations. However, this alignment often breaks due to popularity bias. The effect can be even stronger in counterfactual sets, which are typically small and thus more sensitive to such skew.

Bridging the gap between model faithfulness and user-alignment is crucial for the next generation of explainable recommenders. Explanations should not only reflect the model's reasoning but also be framed in ways that resonate with the user, fostering trust and understanding. This thesis investigates using Large Language Models (LLMs) to enhance counterfactual explanation frameworks, leveraging their contextual understanding and nuanced reasoning to reduce misalignment. The goal is to steer existing methods toward explanations that remain rigorous in their counterfactual logic while better matching what users find intuitive and convincing.

1.2. Research Questions and Contributions

Research Questions To guide the approach that this thesis will take to improve the user-alignment of counterfactuals, the following research questions will be answered:

> **Research Question 1**: *To what extent are counterfactual explanations generated by ACCENT affected by misaligned items?*

Before improving user alignment, it is essential to understand the extent and nature of the problem in an established CFE framework. ACCENT [59] is a recent neural recommender explanation method that leverages influence functions to identify the most impactful training interactions for generating counterfactuals. While effective in producing faithful explanations, its reliance on internal influence signals may surface items that are not representative of a user's core preferences. For example, an influential but out-of-character item could dominate the explanation, making it appear unintuitive or irrelevant. Quantifying this misalignment provides the empirical basis for this research.

> **Research Question 2**: *How effectively can the proposed LLM-augmented framework improve the generation of counterfactual explanations?*

Large language models can capture semantic relationships that traditional numerical methods can overlook. Integrating them into the counterfactual explanation pipeline enables reasoning over item meaning and user intent, which could guide the selection of counterfactuals closer to the user's perceived preferences.

> **Research Question 3**: *How does the size of the LLM influence the quality of the counterfactual explanations?*

Model size is a key variable in LLM performance, with larger models generally offering better reasoning and contextual understanding. However, bigger is not always better for downstream tasks constrained by latency or cost. The final question examines how the parameter count affects CFE quality, comparing the performance of a small (4B) and a medium (8B) sized instruction-tuned model in the proposed framework.

Contributions This thesis makes the following contributions:

- **Analysis of existing approach**: Viewing a recent CFE method through the lens of popularity bias, current algorithms are shown to often generate counterfactual item sets that diverge from users' actual interaction histories.
- **Methodology**: A novel framework is introduced that integrates LLMs in the process of generating counterfactual explanations for the recommendation task. The method uses LLMs to create a user representation and evaluates each item's contribution in their history. Based on this, certain items are left out of the counterfactual generation process, leading to a counterfactual that aligns more with user preferences.
- **Implementation and Evaluation**: The proposed approach is implemented and evaluated on two public benchmarks, MovieLens 1M and Amazon VideoGames. Three metrics are further introduced: two to identify popularity misalignment, and one to evaluate explanation quality.
- **Analysis of LLM Size Impact**: The effectiveness of the framework is examined using two different LLMs, Gemma3 (4B parameters) and Qwen3 (8B parameters). By comparing the results, the influence of model size on the quality of explanations is shown.

Overall, this thesis advances the state of the art in explainable recommender systems by demonstrating how large language models can be leveraged to enhance the user alignment of counterfactual explanations. It ultimately aims to make personalized recommendations more transparent and user-centric.

1.3. Structure

The remainder of this thesis is organized as follows. Chapter 2 provides the necessary background, covering the fundamentals of recommender systems, explainable recommender systems, and large language models. Chapter 3 focuses on counterfactual explanations, detailing existing approaches in recommender systems, influence analysis, and the ACCENT framework, and formulating the problem studied in this thesis. Chapter 4 provides a detailed description of the proposed LLM-based augmentation step. Chapter 5 describes the datasets, experimental design, and implementation details, followed by the presentation of results in Chapter 6. Chapter 7 discusses the findings, and Chapter 8 concludes the thesis with a summary.

2 Background

In this chapter, the technical foundation for this thesis is established. § 2.1 starts by summarizing the fundamentals of modern recommender systems and the evolution towards neural approaches, followed by a brief review of popularity bias and common evaluation methods for RS. § 2.2 explores explainable recommender systems, examining types of explanations, the usage landscape, and the evaluation of explanations. Finally, § 2.3 provides a concise primer on language models.

2.1. Recommender Systems

Fundamentals Recommender systems are a fundamental area of research in Computer Science and play a key role in many modern applications, such as e-commerce (Amazon), streaming (Spotify, Netflix), and social media (TikTok, YouTube). Their primary objective is to predict user preferences based on historical data, enhancing user experience through personalized content delivery. Traditionally, recommender systems have relied on two main approaches:

- **Collaborative Filtering** (CF) [47] makes recommendations purely based on historical user-item interaction data, assuming that similar users have similar preferences (user-based CF) or that users prefer similar items (item-based CF). Early collaborative filtering methods utilized nearest-neighbor approaches [46].
- **Content-Based Filtering** [60] suggests items based on explicit item attributes and a user's past preferences. Features such as genres, keywords, or textual descriptions are used.

Popularized by the Netflix Prize [3], matrix factorization (MF) is a particularly influential method in collaborative filtering. It maps users and items into a shared latent factor space, modeling user-item interactions through the inner product of their corresponding vectors. This latent representation allows MF models to capture complex preference structures in a compact form, generalizing beyond observed interactions. The Netflix Prize competition demonstrated that MF models outperform classic nearest-neighbor techniques by enabling the incorporation of additional information, such as temporal effects and implicit feedback [30].

Neural Collaborative Filtering Despite their success and widespread adoption, MF-based methods are reliant on the inner product, which limits their ability to capture nonlinear relationships. To overcome these limitations, deep learning models have gained traction in recommendation tasks over the past decade. A notable development was the introduction of Neural Collaborative Filtering (NCF) [19], a general framework for learning user-item interactions through neural networks. Within this framework, the authors proposed several models, among which Neural Matrix Factorization (NeuMF) has become widely used. Figure 2.1 shows an overview of the model.

NeuMF combines a generalized matrix factorization (GMF) component, which models linear

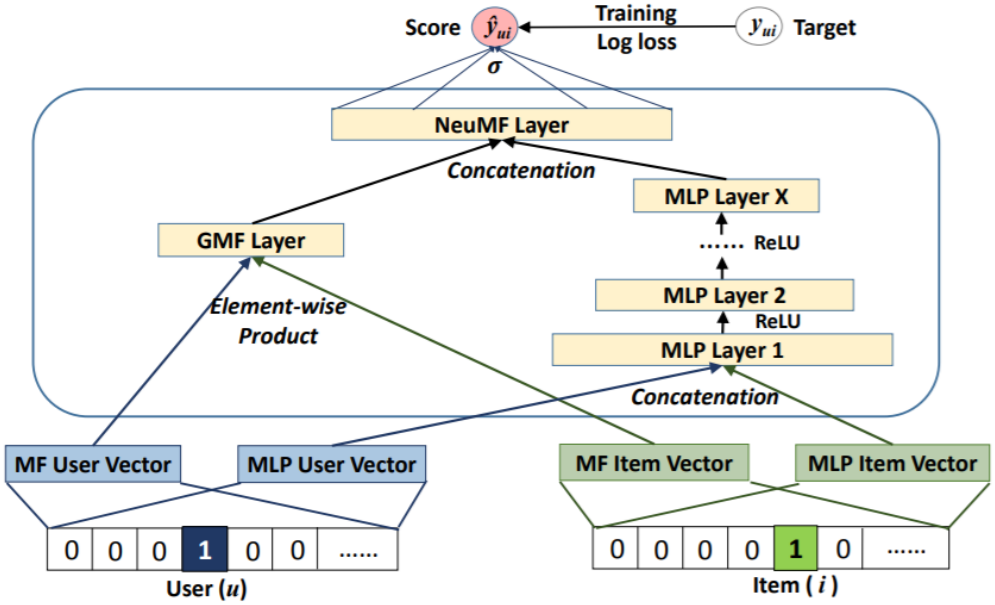


Figure 2.1: Excerpt from [19]. Architecture of the Neural Matrix Factorization (NeuMF) model. NeuMF unifies two complementary approaches: the Generalized Matrix Factorization (GMF) branch, which captures linear user-item interactions via element-wise multiplications of embeddings, and the Multi-Layer Perceptron (MLP) branch, which captures nonlinear relationships through successive nonlinear transformations of concatenated embeddings. By combining these two perspectives, NeuMF can model both simple and complex interaction patterns, leading to improved accuracy over traditional MF. The output of both branches is concatenated and fed into a final prediction layer, producing a score representing the likelihood of user-item engagement.

interactions via element-wise product, with a multi-layer perceptron (MLP) component that captures nonlinear patterns by processing concatenated user and item embeddings. The outputs of both components are fused and passed through a final prediction layer, allowing NeuMF to learn both low- and high-order interaction functions. This hybrid design enables NeuMF to surpass the limitations of MF's linearity while still reaping the benefits of its powers. As a result, NeuMF has become a foundational deep learning-based recommender model.

Bias Recommender systems are not neutral. They learn from historical data, which inevitably reflects the preferences and consumption patterns of the user base. As a result, these systems inherit and often amplify systematic biases, leading to recommendations that disproportionately favor certain items, users, or viewpoints.

Among many forms of bias identified in recommendation studies, popularity bias remains one of the most widespread and consequential. Popularity bias refers to the phenomenon where already popular items (with many past interactions in the training data) receive disproportionately more recommendations. This contributes to a "rich-get-richer" effect that limits diversity and reduces exposure for niche content [28, 1].

In real-world settings, biases can manifest in phenomena such as the filter bubble and echo chambers, where users are repeatedly exposed to similar viewpoints, content, or products, narrowing their informational landscape [25]. This can reinforce existing beliefs, reduce serendipity, and hinder the discovery of new interests. Such effects have motivated the field of fair and responsible recommender systems, which aims not only to improve accuracy but also to promote diversity, fairness, and transparency. By aligning recommendation strategies with broader social values, these systems strive to mitigate harm, increase inclusivity, and ensure that recommendations serve the interests of both individuals and society at large.

Addressing bias often requires multi-faceted approaches: re-weighting or re-sampling training data to better represent minority cases, incorporating fairness-aware objectives into model training, adding diversity-promoting constraints to ranking algorithms, or explicitly modeling bias factors to correct for them at inference time [5]. Increasingly, research also focuses on user-centric evaluations, ensuring that debiasing techniques not only improve statistical measures but also align with perceived fairness and trust from the user's perspective.

Evaluation Offline evaluation in RS often depends on the nature of the prediction task. Two widely used metrics are Root Mean Square Error (RMSE) for rating prediction tasks, and Normalized Discounted Cumulative Gain (nDCG) for ranking tasks.

For rating prediction, RMSE measures the average magnitude of the prediction error between the system's predicted ratings and the true ratings in the test set [30]. A lower RMSE indicates that the model is better at estimating users' explicit feedback. It is defined as:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{2.1}$$

Where $\hat{y}_i$ are predicted values, y_i are the ground truth values, and n is the number of test points. RMSE treats all errors equally, regardless of whether they occur for high or low ratings, making it a straightforward, albeit sometimes blunt tool for measuring recommendation quality.

For the top-K recommendation task, the focus shifts from predicting exact scores to ranking items so that the most relevant appear at the top of the list. Among ranking metrics, $nDCG@K$ is particularly valued because it accounts for both the relevance of recommended items and their positions in the top K rankings [23]. It first computes the Discounted Cumulative Gain (DCG):

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.2}$$

where rel_i is the graded relevance of the item at rank i. Higher relevance scores contribute more to the gain, but their impact is discounted logarithmically with rank, reflecting the diminishing attention users pay to lower-ranked items. To normalize this across users with different numbers of relevant items, the DCG is divided by the Ideal DCG (IDCG), which is the DCG obtained if all relevant items were ranked in the best possible order:

$$nDCG@K = \frac{DCG@K}{IDCG@K} \tag{2.3}$$

This gives us an nDCG score that ranges from 0 to 1, with 1 representing a perfect ranking.

In large-scale recommenders, computing nDCG over the entire item catalog can be computationally expensive. A common workaround is nDCG with negative sampling, where a small subset of items not interacted with by the user are randomly sampled to act as non-relevant items alongside the relevant test items [19]. The system then ranks this reduced candidate set instead of the full catalog. While this approach significantly reduces computation time, it is important to note that it produces an estimate of nDCG rather than the exact value.

2.2. Explainable Recommender Systems

With the move towards deep-learning based approaches, many state-of-the-art recommendation algorithms are difficult to explain. Although increased model complexity often correlates with higher accuracy, systems risk eroding trust and harming the user experience by not intuitively explaining why a user should like an item [20]. At the same time, regulations such as the GDPR [13] and the upcoming AI Act [14] strengthen users' right to explanation and algorithmic accountability. Explanations can also reduce users' cognitive workload and help them better

understand and trust the underlying algorithms. The remainder of this section outlines the major types of explanations, key applications, and evaluation strategies.

Types of Explanations Some machine learning models, such as linear regression or decision trees, are considered inherently interpretable due to their simplicity. These have also been applied to the recommendation task. For instance, [41] uses sparse linear models, learning an item-item weight matrix to generate recommendations directly. Due to the sparse nature, only a few feature weights will be non-zero, which can then easily be interpreted. Similarly, [50] constructs a decision tree for each user, with the decision rules based on a single attribute value. Each leaf represents a predicted rating, allowing the recommendation to be explained directly through the decision path. These transparent models provide clear and faithful explanations, as the scoring function of the model itself is used as an explanation. However, they typically underperform in recommendation accuracy compared to complex black-box models.

To explain black-box models, explanation methods are commonly grouped into three categories [36, 51].

Surrogate models: A black-box model can be approximated with a simpler, inherently explainable model trained on the black box's predictions [38]. The surrogate model is then used for explanations. While this approach is intuitive and straightforward, it should be noted that the global surrogate model only draws conclusions about the black-box model and not the actual data: the surrogate model never sees the true outcome.

Feature importance: Feature-attribution methods quantify how much each input feature (e.g., user demographics, item metadata, interaction history) contributes to a recommendation. *Pertubation-based approaches*, such as LIME [44] and its variation for recommender systems [42], modify these features to observe the impact on the model's output. *Gradient-based methods*, often used with neural recommenders, utilize gradients to evaluate the effect of small changes in input features on model predictions. Integrated Gradients is a widely used example, computing attributions by averaging gradients along a path from a baseline to the actual input [52].

Examples: Example-based explanations justify recommendations by referencing specific data points. *Adversarial examples* are small, imperceptible perturbations to an input that can cause a system to make incorrect predictions [54], typically used to improve model robustness during training [65]. *Influential instances* detect which inputs from the training data were the most influential for a prediction. The easiest way to do this is by retraining the model without specific inputs and assessing the impact. However, this is impractical for large amounts of data. In practice, mathematical techniques are used that do not require retraining [29]. Such methods can also be adapted for recommender systems [7]. These techniques can also underpin the third category: *Counterfactual examples*, which show how minimal changes to the input can alter the models output. Influence-based and alternative approaches to counterfactual generation are discussed in Chapter 3, with a focus on the former due to its relevance to the method used in this work.

Figure 2.2 provides an overview of the main categories of explainability techniques applied to recommender systems, as outlined above.

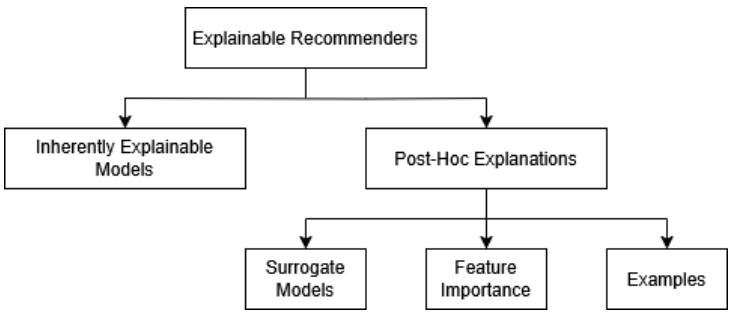


Figure 2.2: Landscape of explainable recommenders.

Usage Landscape Explainable recommenders are no longer just academic prototypes; they are increasingly deployed in real-world systems across multiple domains [69]. Their usage can be broadly grouped in three categories: high-stakes decision support, user engagement and trust, and model development.

High-stakes: The European AI Act, scheduled for enforcement starting in 2026, classifies systems that could pose a significant risk of harm to the health, safety, or rights of a person as high-risk and obliges providers to log clear, human-interpretable explanations for automated decisions [14]. Explainability is therefore not always optional anymore. In the context of recommender systems, some key examples might include suggesting treatment options in healthcare, recommending credit products, personal loans, or financial advice, finding relevant precedents or counter-arguments for case preparation, matching vacancies to job-seekers, or ranking applicants for recruiters.

User engagement and trust: The most widespread use of explainable recommendations today is in consumer-facing platforms, such as streaming services, e-commerce sites, and news apps. These systems often display simple justifications to increase perceived transparency and trust. Empirical studies show that even relatively shallow explanations can improve perceived transparency, satisfaction, and sometimes choice confidence [31]. Typical interface patterns include:

- Recommendations tied to recent activity, such as ”Because you watched...” or ”Frequently bought together...”
- Cold-start onboarding prompts, such as ”Did we guess right?” to quickly capture preferences
- Justifications for diverse or unexpected suggestions, aimed at increasing acceptance of serendipitous or long-tail items

Model development: For system designers and researchers, explanations provide insight into the model’s internal behavior, helping to diagnose failure cases, identify and mitigate biases, and refine algorithms without relying solely on aggregate metrics or expensive online A/B tests. In this setting, explanations are primarily a tool for debugging and analysis, rather than for end-user persuasion.

Evaluating Explanations It isn’t easy to measure the quality of an explanation due to how subjective this can be. A justification that one user finds useful might not be helpful at all for another, and vice versa. As such, there are few approaches for quantifying the effectiveness of recommender explanations.

Ideally, explanations would be evaluated offline, without requiring resource-intensive user studies. This is easier said than done. A comprehensive study on offline metrics for evaluating explanations in recommender systems [68] finds that there is no consensus on which metrics to use. In a review of 103 papers, the authors identify a broad range of metric families, including but not limited to precision/recall-style measures against some form of ground truth, path-based measures for knowledge-graph explanations, text-overlap metrics such as BLEU and ROUGE, correlation-based analyses, and counterfactual metrics that test whether elements in the explanation are causally relevant to the recommendation. In many cases, the choice of metric is tightly coupled to how the explanation method is constructed.

While these offline metrics can provide valuable insights and can scale well, many fail to capture the nuanced subjective value of an explanation to real users. BLEU and ROUGE, for example, are some of the most commonly used metrics for offline evaluation, but research suggests they only very weakly correlate with human perception of meaningfulness [35]. BLEU and ROUGE, together with anecdotal evidence–which also is not rigorous [40]–make up about 60% of the offline evaluations done in the papers reviewed in [68]. Offline evaluation should thus be interpreted with care and, where possible, complemented with user-based evaluation.

One approach that mixes offline and online evaluation is described in [64]. The authors conduct a user study measuring the effectiveness of machine-generated explanations and then train machine learning models on the collected scores and justifications to derive a numerical evaluation

metric. Although the resulting models do not achieve high accuracy against the human data, the generated scores correlate better with human judgments than traditional explainability metrics.

2.3. Language Models

Embedding models Embedding models map discrete objects such as words, users, or items into continuous vector spaces where semantic or behavioral similarity corresponds to geometric proximity. Early approaches like word2vec learn embeddings by predicting neighboring tokens in large corpora, yielding dense representations that capture syntactic and semantic regularities [37]. Building on this idea, more recent transformer-based models such as Sentence-BERT produce sentence- and document-level embeddings that can be used directly for similarity search, clustering, and downstream tasks [43]. In recommender systems, embedding models make it possible to represent users and items in a shared latent space, enabling efficient computation of relevance and flexible integration of textual metadata.

LLM basics Large language models are a class of artificial intelligence models that learn to generate and understand text by predicting the next token in a sequence. A token is a unit of text used by the model, often a word, part of a word, or even a single character. LLMs are trained on massive, diverse, textual datasets that contain trillions of tokens.

At their core, LLMs are giant neural networks, based on the transformer architecture [61], which introduced self-attention to efficiently capture long-range dependencies in text. During inference, the model takes the existing sequence of tokens and estimates the probability distribution over possible next tokens, selecting one according to its decoding strategy. Figure 2.3 illustrates this process.

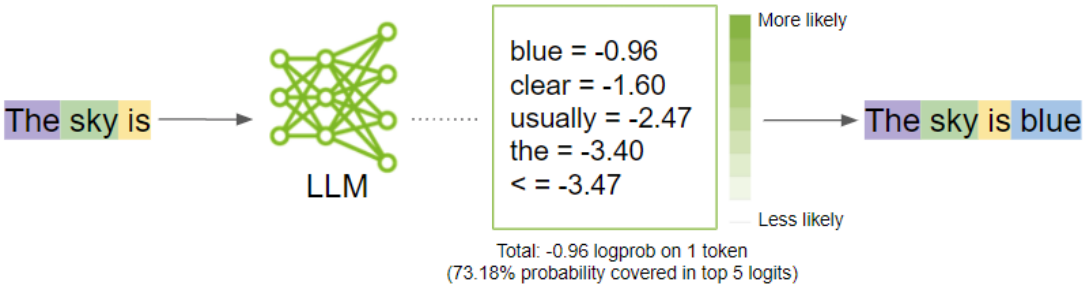


Figure 2.3: Excerpt from [53]. LLMs work on basis of next-token prediction: given the sequence ”The sky is”, the model assigns probabilities to possible continuations, with ”blue” being the most likely in this case.

The performance of LLMs follows scaling laws, which show that prediction accuracy improves smoothly as model size, training data, and computing resources increase, driving a rapid growth in model sizes [27]. After the initial pre-training, LLMs undergo instruction tuning. By fine-tuning on prompt-response examples, the model is adapted to follow natural language instructions. Then, as a final, optional step, LLMs can be adapted for the task at hand through supervised fine-tuning and reinforcement learning from human feedback, refining the model based on human-rated prompts and preferences.

LLMs in Recommender Systems LLMs are increasingly being explored in recommender systems, both as standalone recommenders and as components that enhance existing methods. As recommenders, they can use natural language prompts, product descriptions, and user reviews to generate personalized suggestions without relying on traditional user-item interaction data. For example, work on generative recommendation treats recommendation as a text-generation problem, prompting an LLM to output item identifiers or titles directly as a ranked list of suggestions [24].

They have also shown potential in improving explainability. LLMs can synthesize information from user histories, item metadata, and reviews into fluent, human-readable justifications. XRec, for instance, combines graph-based collaborative filtering with an LLM to generate explanations

that are conditioned on both interaction signals and textual context, leading to more informative and personalized explanations than template-based baselines [34].

Recently, research has also been done on using LLMs to mitigate bias in recommendations. Lichtenberg et al. probe whether LLM-based recommenders contribute to or can alleviate popularity bias [33]. Using a new metric, traditional RS is compared to LLM-based models. LLM-based recommenders are found to usually have less popularity bias than traditional approaches. By including additional instructions, they find that it is possible to lower popularity bias even further.

Limitations Despite their capabilities, LLMs have several important limitations that affect how they should be used in practice. They can produce hallucinations: fluent, confident outputs that are factually incorrect, logically inconsistent, or entirely fabricated [70]. Because these models operate by learning statistical patterns from large text corpora, they do not possess genuine understanding of their inputs or outputs, and are unable to reliably distinguish between true and false statements [2]. As a result, they may also reproduce and amplify biases present in their training data, and their behavior can vary greatly depending on prompt used. Finally, training and deploying LLMs requires substantial computational resources, which constrains who can build and operate them and raises concerns about cost and environmental impact [2].

3

Counterfactual Explanations

In this chapter, counterfactual explanations are examined in greater detail. Wachter et al. [62] argue for CFEs as a means to provide intuitive ”what-if” scenarios to help users understand why a particular decision is made. For example:

> You were denied a loan because your annual income was $30.000. If your income had been $45.000, you would have been offered a loan.

The black-box model's decision is followed by a counterfactual: a statement of how the variables would have to be different for the model to give a different desirable output. While a lot of the earlier work focused on CFEs in the classification task, they can also be applied to recommender systems. Figure 3.1 shows an example.

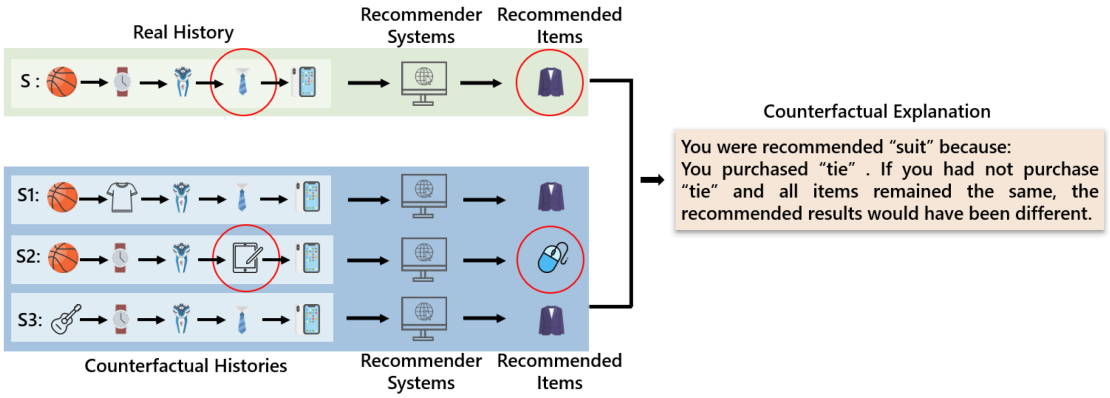


Figure 3.1: Excerpt from [18]. CFEs in recommendation attempt to find a change in user history that would result in a new item being recommended. The items perturbed in the user history form the counterfactual set, and the item recommended for this updated history is referred to as the replacement item. In this instance, if the user would have purchased a tablet instead of a tie, the user would have been recommended a mouse instead of a suit.

Recent studies show that users appreciate this style of explanation, especially when the recommendations influence meaningful decisions. For example, Shang et al. [49] found that users preferred counterfactual explanations over generic transparency statements, particularly when the stakes were perceived as high.

In the remainder of this chapter, several approaches for generating CFEs in recommender systems are discussed in § 3.1, followed by a closer examination of influence functions in § 3.2 and the ACCENT framework used as the baseline in § 3.3. Finally, § 3.4 explores how bias may manifest in ACCENT and outlines why LLMs are a promising candidate to mitigate it.

3.1. Approaches for CFEs in RS

Various approaches have been proposed to generate CFEs in recommender systems. While these methods differ significantly in focus and technical detail, they can generally be divided into two categories: optimization-based methods, searching for changes to user representations or item features, and score-based methods, assessing the importance of individual user interactions. Although some methods blur the lines between the two, this distinction captures the dominant methodological patterns in the field.

Optimization-based methods Optimization-based methods frame the generation of CFE as a constrained search problem. Rather than removing specific interactions, these methods perturb features, embeddings, or user profiles to shift the model's output. The goal is to find minimal, plausible changes that lead to a different recommendation.

CountER [55] frames counterfactual generation as a causal-inference problem over item attributes such as price or genre. It sets up a joint optimization problem for a given recommendation, balancing three terms: effectiveness (the rank of the original recommendation must drop), sparsity (fewer items are preferred), and plausibility (changes should stay in realistic bounds).

LiCE4Recommenders [4] treats counterfactual generation as a mixed-integer optimization over the full binary interaction vector. To avoid unrealistic histories, it augments the objective with a likelihood term learned by a sum-product network that scores the plausibility of a given user history. The solver searches for a minimal set of flips (also allowing adding items instead of only removals) that satisfies the plausibility threshold and pushes the original item out of the recommendation list.

CAVIAR [22] is a counterfactual framework that targets multi-modal recommenders that rely on item images. CAVIAR searches for the smallest change to an item's visual embedding that demotes it below the top-k. It solves a two-term optimization problem, optimizing for effectiveness (the edited embedding must not be recommended) and simplicity (the perturbation should be spatially sparse). The perturbed embedding is mapped back to a human-interpretable explanation using a CLIP model, yielding natural language counterfactuals.

Score-based methods Score-based methods aim to identify a small subset of user interactions whose removal would change the system's recommendation. These methods assign importance scores to items in the user history, based on graph structure, gradients, or other internal signals. These scores are then used to select items for removal and find a replacement.

PRINCE [16] is one of the earliest frameworks applying counterfactual reasoning to graph-based recommenders. It models each user's interaction history in a dynamic interaction graph and searches for the smallest set of edges (i.e., past interactions) whose removal would dethrone the current top recommendation.

Model-Agnostic Counterfactual Search (MACS) [26] is a simple, heuristic approach that requires no access to model internals. Starting from the user's full history, it performs a greedy search: at each step, it temporarily removes every remaining interaction, queries the recommender, and measures how much the rank of the target item changes. The interaction that produces the greatest rank drop is permanently deleted, and the process repeats until the original recommendation is no longer top-ranked or a budget is reached. Because the importance score is computed purely from observed rank changes, MACS can be applied to any recommender that allows modified user histories at inference.

Fast Influence Analysis (FIA) [7] uses influence functions [29] to estimate how much each training interaction influences a prediction in latent factor models. Approximating these influence scores enables counterfactual explanations that cite a handful of past interactions without retraining the model. Further discussion of influence analysis and FIA is provided in § 3.2.

ACCENT [59] builds on FIA and adapts it for neural recommenders. It extends influence analysis to pairs of items–the top recommendation and a plausible alternative–and iteratively searches

for the smallest set of user history items whose removal flips the ranking. Further discussion of ACCENT is provided in § 3.3.

3.2. Influence Analysis

Koh et al. [29] introduced the idea of explaining a model by tracing a prediction back to the training data and asking the counterfactual: what would happen to the prediction without this training point, or if it were slightly different? As this is infeasible to achieve by perturbing the training data and retraining due to computational cost, influence functions—a technique from robust statistics—are used. Figure 3.2 shows an example of influence functions detecting the most helpful training points for two fish vs. dog classification models.

Figure 3.2: Excerpt from [29]. Influence functions applied to identify the most helpful training points for a clownfish test image (center right). Left: Influence score versus Euclidean distance for two models. In an RBF SVM (top), a classifier looking at pixel-level similarities, the most helpful examples (green) are close to the test image in pixel space. In an Inception deep network (bottom), high-influence points can be far in pixel space but close in learned feature space. Middle: the two most helpful training images for each model. While both models surface fish images, the images most helpful to the Inception model for this prediction are actually clownfish. An image of a dog is also found to be helpful.

Mathematical Foundation With n training points, each training point has a weight of $\frac{1}{n}$. The parameter change can be linearly approximated if a training point z were removed by upweighting it by $-\frac{1}{n}$:

$$\hat{\theta^{-z}} - \hat{\theta} = \frac{1}{n} H_{\hat{\theta}}^{-1} \nabla L(z, \hat{\theta}) \tag{3.1}$$

where $\hat{\theta}$ are the optimized model parameters, $\hat{\theta^{-z}}$ are the model parameters after removing training point z, $H_{\hat{\theta}}$ is the Hessian matrix computed as $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\hat{\theta}}^2 L(z_i, \hat{\theta})$ and $L(z, \theta)$ is the loss of a training point z on parameters θ.

The chain rule can then be applied to measure how upweighting z influences the loss at a test point z_{test}, which gives:

$$Influence(z, z_{test}) = -\nabla_\theta L(z_{test}, \hat{\theta})^\intercal H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{3.2}$$

Note that influence functions are based on a convex and differentiable model. For non-convex models, an approximation can still be found by enforcing the invertibility of $H_{\hat{\theta}}$ by adding a small dampening factor λ to its diagonals. If the derivatives of the loss ($\nabla_\theta L$ and ∇_θ^2) do not exist, a smooth approximation can be used.

Influence in RS Fast Influence Analysis (FIA) [7] extends influence functions for generating explanations for recommender systems. Given a recommended item, FIA aims to generate

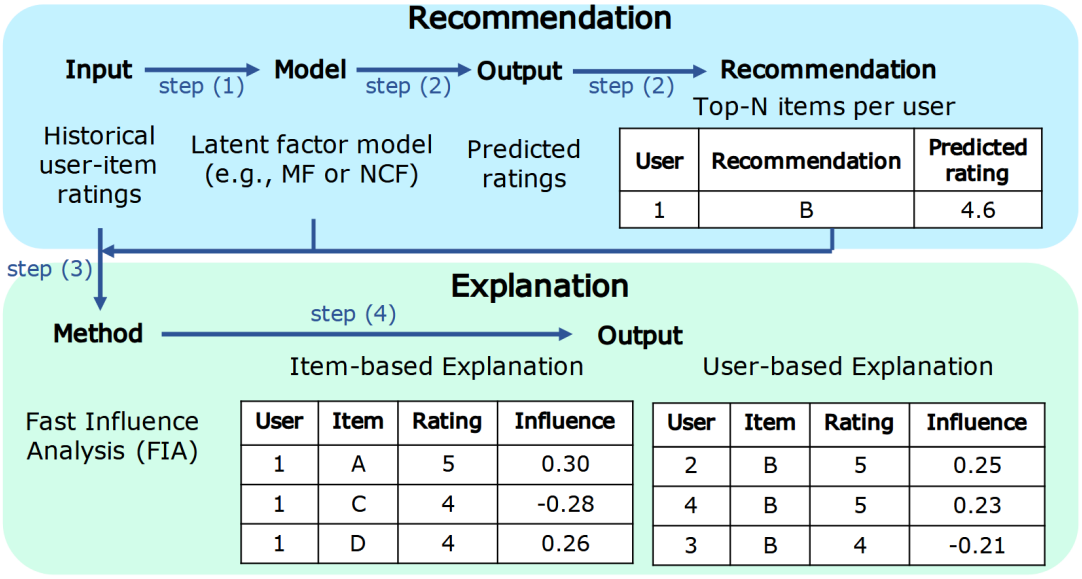


Figure 3.3: Excerpt from [7]. Overview of the Fast Influence Analysis (FIA) framework. In the recommendation stage (blue), historical user-item ratings are used to train a latent factor model, producing predicted ratings and top-N recommendations. In the explanation stage (green), FIA estimates the influence of historical ratings on a specific recommendation without retraining. FIA is able to find both item-based explanations (most influential items rated by the target user) or user-based explanations (most influential ratings from other users for the target item), providing intuitive neighbor-style justifications.

neighbor-style explanations, containing the most influential items that a user has rated. An overview of the FIA framework can be found in Figure 3.3.

The considerable size of the parameters of recommendation models leads to a high computational cost for regular influence functions. To accelerate the process, FIA takes advantage of the observation that for a given test point (u_t, i_t), the predicted rating $\hat{y}(u_t, i_t)$ by matrix factorization is only denoted by a small fraction of the model parameters. This fraction of the model parameters consists of the latent factors of u_t and i_t, denoted by $\theta_t = \{p_{u_t}, q_{i_t}\}$. R_t is defined as the set of training points in the historical ratings of u_t and i_t. The formulation for influence in matrix factorization is as follows:

$$Influence(z, u_t, i_t) = -\frac{1}{n'} \nabla_{\theta_t} L(z, \hat{\theta})^\intercal H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta}) \tag{3.3}$$

where $z \in R_t$, $n' = |R_t|$ and $H_{\hat{\theta}_t} = \frac{1}{n'} \sum_{i=1}^{n'} \nabla_{\hat{\theta}}^2 L(z_i, \hat{\theta})$, as the parameters θ_t are only optimized by training points in R_t.

[7] also extends FIA to the Neural Collaborative Filtering setting. The parameters involved in NCF are divided into two parts, θ_e and θ_n, where θ_e is the embedding of the users and items, and θ_n is the parameters of the interaction function. These two parts of the parameters are affected differently by the training points: θ_e is optimized by the training points in R_t, similarly to θ_t in matrix factorization. θ_n, however, is learned using the complete set of training points. The following equations thus formulate influence in the NCF setting:

$$Influence(z, u_t, i_t) \approx Influence(z, u_t, i_t | \Delta\theta_e) + Influence(z, u_t, i_t | \Delta\theta_n) \tag{3.4}$$

Simplifications described in [7] allow for approximation

$$Influence(z, u_t, i_t | \Delta\theta_e) = -\frac{1}{n'} \nabla_{\theta_e} L(z, \hat{\theta})^\intercal H_{\hat{\theta}_e}^{-1} \nabla_{\theta_e} \hat{y}(u_t, i_t, \hat{\theta}) \tag{3.5}$$

Where the Hessian is over n'

$$Influence(z, u_t, i_t|\Delta\theta_n) = -\frac{1}{n}\nabla_{\theta_n} L(z, \hat{\theta})^\intercal H_{\hat{\theta_n}}^{-1} \nabla_{\theta_n} \hat{y}(u_t, i_t, \hat{\theta}) \tag{3.6}$$

Where the Hessian is over n.

3.3. ACCENT

One approach that utilizes the aforementioned influence functions to produce counterfactual explanations for NCF models is ACCENT [59]. Given a user u that has an interaction history I_u and a recommendation rec, ACCENT attempts to find a counterfactual set whose removal from the training data results in a different recommendation rec^*. To achieve this, ACCENT extends the use of influence scores to pairs of items, rec and rec^*.

The rating of user u for item i before and after the removal of training point z is denoted as $\hat{y}_{u,i}$ and $\hat{y}_{u,i}^{-z}$ respectively. The influence of z on $\hat{y}_{u,i}$ is defined as:

$$Influence(z, \hat{y}_{u,i}) = \hat{y}_{u,i} - \hat{y}_{u,i}^{-z} \tag{3.7}$$

The influence of z on the *score gap* between two items i and j can be estimated:

$$\begin{aligned} Influence(z, \hat{y}_{u,i} - \hat{y}_{u,j}) &= (\hat{y}_{u,i} - \hat{y}_{u,j}) - (\hat{y}_{u,i}^{-z} - \hat{y}_{u,j}^{-z}) \\ &= (\hat{y}_{u,i} - \hat{y}_{u,i}^{-z}) - (\hat{y}_{u,j} - \hat{y}_{u,j}^{-z}) \\ &= Influence(z, \hat{y}_{u,i}) - Influence(z, \hat{y}_{u,j}) \end{aligned} \tag{3.8}$$

$$Influence(z, \hat{y}_{u,i} - \hat{y}_{u,j}) = Influence(z, \hat{y}_{u,i}) - Influence(z, \hat{y}_{u,j})$$

To determine the influence of removing a set of training points on the score gap, the effects of removing each individual point can be summed. Note that the accuracy of this estimation deteriorates for larger sets.

To replace rec with rec^*, the counterfactual set $Z \subseteq I_u$ must be identified whose removal results in:

$$\begin{aligned} &\hat{y}_{u,rec}^{-Z} - \hat{y}_{u,rec^*}^{-Z} < 0 \\ &\Leftrightarrow \hat{y}_{u,rec} - \hat{y}_{u,rec^*} - \hat{y}_{u,rec}^{-Z} + \hat{y}_{u,rec^*}^{-Z} > \hat{y}_{u,rec} - \hat{y}_{u,rec^*} \\ &\Leftrightarrow Influence(Z, \hat{y}_{u,rec} - \hat{y}_{u,rec^*}) > \hat{y}_{u,rec} - \hat{y}_{u,rec^*} \\ &\Leftrightarrow \sum_{k=1}^{m} Influence(z_k, \hat{y}_{u,rec} - \hat{y}_{u,rec^*}) > \hat{y}_{u,rec} - \hat{y}_{u,rec^*} \end{aligned} \tag{3.9}$$

The optimal way to replace rec with rec^* is thus to add training points z_k to Z in order of decreasing $Influence(z_k, \hat{y}_{u,rec} - \hat{y}_{u,rec^*})$ until the equation is satisfied.

To select rec^*, the item to be replaced with the initial recommendation, any item can be chosen. However, checking all items is computationally expensive, and a practical choice is to use the original set of top-k recommendations, as it ensures that rec^* remains relevant to u.

3.4. Problem Formulation

Limitations of influence-based CFEs. In recommender data, training signals are highly uneven: interactions involving popular items appear far more frequently and can exert disproportionate influence on the learned model compared to those involving long-tail items [1]. ACCENT estimates how much each of these signals would shift a target recommendation by computing their influence function. Figure 3.4 illustrates the idea in the simplest setting of linear regression: a handful of high-leverage outliers bend the fitted line away from the cloud of inlier points, whereas deleting them produces a model that follows the main trend. Influence plots highlight exactly these points because their removal would cause the greatest change in the estimator.

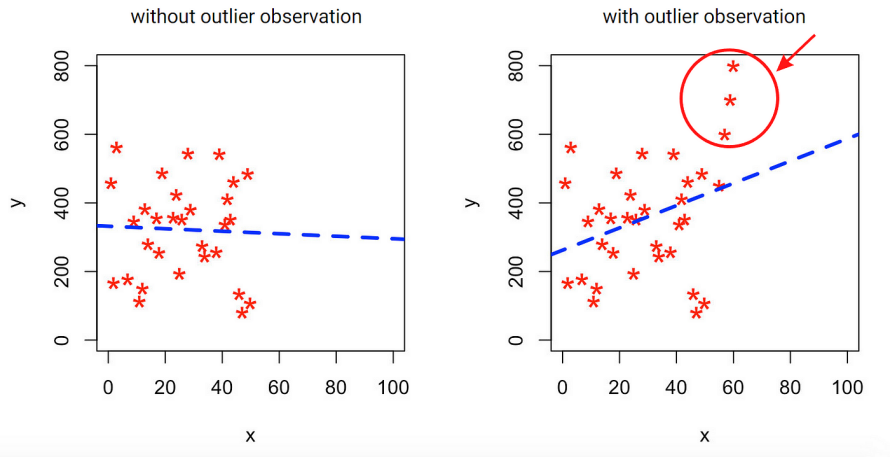


Figure 3.4: Only a handful of outliers can distort a linear regression model. These are the most influential points: they have the largest impact on the model predictions.

A similar effect can arise in recommender systems. Items that exert disproportionate leverage on the predicted score are flagged as ”most influential”, even when they are not necessarily central to the user’s preferences. When a systematic mismatch exists between the training signals that dominate influence and the subset that actually reflects a user’s taste, ACCENT’s counterfactual set can surface poorly aligned items, such as mainstream blockbuster items for a niche-oriented user or obscure long-tail items for a mainstream-oriented one.

A related phenomenon is observed in the work that introduced influence functions for machine learning [29]. There, the points with the highest influence often correspond to mislabeled or problematic training examples, and inspecting these emails allowed humans to detect and correct most flipped labels in a spam-classification dataset. This illustrates that highly influential points need not be ”typical” examples, but rather those that most strongly affect the model.

Popularity bias in ACCENT In recommender data, popular items converge the fastest [28]. As there are many more interactions, there are more training iterations and updates, giving them disproportionate weight during learning. ACCENT is based on influence functions, which tell us which training points are most used for a recommendation. However, due to the disproportionate weight of popular items due to popularity bias, these are more influential in most predictions. These popular items thus usually have a high influence score, making it likely that ACCENT picks them.

Towards bias-aware CFEs with LLMs Keeping in mind this property of influence functions, creating CFEs using them might lead to low user-alignment. However, it is possible to attempt to correct for it. LLMs can encode high-level preferences, such as tone, themes, or narrative style, by reading free- text descriptions of consumed items. Recent work shows that LLM-based recommenders exhibit lower popularity bias than traditional baselines, and that careful prompting can reduce bias further [33]. This thesis therefore investigates whether LLMs can act as a preprocessing step, filtering a user’s history so that existing CFE frameworks such as ACCENT

can start from a less biased candidate pool, leading to an explanation that aligns more with user interest.

4

LLM-augmented ACCENT

The central hypothesis of this work is that certain items in a user's interaction history may disproportionately influence the generation of counterfactual explanations, leading to explanations that are technically correct but misaligned with the user's preferences. This chapter proposes a novel framework that identifies and excludes such items from the counterfactual generation process to address this, aiming to produce counterfactual sets that are more representative of a user's perceived interests.

Two approaches are proposed for finding these misaligned items, both of which are based on greedy heuristics. The first approach, described in § 4.1, employs a transformer-based embedding model, whereas the other approach uses large language models, which is described in § 4.2. Figure 4.1 presents a global overview of our framework.

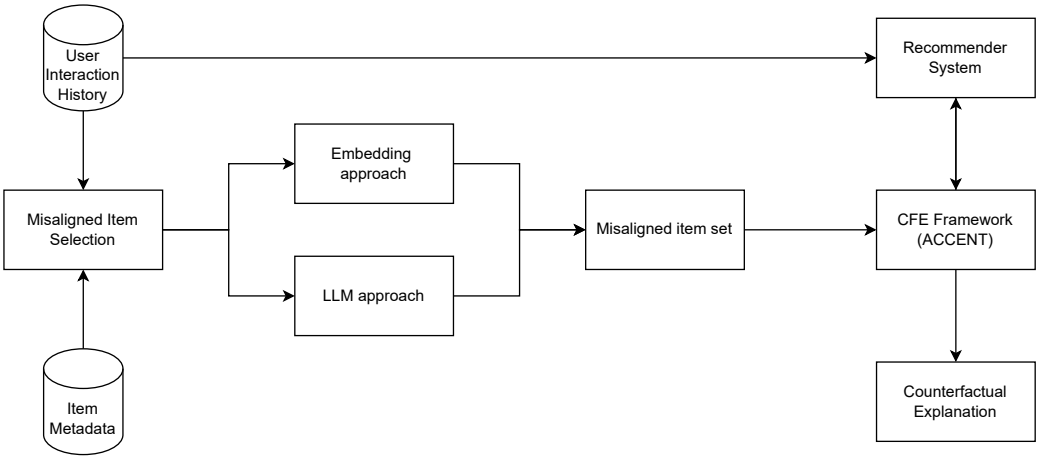


Figure 4.1: Global overview of the LLM-augmented counterfactual explanations framework.

4.1. Embedding-based identification

Our initial method leverages an embedding model to quantify the influence of individual items in a user's history on their overall preference profile. These high-dimensional embeddings aim to capture semantic similarities based on textual item metadata, such as item titles, descriptions, or categories, and are generated using an attention-based transformer model.

A high-dimensional vector embedding for a singular item i using Sentence Transformer (SBERT) [43] can be created as follows:

$$e_i = \text{SBERT}(p_i) \tag{4.1}$$

Where p_i is the present metadata of item i.

User Embeddings by Aggregating Item Embeddings Having defined the embedding of a singular item, this can now be extended to the user level. Let H_u denote the interaction history of user u, and let e_i be the embedding of item $i \in H_u$. The embedding of a user is then given by the sum of the embeddings of the items in the user's history:

$$e_u = \sum_{i \in H_u} e_i \tag{4.2}$$

Key to our approach is comparing the similarity of alternate user histories. A modified user embedding after removing an item or a set of items $S \subseteq H_u$ from the user profile is defined as:

$$e_{u'} = e_u - \sum_{i \in S} e_i \tag{4.3}$$

Finally, the impact of the removal of S on the user embedding with the cosine dissimilarity between e_u and $e_{u'}$ is quantified:

$$\text{removal\_impact}(S) = 1 - \frac{e_u \cdot e_{u'}}{\|e_u\| \|e_{u'}\|} \tag{4.4}$$

A high influence score indicates that removing S significantly alters the user's profile: the removed items strongly affected the user's latent representation, suggesting that they may overly dominate the recommendation and counterfactual generation logic.

Picking the Items The goal is to find the subset that, when removed, maximally alters the user embedding, indicating that its items might misalign with a user's self-perceived preference. The size n of this subset is left as a hyperparameter. One possibility is to generate and check all possible unique subsets of the user history of size n and smaller, but this quickly blows up the search space due to the factorial nature of subsets. This makes it infeasible when working with longer histories, which is the standard in many domains such as multimedia recommendation. To combat this, a greedy algorithm is adopted that iteratively selects the most impactful removal at each step.

At step zero, the process starts with a user's full interaction history $H_u^{(0)} = H_u$. Then, for a fixed number of steps n, the following is iterated:

- For every remaining item $i' \in H_u^{(j-1)}$ (history before step j), compute the embedding obtained after its removal $e_{u \setminus \{i'\}}^{(j-1)}$.
- Select the item whose removal produces the largest cosine dissimilarity with the current profile:

$$i_j^* = \underset{i' \in H_u^{(j-1)}}{\arg\max} (1 - \cos(e_u^{(j-1)}, e_{u \setminus \{i'\}}^{(j-1)})) \tag{4.5}$$

- Permanently discard i_j^*: $H_u^{(j)} = H_u^{(j-1)} \setminus \{i_j^*\}$.

The pseudo-code for this algorithm is presented in Algorithm 1.

4.2. LLM-based identification

Our second approach attempts to harness the richer and more flexible reasoning of LLMs. Instead of pre-computing item embeddings, LLMs are employed.

Algorithm 1: Greedy identification of misaligned items, embedding model approach.

Input: H_u: set of item IDs representing the user's interaction history
I_e: mapping from item IDs to their embedding vectors
n: size of the misaligned set to create
Output: A subset of n items from H_u that, when removed, maximally reduce dissimilarity

Compute $e_u^{(0)}$ using Eq. 4.2;
Set $H_u^{(0)} \leftarrow H_u$, $S \leftarrow \emptyset$;
for $j = 1$ **to** n **do**
  foreach $i' \in H_u^{(j-1)}$ **do**
    Compute $e_{u\setminus\{i'\}}^{j-1}$ using Eq. 4.3;
    Compute dissimilarity $d = 1 - cos(e_u^{(j-1)}, e_{u\setminus\{i'\}}^{(j-1)})$;
  end
  Select $i_j^* = \arg\max d$;
  Update $H_u^{(j)} \leftarrow H_u^{(j-1)} \setminus \{i_j^*\}$;
  Update $S \leftarrow S \cup \{i_j^*\}$;
  Update $e_u^{(j)} \leftarrow e_{u\setminus\{i_j^*\}}^{j-1}$;
end
return S;

User Embeddings with LLM-Generated User Profiles The expressive capacity of LLMs is harnessed by prompting them to generate free-form natural language descriptions of a user's preferences, based on their interaction history. This step reframes a list of consumed items into a coherent and human-readable user profile. This builds on the work of Sabouri et al., who use LLMs to uncover temporal dynamics in user preferences [45]. While their focus is on highlighting the temporal preferences of users, they also highlight the general utility of using LLMs for textual user profiles.

The prompting template asks the LLM to describe the user in fewer than 300 words, avoiding overly broad genre classifications or direct item references. Instead, the model is encouraged to extract recurring patterns in tone, themes, character types, or narrative structures. A list of all items in the user's interaction history is provided, including features when possible. This allows us to see not only what the user consumed, but why they might have liked it. Once the textual profile is generated, it is encoded into a high-dimensional embedding using a pre-trained embedding model. An example prompt for the movie domain is provided in Figure 4.2.

The mathematical definition of generating a textual profile t_u of the user's preferences using an LLM prompted on their full interaction history and item descriptions is as follows:

$$t_u = \text{LLM}(\underset{i \in H_u}{\text{concat}}(p_i)) \tag{4.6}$$

Then, embed t_u into a high-dimensional vector e_u using Sentence Transformer (SBERT) [43]:

$$e_u = \text{SBERT}(t_u) \tag{4.7}$$

Where p_i is the description of item i. Textual profiles and embeddings without items S are again defined as follows:

$$t_{u\setminus S} = \text{LLM}\left(\underset{i \in H_u \setminus S}{\text{concat}}(p_i)\right) \qquad e_{u\setminus S} = \text{SBERT}(t_{u\setminus S}) \tag{4.8}$$

Figure 4.2: Example LLM Query and Response (Movie Domain)

Query: *Your task is to analyze a list of movies a user has interacted with and describe the profile and the preferences of the user in less than 300 words. Try to not be too broad (e.g. mention too many general categories such as action or comedy). Do not mention specific movie titles.*

The user has watched the following movies:

Die Hard (1988) - Action, Thriller - NYPD cop John McClane's plan to reconcile with his...
Raiders of the Lost Ark (1981) - Action, Adventure - When Dr. Indiana Jones–the tweed...
...

Response: *The user enjoys action-packed, high-stakes narratives with strong protagonists overcoming formidable odds. They prefer films that blend suspense, tension, and personal growth. There's a clear affinity for classic cinema, particularly Westerns and gritty, character-driven stories. The user appreciates intelligent storytelling, moral complexity, and visually striking sequences. They also enjoy sci-fi and fantasy elements, especially those with a grounded or realistic approach. Themes of redemption, survival, and resilience are prominent. The user is drawn to films that challenge the mind and deliver emotional depth, often favoring well-crafted plots over spectacle. They appreciate both solo heroics and ensemble dynamics, and have a taste for both serious and slightly humorous tones.*

Picking the Items Rather than iterating over raw item embeddings, a textual profile is repeatedly created from the current history via Eq. 4.6, embed that profile with SBERT (Eq. 4.7), and score the effect of removing each candidate item by recomputing the profile without that item and measuring cosine dissimilarity to the current profile. In each iteration, the item that maximally perturbs the profile is removed, continuing until n items have been discarded. This mirrors the strategy discussed in § 4.1 but swaps the representation: user state is tracked as an LLM-written profile $t_u^{(j)}$ and its embedding $e_j^{(u)} = SBERT(t_u^{(j)})$. The pseudo-code for this approach is shown in Algorithm 2.

4.3. Updating the Counterfactuals

After n iterations, the set of all discarded items is $\tau_u = \{i_1^*, \ldots, i_n^*\}$. Finally, the filtered history $H_u^{(n)} = H_u \setminus \tau_u$ is created and fed it to ACCENT (or any other history-based CFE framework), ensuring explanations exclude these misaligned items.

Recall that ACCENT creates counterfactual sets by iteratively adding training points from the user history to the explanation, prioritizing those with the most significant influence score gap relative to a potential replacement item. To exclude misaligned items, the process is modified and these items are filtered out of the user history in a pre-processing step, by simply passing the filtered history $H_u^{(n)}$ into ACCENT.

Algorithm 2: Greedy identification of misaligned items, large language model approach.

Input: H_u: set of item IDs representing the user's interaction history
n: size of the misaligned set to create
Output: A subset of n items from H_u that, when removed, maximally reduce dissimilarity

Generate initial profile $t_u^{(0)}$ using Eq. 4.6;
Embed into $e_u^{(0)}$ using Eq. 4.7;
Set $H_u^{(0)} \leftarrow H_u$, $S \leftarrow \emptyset$;
for $j = 1$ **to** n **do**
    foreach $i' \in H_u^{(j-1)}$ **do**
        Form reduced profile t' and embed e' using Eq. 4.8;
        Compute dissimilarity $d = 1 - cos(e_u^{(j-1)}, e')$;
    end
    Select $i_j^* = \arg\max d$;
    Update $H_u^{(j)} \leftarrow H_u^{(j-1)} \setminus \{i_j^*\}$;
    Update $S \leftarrow S \cup \{i_j^*\}$;
    Update $e_u^{(j)} \leftarrow e_{u \setminus \{i_j^*\}}^{j-1}$;
end
return S;

5

Methodology

This chapter outlines the datasets, experimental design, and implementation details used to evaluate the proposed framework. Users are first partitioned into groups with predominantly mainstream (blockbuster) or niche preferences, as described in § 5.1. The underlying datasets are introduced in § 5.2, followed in § 5.3 by the baselines against which the proposed method is compared. Evaluation metrics in § 5.4 cover both recommendation performance and properties of the generated explanations, enabling a joint view on accuracy and popularity alignment. Finally, § 5.5 details the implementation choices required to reproduce the experiments.

5.1. Defining User Groups

As this work attempts to analyze how well users align with an explanation through a lens of popularity bias, users are first categorized into distinct groups based on their historical preferences. This approach follows the methodology previously proposed by Abdollahpouri et al, who investigate popularity bias from a user-centric perspective [1]. The key idea in their work is to understand how a user's inclination towards popular or niche items interacts with the behavior of the RS. This work looks at the interaction with the counterfactual example.

Items are first classified based on their popularity, where item popularity is computed by considering the fraction of total interactions involving the item (Equation 5.1). After computing item popularity scores, all items are sorted. The top 20% are defined as popular and the bottom 20% as niche. All other items are classified as diverse.

$$popularity_i = \frac{\text{number of interactions with } i}{\text{total number of interactions}} \tag{5.1}$$

The fraction of each item category in each users' history is calculated, which is in turn used to categorize users into niche, blockbuster, and diverse:

- **Niche users**: These are defined as the bottom 20% of users in terms of the ratio of popular items in their profile.
- **Blockbuster users**: These are defined as the top 20% of users in terms of the ratio of popular items in their profile.
- **Diverse users**: These are defined as all of the users that do not fit into the niche or blockbuster categories.

This grouping enables us to examine how counterfactual explanations behave across different user types, reflecting the heterogeneity in user preferences that is often overlooked in global evaluations.

5.2. Datasets

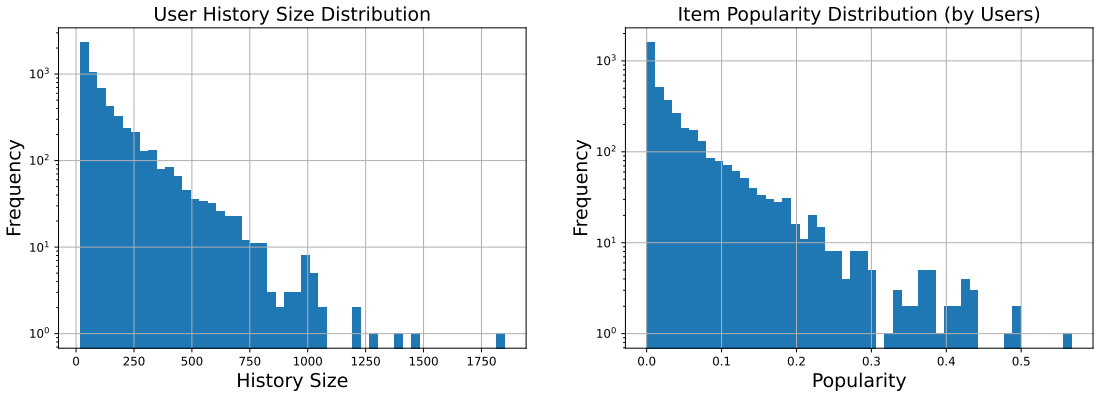
To evaluate the proposed approach, two multimedia-oriented datasets are utilized: the MovieLens 1M dataset [17] and the Amazon Video Games dataset, a subset of the Amazon Reviews 2023 dataset [21]. Multimedia-oriented datasets were chosen as they provide not only user-item interactions, but also the rich item metadata, including textual descriptions and categories required for the proposed pre-processing step. Table 5.1 presents the high-level details of the datasets.

Dataset	No. Users	No. Items	No. Interactions	Density
MovieLens 1M	6,040	3,952	1,000,209	4.19%
Amazon VideoGames (5-core)	94,761	25,612	814,587	0.03%

Table 5.1: High-level dataset statistics.

In each dataset 20% of every user's interactions was held out to construct the test set, with the remaining 80% used for training.

MovieLens 1M The MovieLens 1M dataset is a well-established benchmark in recommender systems. It contains just over one million user-item interactions in the form of movie ratings, making the MovieLens data dense relative to typical recommendation datasets.



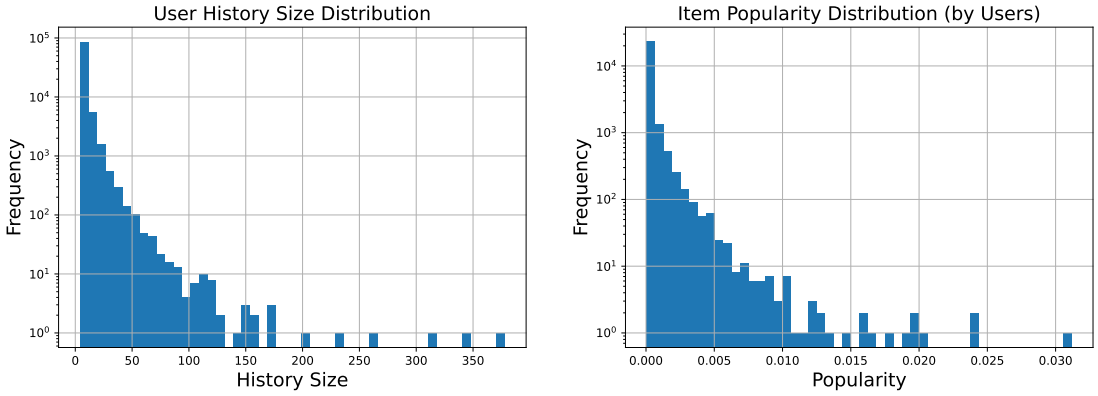
(a) User profile length distribution.

(b) Item popularity distribution.

Figure 5.1: Distributions in our training split of the MovieLens 1M dataset (log-scaled y-axis). Left: most users have 20–200 interactions, indicating density. Right: both popular and niche items are present, but the long tail is less extreme compared to Amazon Video Games.

Figure 5.1 shows the history size and item popularity distributions for MovieLens 1M. Panel 5.1a illustrates that most users have relatively long histories, with the median history consisting of 76 interactions and many users providing several hundred ratings. Panel 5.1b shows the popularity distribution: while some movies attract many interactions, the distribution is much less skewed than in e-commerce domains. Both blockbuster and niche titles are represented in meaningful quantities.

Amazon Video Games The Amazon Video Games dataset is a category-specific subset of the Amazon Reviews 2023 dataset. The 5-core version was used, ensuring all users and items have at least five interactions. This results in a significantly sparser dataset than MovieLens, with a long-tail distribution in both item and user activity. In addition to core video game titles, the dataset contains peripheral items such as accessories and collectibles. These are kept to preserve realistic user profiles, but could introduce noise.



(a) User profile length distribution.

(b) Item popularity distribution.

Figure 5.2: Distributions in our training split of the Amazon Video Games dataset (log-scaled y-axis). Left: most users have very short histories, reflecting extreme sparsity. Right: a few blockbuster items dominate interactions, while the majority fall into the long tail.

Figure 5.2 highlights the contrasting properties of the Amazon Video Games dataset. Panel 5.2a shows that most users interact with only a few items, with the median history size being only 4. Panel 5.2b shows that the popularity distribution follows an extreme long tail pattern, with a couple of blockbuster items dominating. The vast majority of items are rarely consumed.

Filtering Due to the nature of the LLM-based approach, generating a single misaligned item set can take a considerable amount of time, particularly for users with large histories. Because evaluating every user in full would exceed the available computational resources, two filtering steps are applied. First, the maximum history size is limited: generating a textual user profile with the LLM-based approach described in § 4.2 takes 10–15 seconds per query, depending on the model, hardware, and query length. Since the number of queries scales with both the number of problematic items sought (n) and the size of the user history (s), the overall cost grows quickly ($n \times s$). To keep the evaluation feasible, users with more than 100 interactions are excluded. Second, the focus is placed on users most likely to exhibit alignment issues. Specifically, the 250 most niche and 250 most blockbuster users are selected for each dataset, as these extremes are expected to show the most pronounced gaps between model-generated and user-expected explanations.

5.3. Baselines

To assess the effectiveness of the proposed framework, its counterfactual sets are compared against several baselines. Two categories are distinguished: direct counterfactual baselines, which generate complete counterfactual sets directly, and alternate-removal baselines, which first identify potentially misaligned items and then apply ACCENT on the filtered history.

Direct The direct baselines represent alternative ways of constructing counterfactual sets from a user's history. Three methods are considered:

- **ACCENT**, which explains recommendations via influence-based counterfactuals. ACCENT selects items from the user's history that, when removed, change the top-1 prediction. This serves as the primary baseline, since the proposed framework builds directly upon it.
- **Popularity-based counterfactuals**, where counterfactual sets are constructed from the most popular items in the user's history, with set sizes matched to those produced by ACCENT.
- **Random counterfactuals**, a lower-bound baseline that selects items uniformly at random from the user's history, used to assess whether more sophisticated methods provide benefits beyond chance.

Alternate removals The alternate-removal baselines evaluate the effect of different filtering strategies when combined with ACCENT. In this setting, a set of potentially misaligned items is first removed from the user's history, after which ACCENT is applied to generate counterfactuals on the reduced profile. Three variants are considered:

- **FIA-based removal**, which selects the most influential items based on raw influence scores, providing a baseline that uses the unprocessed influence signal.
- **Popularity-based removal**, which removes the most popular items from the user's history before running ACCENT.
- **Random removal**, which removes a random subset of history items prior to running ACCENT.

5.4. Evaluation Metrics

The updated counterfactuals are evaluated along two main dimensions: user alignment, i.e., whether the counterfactuals reflect the user's underlying preferences, and validity, i.e., whether the counterfactual effectively alters the recommendation outcome.

User alignment In the user-alignment axis, the analysis examines whether the suggested approaches result in counterfactuals that are closer in content to what the user actually prefers. As the augmentations rely on a general "*does this suit the user?*" test rather than on attribute-specific rules, they should, in theory, alleviate many kinds of mismatch. The approach is evaluated solely on one well-known dimension, popularity bias. Two new metrics are proposed.

Popularity Distribution Similarity (PDS): To evaluate how well the counterfactual sets align with users' general taste profiles, the popularity distribution of items in the counterfactual sets is compared to that of the items in each user's history. Let H be the complete collection of items that appear in users' interaction histories and C the collection of items that appear in the generated counterfactual explanations. If an item occurs multiple times (e.g., across different users), it is included that many times in the collection. Each item is converted to its global popularity score and the scores are placed into B equal-width bins, producing two histograms (empirical distributions) P_H and P_C. The similarity between the two popularity profiles can then be defined as the χ^2 distance:

$$\text{PDS} = \sum_{b=1}^{B} \frac{(P_H(b) - P_C(b))^2}{P_C(b) + \epsilon} \tag{5.2}$$

with $\epsilon = 10^{-10}$ for numerical stability. A smaller PDS indicates that counterfactual items follow nearly the same popularity distribution as the historical items, suggesting lower popularity bias and better alignment.

Expected Popularity Deviation (EPD): EPD is introduced as a more fine-grained, user-level metric. For each user, the average popularity of items in their history and in their counterfactual set is computed, and the squared difference is taken:

$$\text{EPD}u = (p^u cf - p^u_{hist})^2 \tag{5.3}$$

This value is then averaged over all users to obtain a global view:

$$\text{EPD} = \frac{1}{|U|} \sum_{u \in U} \text{EPD}_u \tag{5.4}$$

A lower EPD implies that the counterfactuals are more similar in popularity distribution to the user's original profile, which can be interpreted as a proxy for better personalization.

Validity The validity of a counterfactual refers to whether it effectively alters the recommendation outcome. This section discusses established approaches–necessity, sufficiency, and counterfactual set size–and introduces an aspect-based evaluation.

Necessity and sufficiency: A natural way to evaluate counterfactual explanations is to retrain the recommender system after removing the training points associated with the explanation set E_u, and observing whether the explained recommendation disappears. This strategy is used by ACCENT, and formalized as Counterfactual Proximity by Yao et al. [67]. While this provides a faithful measure of causal inference, it is computationally expensive, as a new recommender must be trained for every explanation that is evaluated.

Instead, the same reasoning process is approximated using a lightweight surrogate model that captures how a user's history H_u affects the likelihood of recommending a specific item, as is done by other explanation approaches [42]. For each evaluation, a lightweight surrogate classifier f_{sur} is trained that predicts the probability a users top-1 recommended item x would be recommended given their interaction history. The surrogate is trained on all users' histories, where the target label is set to 1 if x appeared in that user's top-K recommendations, and 0 otherwise. Once trained, f_{sur} can be queried multiple times with modified versions of the same user history to estimate how the recommendation of x changes when certain items are removed or isolated.

The effects of these modified user histories are quantified using the necessity and sufficiency framework proposed by Watson et al. [63], which previous works also have used to evaluate counterfactuals [56]. Necessity measures whether the explanation items are required for the recommendation to occur, and thus directly corresponds to counterfactual proximity:

$$\text{Necessity}(u, x, E_u) = f_{sur}(H_u) - f_{sur}(H_u \setminus E_u) \tag{5.5}$$

Sufficiency measures whether the items in the counterfactual set on their own are capable of reproducing the recommendation:

$$\text{Sufficiency}(u, x, E_u) = f_{sur}(E_u) \tag{5.6}$$

Both necessity and sufficiency speak to the quality of the counterfactual: a high necessity score means the explanation captures factors responsible for recommending x, while a high sufficiency implies the explanation is self-consistent.

Counterfactual size: While the size of the counterfactual set does not directly determine its quality, it significantly impacts its interpretability. A smaller counterfactual set allows users to examine and comprehend the explanation more easily. This follows naturally from the principle of cognitive load, which states that humans process and interpret smaller sets of information more effectively.

Aspect overlap: Counterfactuals are also evaluated at a semantic level using aspect overlap. An LLM is prompted to extract key aspects from a user's interaction history and, separately, from the items in the counterfactual set. The underlying intuition is that a counterfactual explanation is more meaningful when it appeals to the same properties that matter to the user. If the user's history repeatedly highlights aspects such as "small form factor" and "RGB lighting" for peripherals, then a counterfactual set for a recommended headset is more convincing when those same aspects are prominent. The explanation then essentially says that the recommendation fits the user for the same reasons they liked the previous items. Conversely, if the counterfactual items revolve around unrelated aspects, the justification may feel arbitrary or unpersuasive. Greater overlap between aspects in the user history and in the counterfactual items therefore serves as a signal that the explanation is grounded in the user's established preferences.

5.5. Implementation Details

For both the embedding model approach and the embedding of textual user profiles generated by LLMs, Mixbread's mxbai-embed-large-v1 model [32] was used. This open-weight model achieves state-of-the-art performance in its size class. All embedding operations were performed using the SentenceTransformers (SBERT) Python library [43].

The performance of a small and medium-sized recent open-weight large language model is compared on our LLM-augmented counterfactuals, neither of which was fine-tuned on multimedia data. For the small model, Gemma3-4B was used, an lightweight 4-billion-parameter instruction-tuned model developed by Google DeepMind as part of the open Gemma3 model family [57], based on the same research and technology that powers the Gemini 2.0 models. Gemma3-4B provides strong performance on instruction-following tasks at a much lower resource cost than larger models. For the medium-sized model, Qwen3-8B was used, an 8-billion-parameter instruction-tuned model developed by Alibaba Group as part of the Qwen3 series [66]. Qwen3 allows for both reasoning and traditional queries. In our experiments, the reasoning functionality is not used. A quantized model (Q3_K_S) created using Unsloth was used to improve performance [9]. For the aspect overlap evaluation, the Deepseek v3 API was used [10].

For recommendation, a standard Neural Matrix Factorization (NeuMF) model was used [19]. The model was trained with an embedding size of 16, a learning rate of 1e-3, and a weight decay of 1e-3. The implementation and hyperparameters of the recommendation model and the counterfactual approach were forked from the publicly available ACCENT codebase [1]. The implementation of the proposed method can be found at https://github.com/ahasami/llm-augmented-cfs.

[1] https://github.com/hieptk/accent

6

Results

In this chapter, the empirical results of the study are presented. § 6.1 addresses the first research question by examining the extent of misalignment in raw ACCENT. § 6.2 then considers the second research question, analyzing how LLM-augmented ACCENT improves the quality of counterfactual explanations. Finally, § 6.3 discusses the third research question, evaluating the impact of LLM size on performance

The first step towards answering our research questions, is the training of two Neural Collaborative Filtering models for MovieLens 1M and Amazon Video Games respectively. Using the negative-sampling based nDCG calculation as described in § 2.1, nDCG@10 scores of 1.303 and 0.0481 are obtained on MovieLens 1M and Amazon Video Games, respectively. This difference in performance can be expected when looking at Table 5.1, where the Amazon dataset has a density of only 0.03% compared to 4.19% for MovieLens.

6.1. Misalignment in ACCENT

The first research question asks, "*To what extent are counterfactual explanations generated by ACCENT affected by misaligned items?*". This question is examined from a popularity-bias perspective using two proposed metrics: Popularity Distribution Similarity (PDS) and Expected Popularity Deviation (EPD). A total of 736 user histories across two datasets, MovieLens 1M and Amazon VideoGames, were selected based on their inclination towards popular or niche items, with users in the extremes being selected. For these users, counterfactual explanations were generated and inspected.

Heuristic baselines The PDS and EPD values for the baselines in Table 6.2 show that the heuristic CF-set construction approaches perform worst across both datasets. Sets composed of the most popular items yield extremely high PDS values, indicating severe misalignment. Although the corresponding EPD values may appear small (e.g., 0.095 for niche users in ML-1M), these numbers need to be interpreted in the context of the underlying popularity scale. Table 6.1 shows that the average popularity of items in ML-1M is only 0.043, with niche items averaging as low as 0.0008. Against this backdrop, a deviation of 0.095 means that the counterfactual sets for niche users are, on average, shifted toward items more than twice as popular as the user's historical norm. What looks like a small numerical value, therefore, represents a substantial misalignment.

For the random CF-set baseline, the misalignment is less extreme than for top-popular but still clear. Since items are sampled uniformly from histories, these sets largely reflect the dataset average rather than user-specific patterns. This is visible in the high PDS values (e.g., 20.65 for niche users in ML-1M and 6.15 in Amazon), which shows a poor match with user histories. Unlike the top-popular baseline, this mismatch is not caused by an explicit bias toward popular items, but rather by the randomness of the sampling, which fails to preserve the characteristic

skew of individual user profiles

In sum, top-popular counterfactuals illustrate the extreme case of popularity bias, while random sets highlight the incoherence of ignoring user alignment altogether. These baselines are thus reported for comparison.

Dataset	Item Group	Group Size	Avg. Popularity	Std. Dev.
ML-1M	Niche	776	0.0008	0.0008
	Diverse	2330	0.0234	0.0179
	Blockbuster	777	0.1421	0.0775
	Total	**3883**	**0.0426**	**0.0628**
Amazon	Niche	5122	$5.68 * 10^{-5}$	$5.13 * 10^{-6}$
	Diverse	15367	$1.47 * 10^{-4}$	$7.17 * 10^{-5}$
	Blockbuster	5123	$1.18 * 10^{-3}$	$1.57 * 10^{-3}$
	Total	**25612**	$\mathbf{3.36 * 10^{-4}}$	$\mathbf{8.24 * 10^{-4}}$

Table 6.1: Average popularity of items, measured by the proportion of users who interacted with them. Items are grouped into three categories: the top 20% most popular (blockbuster), the bottom 20% least popular (niche), and the remaining 60% (diverse).

ACCENT ACCENT's counterfactual explanations are also affected by popularity bias, though less so than the heuristic baselines. On ML-1M, niche users obtain a PDS of 7.12 and blockbuster users 18.36, both higher than the random-augmentation or top-influential strategies discussed later. On Amazon, the gap between user groups becomes even clearer: niche users have a PDS of 3.54, while blockbuster users reach 14.29. The corresponding EPD values, although numerically small, consistently indicate that ACCENT counterfactuals are shifted toward more popular items than those in the user histories. The effect is particularly strong for blockbuster users, whose explanations tend to overemphasize already popular items, thereby amplifying existing bias. This pattern is visible across both datasets and confirms that ACCENT is not neutral to popularity, but systematically misaligned for large groups of users.

Answer to RQ1

Counterfactual explanations generated by ACCENT are indeed affected by misaligned items. Both in ML-1M and Amazon, the popularity distribution of ACCENT explanations diverges significantly from user histories. While less extreme than naïve heuristics, these results show that ACCENT counterfactuals are systematically misaligned in popularity, failing to fully reflect user preferences.

6.2. Language models for increased alignment

The second research question asks, "*How effectively can the proposed LLM-augmented framework improve the generation of counterfactual explanations?*". While the first question established that ACCENT itself is prone to misalignment, the next step is to evaluate whether augmenting ACCENT with additional filtering strategies can address this issue. Returning to the same set of 736 users introduced earlier, the augmentation procedure described in Chapter 4 is applied, where potentially misaligned items are removed from user histories before running ACCENT. First, simple augmentation baselines are examined, and then they are compared against the proposed language-model-based filtering.

Augmentation baselines The augmentation baselines that build on ACCENT by filtering misaligned items perform consistently better than the direct set-creation baselines. For the random and top-popular variants, EPD values remain close to those of randomly constructed counterfactual sets, indicating limited improvement. However, the PDS scores improve notably for random removal compared to random set creation, showing that ACCENT itself imposes some useful

Dataset	Method Type	Method	Niche		Blockbuster	
			PDS ↓	EPD ↓	PDS ↓	EPD ↓
ML-1M	Heuristic	Random	20.65	0.014	11.24	0.015
		Top Popular	1290.24	0.095	839.63	0.057
	Augmentation	Random	7.38	0.011	7.93	0.016
		Top Popular	14.48	0.013	18.41	0.015
		Top Influential	2.69	**0.007**	7.05	**0.012**
	SOTA	ACCENT	7.12	0.010	18.36	0.015
	Proposed	Embedding-Augmented	**2.42**	0.010	7.36	0.016
		LLM-Augmented (4B)	7.49	0.008	10.39	0.017
		LLM-Augmented (8B)	5.30	0.008	**6.26**	0.014
Amazon	Heuristic	Random	6.15	$4.1 * 10^{-9}$	3.12	$1.3 * 10^{-9}$
		Top Popular	199.99	$9.0 * 10^{-9}$	165.87	$4.2 * 10^{-5}$
	Augmentation	Random	**2.14**	$4.1 * 10^{-9}$	**1.63**	$1.6 * 10^{-5}$
		Top Popular	4.59	$4.2 * 10^{-9}$	8.34	$1.2 * 10^{-5}$
		Top Influential	3.16	$\mathbf{2.9 * 10^{-9}}$	5.78	$\mathbf{1.1 * 10^{-5}}$
	SOTA	ACCENT	3.54	$4.1 * 10^{-9}$	14.29	$1.4 * 10^{-5}$
	Proposed	Embedding-Augmented	2.93	$3.8 * 10^{-9}$	6.77	$1.2 * 10^{-5}$
		LLM-Augmented (4B)	4.77	$4.0 * 10^{-9}$	6.30	$1.6 * 10^{-5}$
		LLM-Augmented (8B)	3.35	$3.7 * 10^{-9}$	6.06	$1.4 * 10^{-5}$

Table 6.2: Popularity alignment comparison of methods with respect to item groups on ML-1M and Amazon datasets.

structure, even when combined with a naive filtering strategy. For example, in ML-1M, the random augmentation reduces the PDS for niche users from 20.65 (random set creation) to 7.38, bringing it in line with the 7.12 score achieved by ACCENT. On the Amazon dataset, random augmentation unexpectedly yields the best PDS scores. This likely reflects the extreme sparsity of Amazon user histories, often consisting of only four or five interactions. Here, removing a single item has a disproportionate effect, making results less stable and harder to generalize. Running ACCENT with the top-popular items removed also distorts the PDS scores for both datasets, though less severely than in the direct top-popular set creation, which again suggests that ACCENT counteracts the introduced bias.

The top-influential strategy is particularly noteworthy among the baselines. It achieves the lowest overall EPD for both datasets and user groups. This supports the hypothesis that influence functions often surface items that are impactful for the model but not representative of the user's actual preferences. By filtering out these items before applying ACCENT, the resulting counterfactuals align more closely with user histories.

Proposed approach Turning to the proposed methods, which augment ACCENT by excluding misaligned items found with either an embedding-based filter or a large language model filter, and remove them from the CF generation process.

The embedding-based variant shows mixed results. On MovieLens, it is particularly effective for niche users, lowering the PDS from 7.12 with ACCENT to 2.42. This indicates that the embedding filter can identify and exclude popular outliers that distort explanations for niche users. For blockbuster users, the method does not improve over ACCENT (PDS 7.36 vs. 7.05), suggesting that embeddings are less effective at capturing mainstream alignment. On Amazon, PDS for blockbuster users is lowered from 14.29 to 6.77, but niche users only improve marginally (3.54 vs. 2.93). EPD values remain close to baseline across all user groups and datasets.

LLM-based augmentation produces more consistent improvements. On MovieLens, the PDS for blockbusters falls sharply from 18.36 with ACCENT to 6.26, while niche users also improve,

though less dramatically (7.12 vs. 5.30). On Amazon, the effect is similar: blockbuster users see clear gains (14.29 vs. 6.06), whereas niche users improve only marginally (3.54 vs. 3.35). A slight improvement in EPD for all user groups and datasets is also visible.

Taken together, these results show that while embeddings can help, their impact is limited and can be inconsistent. The LLM-based approach delivers more consistent improvements across datasets, particularly in scenarios where ACCENT produces the largest misalignment. This makes LLMs a more effective strategy for improving user alignment of counterfactuals. Further strengthening this point, the only result found to be statistically significant improvement over ACCENT was the overall EPD on MovieLens (blockbuster and niche combined, not shown) using LLM-augmentation (8B). The proposed approach outperformed ACCENT (0.017 vs 0.014, $p < 0.05$).

Steering the explanations Beyond improvements in PDS and EPD, Figure 6.1 illustrates how augmentation strategies actively steer the composition of counterfactual explanations. The plots show the shift in average counterfactual popularity against the shift in position within the popularity-sorted user history when moving from pure ACCENT to the augmented variants.

In MovieLens, the filtered counterfactuals move downward and slightly rightward for bins of niche users (left panels), indicating less-popular items drawn from deeper in each user's long-tail history. For the bins of blockbuster users (right panel), the same mechanism pushes explanations upward and leftward, highlighting even more popular items near the head of the profile. The result is a bidirectional correction. The same analysis on Amazon is noisier. Some bins of blockbuster users benefit from a popularity boost, and for niche users, the range of the user profile from which counterfactuals are generated is tightened.

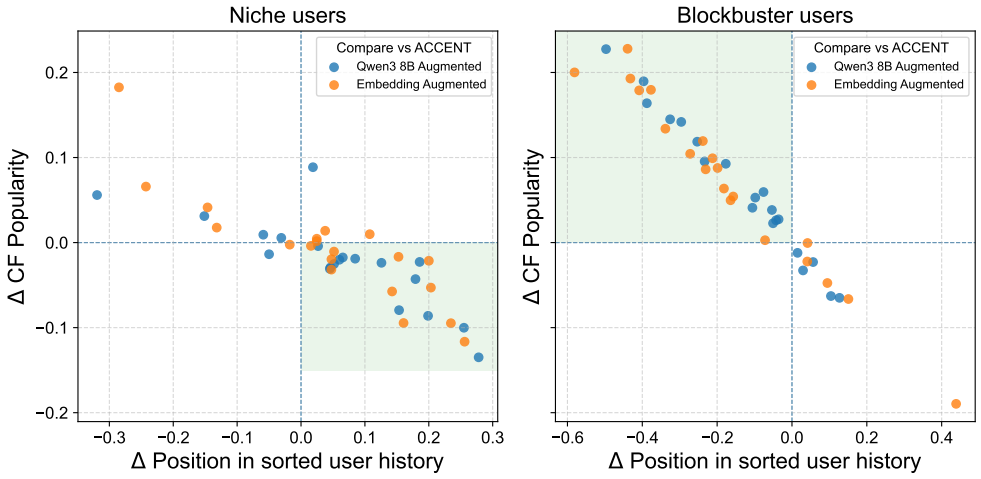
Quality of explanations The effect of the augmentation on the overall quality of the counterfactual explanation was also examined. Table 6.3 shows that the average CF set sizes remain nearly identical across methods, indicating no loss in conciseness. Aspect overlap improves slightly on MovieLens (0.207 vs. 0.189) and stay stable on Amazon. This suggests that augmented sets do not harm fidelity, and might even better capture the themes of the explained items. The necessity and sufficiency scores also remain comparable between the original and augmented methods across both datasets, indicating that the augmented counterfactuals retain their causal validity.

These results suggest that the augmentation does not degrade explanation quality, which is expected: since the LLM-augmentation is a pre-processing step, the explanations are in the end still found by ACCENT, and thus should maintain the same quality in underlying causal relationships.

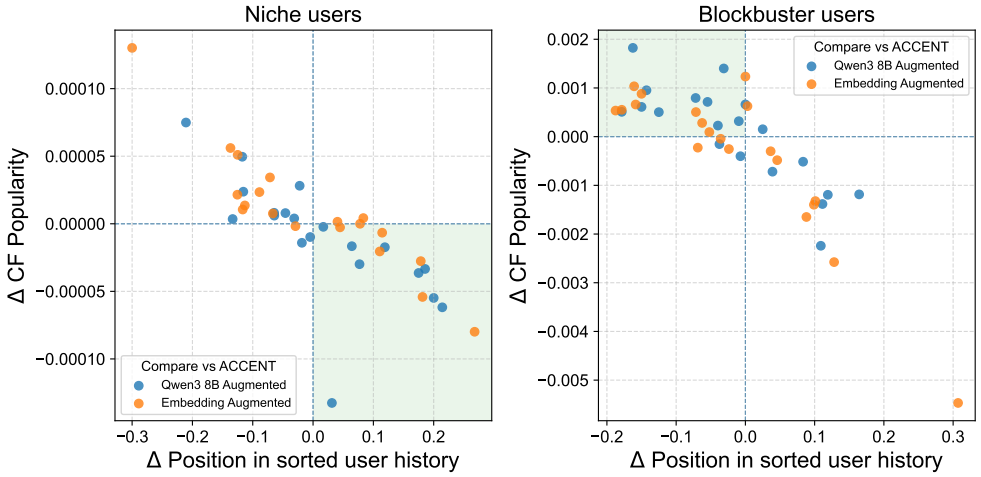
Dataset	Method	Mean CF Set Size	Aspect Overlap	Necessity	Sufficiency
ML-1M	ACCENT	1.35	0.189	**0.019**	**0.386**
	LLM-Augmented (8B)	**1.32**	**0.207**	0.012	0.379
Amazon	ACCENT	1.19	**0.270**	**0.100**	0.259
	LLM-Augmented (8B)	**1.13**	0.269	0.096	**0.260**

Table 6.3

Explanation coverage A slight decrease in cases where an explanation can be found is generally noticed after augmenting ACCENT. Table 6.4 shows that coverage on MovieLens drops from 80.2% to 78.8% with LLM augmentation (8B), and similar small reductions appear for most baselines. On Amazon, coverage remains closer to ACCENT levels, with LLM augmentation (8B) reaching 72.7% versus 72.9%. These results indicate that filtering occasionally prevents ACCENT from producing a valid counterfactual, but the effect remains minor.



(a) MovieLens 1M, $n = 5$.



(b) Amazon VideoGames, $n = 1$

Figure 6.1: Average popularity of counterfactual sets versus average position of counterfactual items in the normalized popularity-sorted user history, delta view. Dots represent the shift made from pure ACCENT to the variants of augmented ACCENT. Users have been grouped into 20 bins based on the average popularity of the user history.

Answer to RQ2

The results show that LLM augmentation improves the alignment of CFEs, especially on ML-1M, where user histories are richer and more stable. Bidirectional corrections are consistent: niche users get less popular explanations, while blockbuster users are nudged toward mainstream ones. Gains in EPD on ML-1M are small but statistically significant. On Amazon, impact is weaker—sparser histories limit the LLM's ability to infer profiles and guide filtering. Still, the method generally does not harm performance, suggesting it remains safe to apply even with limited effect.

6.3. Impact of LLM size

The third and final research question asks, "*How does the size of the LLM influence the quality of the counterfactual explanations?*". This question arises from the observation that larger language models generally provide stronger reasoning capabilities and more nuanced contextual understanding, at the cost of increased computational requirements. To examine this trade-off, the performance of a smaller model (Gemma3 4B) is compared with the previously examined

Dataset	Method	Coverage
ML-1M	ACCENT	80.2%
	Random Augmented	78.8%
	Top Popular Augmented	80.5%
	Top Influential Augmented	78.5%
	Embedding-Augmented	79.9%
	LLM-Augmented (8B)	78.8%
	LLM-Augmented (4B)	79.9%
Amazon	ACCENT	72.9%
	Random Augmented	70.0%
	Top Popular Augmented	70.0%
	Top Influential Augmented	70.2%
	Embedding-Augmented	70.4%
	LLM-Augmented (8B)	72.7%
	LLM-Augmented (4B)	69.3%

Table 6.4

medium-sized model (Qwen3 8B). This allows us to assess whether scaling the LLM improves the alignment of the explanations, and if those improvements justify the higher resource footprint.

Quantitative results in Table 6.2 show that scaling from 4B to 8B generally improves alignment. On MovieLens, the 8B model reduces PDS for blockbuster users from 10.39 to 6.26, and for niche users from 7.49 to 5.30. EPD stays stable with only a slight improvement for blockbuster users with the larger model. On Amazon, improvements are smaller but follow suit, with blockbuster users again seeing the largest improvements.

In qualitative evaluation, the 4B model more often failed to follow instructions and occasionally produced hallucinatory summaries, which sometimes led to irrelevant filtering. For instance, despite explicit instructions not to mention specific item titles, it frequently did so. In one instance, the response even switched to a different language halfway through generating the summary. In contrast, the 8B model adhered to the prompt format much more consistently and avoided such failures. This indicates that larger models not only improve alignment metrics but also enhance the reliability of the filtering process.

Answer to RQ3

Increaing model size improves both alignment and reliability of LLM-augmented ACCENT. The 8B model consistently outperforms the 4B model, following instructions more closely and producing fewer hallucinations. However, both models remain relatively small in modern LLM terms, and it is likely that larger models could yield further gains, although the increased computational cost may not be justified for all applications.

7 Discussion

This thesis set out to investigate how counterfactual explanations in recommender systems can be made more user-aligned, with a particular focus on mitigating popularity bias. Through a series of experiments on MovieLens and Amazon VideoGames, we evaluated the baseline behavior of ACCENT, tested several augmentation strategies, and proposed LLM-based filtering as a new approach. The three research questions addressed whether ACCENT counterfactuals suffer from misalignment, whether augmentation can mitigate this issue, and how the size of the LLM affects performance.

The results indicate that ACCENT explanations are indeed misaligned, particularly for blockbuster-oriented users, and that naive heuristics or embedding-based corrections provide only limited improvements. LLM-based augmentation yields more consistent gains, especially on MovieLens, and exhibits a bidirectional correction effect. In the remainder of this chapter, § 7.1 reflects on the implications of these findings, § 7.2 discusses the main limitations of the study, and § 7.3 outlines promising directions for future research. Finally, § 7.4 discusses the computational complexity of the pre-processing steps based on language models.

7.1. Implications

The findings of this work have several implications for the design, evaluation, and deployment of counterfactual explanations in recommender systems.

First, it is demonstrated that user alignment cannot be assumed, even for well-established explanation frameworks. Our results show that ACCENT counterfactuals can diverge significantly from user histories, particularly for blockbuster-oriented users. Explanations that amplify popularity bias risk being dismissed as generic, thereby failing to foster trust. By quantifying this misalignment with PDS and EPD, we establish that alignment should be treated as a possible evaluation dimension in future explanation research, alongside the more common fidelity metrics.

Second, the study shows that LLMs can serve as a pre-processing step in the counterfactual explanation pipeline. Language models are able to reason about the coherence of items with the context of a user's preferences, enabling corrections that adapt dynamically to user group characteristics. This is underscored by the bidirectional correction observed: instead of shifting all explanations toward niche content or the global average, explanations are adjusted to be more popular for blockbuster users and downward for niche users. This is in line with research that use LLMs for other tasks in other areas, such as bias mitigation [33].

The findings also highlight a tension between alignment and efficiency. The improvements provided by LLM augmentation are most evident in MovieLens, where histories are richer, and for blockbuster users, where ACCENT is most misaligned. Yet, these gains come with steep computational cost. This raises practical questions: should LLM augmentation be deployed universally,

or selectively applied only to user groups where it has the largest impact? These considerations are key for large scale, real-world deployment.

Finally, the results show that the evaluation practices for explainable recommender systems have to evolve. As mentioned in § 2.2, evaluating explanations is a task that is already hard due to their subjective nature, and many published works fail to rigorously evaluate their methods [68]. In this work, we have aimed to expose the alignment domain, and popularity-aware metrics such as PDS and EPD provide one lens for this. Further dimensions, such as diversity, novelty and fairness should also be considered to obtain a broader picture. The success of the LLM-based method in improving alignment demonstrates that explanation research can benefit from integrating user-centric metrics more systematically.

7.2. Limitations

While our LLM-augmentation step shows promising improvements in user alignment, several limitations remain.

A first limitation concerns data sparsity. In sparse datasets such as Amazon, where user histories often consist of only a few interactions, the deletion of even a single item can disproportionately distort the profile. In these cases, the LLM may struggle to infer coherent preferences, as the available data is too limited. This explains why the benefits observed on MovieLens do not always transfer to Amazon, and suggest a minimum history density might be required.

Second, the use of language models introduces its own challenges. LLMs are known to generate inconsistent, overly generic, or even hallucinatory results, and such errors directly affect the quality of the filtering. While these issues do not strongly impact global evaluation scores, they imply that a small number of individual users may actually receive worse explanations than with unaugmented ACCENT. Although careful prompt design, validation, and model choice can mitigate these risks, occasional failures remain unavoidable. A related concern is the black-box nature of LLMs: since the goal of counterfactual explanations is to make the recommender model more transparent, there is an inherent tension in relying on another opaque model to achieve this.

Third, the computational cost of the LLM-based approach is high. Each iteration requires multiple LLM calls proportional to the size of the user history, leading to runtimes far greater than embedding-based methods. A full complexity analysis is provided in § 7.4, but in practice this overhead might limit the scalability of the approach to large-scale or real-time applications.

A minor point of consideration is the reduced counterfactual coverage. By removing misaligned items before running ACCENT, the method reduces the pool of candidate items available for constructing explanations. A simple fallback is to return the raw ACCENT explanation if it exists, which ensures that overall coverage matches the baseline.

Finally, while we report statistically significant improvement for the overall EPD on MovieLens, most gains are relatively small in absolute terms. This raises the issue of practical significance. A conclusive answer requires user studies that assess perceived alignment, trust, and satisfaction, as offline metrics alone cannot fully capture the subjective nature of explanations.

7.3. Future work

Several avenues remain open for extending and deepening this work.

A first direction is to move beyond pre-processing and examine the impact of removing misaligned items on the recommendations themselves. In this thesis, the LLM-augmentation step was applied purely as a filter prior to running ACCENT, without modifying the recommendation model or its outputs. Future research could retrain recommenders on histories where misaligned items have been removed, or simulate this effect through the use of influence functions. Such experiments would clarify whether misaligned items not only distort explanations but also degrade the recommendations themselves.

A second avenue is to conduct user studies. While PDS and EPD provide useful signals about

popularity alignment, they do not capture how users perceive explanations in practice. User studies could assess whether LLM-augmented explanations are indeed considered more faithful, whether the bidirectional correction is noticed and valued, and whether alignment improvements translate into higher trust or satisfaction. Such studies would also shed light on the practical significance of improvements that are statistically measurable but small in absolute terms. They would also allow for a broader evaluation than looking at only the popularity; as LLM-augmentation relies on a general ”does this suit the user?” test rather than on attribute-specific rules, it should, in principle, alleviate many kinds of mismatch.

A third line of inquiry involves testing different recommendation models. The experiments in this thesis were carried out on standard NCF-based recommenders, which are themselves known to be popularity-biased. It remains unclear whether the misalignment observed originates primarily from the recommender or from the counterfactual explanation framework. Running the same pipeline with debiased recommendation models would reveal whether LLM augmentation still adds value in systems that already mitigate different forms of bias. Similarly, applying the method to alternative counterfactual explanation frameworks beyond ACCENT would test whether the improvements generalize across explanation paradigms.

The final direction concerns methods for identifying misaligned items. While we explored embeddings, influence scores, and LLMs, many more variants are possible. Larger LLMs may offer stronger reasoning capabilities, while fine-tuned models trained on domain-specific data (e.g., reviews or product descriptions) could improve both accuracy and robustness. Hybrid approaches could also increase efficiency: embeddings could first narrow down candidate misaligned items, after which a smaller or distilled LLM provides the final decision. Exploring these options would help balance alignment quality with computational cost.

7.4. Computational Complexity

The computational complexity of our approach depends on the method used for identifying problematic items.

Embedding-based identification Let s be the number of items in the user's interaction history, and n the number of items to be selected. The first step in the algorithm is to create the complete user embedding. Vector addition is $O(d)$ for d-dimensional vectors, giving us a time complexity of $O(s * d)$ for creating the full user embedding. Then, in each of the n greedy iterations, the algorithm evaluates the influence of removing each of the remaining items in H_u. Each item being evaluated requires a vector subtraction and cosine similarity computation, both of which are $O(d)$ for d-dimensional embeddings. This yields a total complexity of $O((s*d)+(n*(s*d)+(s*d)))$ per user. This simplifies to $O(n*d*s)$. Since the embeddings can be pre-computed, we do not need to consider the inference of the embedding model.

LLM-based identification The LLM-based approach is more costly. Each item being evaluated requires generating a user summary with the LLM and embedding that summary. Assuming LLM generation and embedding take constant time T_{LLM} per call (which in actuality is not the case, as LLM inference is not independent from s), each iteration involves s such calls, resulting in a total complexity of $O(n * s * T_{\text{LLM}})$. Since $T_{\text{LLM}} \gg d$, this method is significantly slower in practice, particularly for large s.

While both methods are greedy and avoid an exhaustive subset search, the LLM-based method incurs a significantly higher real-world runtime due to the repeated generation of prompts.

8

Summary

This thesis investigates how counterfactual explanations (CFEs) in recommender systems can be made more user-aligned, with a particular focus on mitigating popularity bias. While CFEs have the potential to increase transparency by showing users how recommendations would change under small perturbations, they risk losing credibility if the counterfactual sets are systematically misaligned with user preferences. To study this issue, three research questions were posed: (1) to what extent are ACCENT explanations affected by misaligned items, (2) how effectively can augmentation strategies, particularly large language models (LLMs), improve alignment, and (3) how does the size of the LLM influence the quality of the explanations?

Two NCF models were trained on MovieLens 1M and Amazon VideoGames. Using these recommenders, CFEs were generated for 736 extreme users (blockbuster- and niche-oriented), and their alignment with user histories was assessed. Two metrics were proposed for this purpose: popularity distribution similarity (PDS) and expected popularity deviation (EPD).

The core method introduced is LLM-based augmentation of ACCENT. An additional filtering step is applied before running ACCENT, where an LLM is prompted with the user's interaction history and candidate items, and greedily removing the items that, when removed, cause the largest shift in the user's profile. ACCENT then generates counterfactuals on the reduced history. In this way, the LLM acts as a filter that aligns explanations more closely with the user's preferences

The results show that ACCENT explanations are indeed misaligned, and the proposed LLM-based augmentation delivered the most consistent improvements out of the baselines, in particularly on MovieLens, where histories are richer. Importantly, it produced a bidirectional correction: for blockbuster users, counterfactuals became more popular, while for niche users, spurious popular items were removed. This adaptivity distinguishes the LLM approach from simpler heuristics. Statistical testing further showed that overall EPD on MovieLens improved significantly with LLM augmentation, making it the only variant to demonstrate measurable gains over ACCENT at scale.

In sum, this thesis demonstrates that misalignment is a real and measurable problem in counterfactual explanations, and that LLM-based augmentation offers a promising, if computationally costly, solution. The work contributes new metrics for evaluating alignment, empirical evidence of popularity bias in CFEs, and the first systematic study of LLMs for mitigating this bias.

References

[1] Himan Abdollahpouri et al. “The Unfairness of Popularity Bias in Recommendation”. In: *CoRR* abs/1907.13286 (2019). arXiv: 1907.13286. URL: http://arxiv.org/abs/1907.13286.

[2] Emily M. Bender et al. “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? □□”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. ISBN: 9781450383097. DOI: 10.1145/3442188.3445922. URL: https://doi.org/10.1145/3442188.3445922.

[3] James Bennett and Stan Lanning. *The Netflix Prize*. 2007.

[4] Jakub Černý et al. *Plausible Counterfactual Explanations of Recommendations*. 2025. arXiv: 2507.07919 [cs.LG]. URL: https://arxiv.org/abs/2507.07919.

[5] Jiawei Chen et al. “Bias and Debias in Recommender System: A Survey and Future Directions”. In: *ACM Trans. Inf. Syst.* 41.3 (Feb. 2023). ISSN: 1046-8188. DOI: 10.1145/3564284. URL: https://doi.org/10.1145/3564284.

[6] Ziheng Chen et al. *GREASE: Generate Factual and Counterfactual Explanations for GNN-based Recommendations*. 2022. arXiv: 2208.04222 [cs.IR]. URL: https://arxiv.org/abs/2208.04222.

[7] Weiyu Cheng et al. “Incorporating Interpretability into Latent Factor Models via Fast Influence Analysis”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 885–893. ISBN: 9781450362016. DOI: 10.1145/3292500.3330857. URL: https://doi.org/10.1145/3292500.3330857.

[8] Henriette Cramer et al. “The effects of transparency on trust in and acceptance of a content-based art recommender”. In: *User Modeling and User-Adapted Interaction* 18.5 (Nov. 2008), pp. 455–496. ISSN: 1573-1391. DOI: 10.1007/s11257-008-9051-3. URL: https://doi.org/10.1007/s11257-008-9051-3.

[9] Michael Han Daniel Han and Unsloth team. *Unsloth*. 2023. URL: http://github.com/unslothai/unsloth.

[10] DeepSeek-AI et al. *DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437.

[11] Yashar Deldjoo et al. “Recommender Systems Leveraging Multimedia Content”. In: 53.5 (Sept. 2020). ISSN: 0360-0300. DOI: 10.1145/3407190. URL: https://doi.org/10.1145/3407190.

[12] Manqing Dong et al. “A survey for trust-aware recommender systems: A deep learning perspective”. In: *Knowledge-Based Systems* 249 (2022), p. 108954. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.108954. URL: https://www.sciencedirect.com/science/article/pii/S0950705122004622.

[13] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). May 4, 2016. URL: https://data.europa.eu/eli/reg/2016/679/oj (visited on 04/13/2023).

[14] European Parliament and Council of the European Union. *Regulation (EU) 2024/1689 of the European Parliament and of the Council*. of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations EC No 300/2008, EU No 167/2013, EU No 168/2013, EU 2018/858, EU 2018/1139 and EU 2019/2144 and Directives 2014/90/EU, EU 2016/797 and EU 2020/1828. June 13, 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj (visited on 08/12/2025).

[15] Yingqiang Ge et al. "A Survey on Trustworthy Recommender Systems". In: *ACM Trans. Recomm. Syst.* 3.2 (Nov. 2024). DOI: 10.1145/3652891. URL: https://doi.org/10.1145/3652891.

[16] Azin Ghazimatin et al. "PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. WSDM '20. Houston, TX, USA: Association for Computing Machinery, 2020, pp. 196–204. ISBN: 9781450368223. DOI: 10.1145/3336191.3371824. URL: https://doi.org/10.1145/3336191.3371824.

[17] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context". In: *ACM Trans. Interact. Intell. Syst.* 5.4 (Dec. 2015). ISSN: 2160-6455. DOI: 10.1145/2827872. URL: https://doi.org/10.1145/2827872.

[18] Ming He et al. "CETD: Counterfactual Explanations by Considering Temporal Dependencies in Sequential Recommendation". In: *Applied Sciences* 13.20 (2023). ISSN: 2076-3417. DOI: 10.3390/app132011176. URL: https://www.mdpi.com/2076-3417/13/20/11176.

[19] Xiangnan He et al. "Neural Collaborative Filtering". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. ISBN: 9781450349130. DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569.

[20] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. "Explaining collaborative filtering recommendations". In: *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. CSCW '00. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 2000, pp. 241–250. ISBN: 1581132220. DOI: 10.1145/358916.358995. URL: https://doi.org/10.1145/358916.358995.

[21] Yupeng Hou et al. *Bridging Language and Items for Retrieval and Recommendation*. 2024. arXiv: 2403.03952 [cs.IR]. URL: https://arxiv.org/abs/2403.03952.

[22] Neham Jain, Vibhhu Sharma, and Gaurav Sinha. "Counterfactual Explanations for Visual Recommender Systems". In: *Companion Proceedings of the ACM Web Conference 2024*. WWW '24. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 674–677. ISBN: 9798400701726. DOI: 10.1145/3589335.3651484. URL: https://doi.org/10.1145/3589335.3651484.

[23] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Trans. Inf. Syst.* 20.4 (Oct. 2002), pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418. URL: https://doi.org/10.1145/582415.582418.

[24] Jianchao Ji et al. "GenRec: Large Language Model fornbsp;Generative Recommendation". In: *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*. Glasgow, United Kingdom: Springer-Verlag, 2024, pp. 494–502. ISBN: 978-3-031-56062-0. DOI: 10.1007/978-3-031-56063-7_42. URL: https://doi.org/10.1007/978-3-031-56063-7_42.

[25] Ray Jiang et al. "Degenerate Feedback Loops in Recommender Systems". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '19. Honolulu, HI, USA: Association for Computing Machinery, 2019, pp. 383–390. ISBN: 9781450363242. DOI: 10.1145/3306618.3314288. URL: https://doi.org/10.1145/3306618.3314288.

[26] Vassilis Kaffes, Dimitris Sacharidis, and Giorgos Giannopoulos. “Model-Agnostic Counterfactual Explanations of Recommendations”. In: *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*. UMAP ’21. Utrecht, Netherlands: Association for Computing Machinery, 2021, pp. 280–285. ISBN: 9781450383660. DOI: 10.1145/3450613.3456846. URL: https://doi.org/10.1145/3450613.3456846.

[27] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. URL: https://arxiv.org/abs/2001.08361.

[28] Anastasiia Klimashevskaia et al. “A survey on popularity bias in recommender systems”. In: *User Modeling and User-Adapted Interaction* 34.5 (July 2024), pp. 1777–1834. ISSN: 0924-1868. DOI: 10.1007/s11257-024-09406-0. URL: https://doi.org/10.1007/s11257-024-09406-0.

[29] Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1885–1894.

[30] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263.

[31] Pigi Kouki et al. “Personalized explanations for hybrid recommender systems”. In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI ’19. Marina del Ray, California: Association for Computing Machinery, 2019, pp. 379–390. ISBN: 9781450362726. DOI: 10.1145/3301275.3302306. URL: https://doi.org/10.1145/3301275.3302306.

[32] Sean Lee et al. *Open Source Strikes Bread - New Fluffy Embedding Model*. 2024. URL: https://www.mixedbread.ai/blog/mxbai-embed-large-v1.

[33] Jan Malte Lichtenberg, Alexander Buchholz, and Pola Schwöbel. *Large Language Models as Recommender Systems: A Study of Popularity Bias*. 2024. arXiv: 2406.01285 [cs.IR]. URL: https://arxiv.org/abs/2406.01285.

[34] Qiyao Ma, Xubin Ren, and Chao Huang. *XRec: Large Language Models for Explainable Recommendation*. 2024. arXiv: 2406.02377 [cs.IR]. URL: https://arxiv.org/abs/2406.02377.

[35] Ahtsham Manzoor et al. “ChatGPT as a Conversational Recommender System: A User-Centric Analysis”. In: *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. UMAP ’24. Cagliari, Italy: Association for Computing Machinery, 2024, pp. 267–272. ISBN: 9798400704338. DOI: 10.1145/3627043.3659574. URL: https://doi.org/10.1145/3627043.3659574.

[36] John McDermid et al. “Artificial intelligence explainability: the technical and ethical dimensions”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379 (Aug. 2021), p. 20200363. DOI: 10.1098/rsta.2020.0363.

[37] Tomás Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781.

[38] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 3rd ed. Self-published, 2025. ISBN: 978-3-911578-03-5. URL: https://christophm.github.io/interpretable-ml-book.

[39] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. “Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations”. In: *CoRR* abs/1905.07697 (2019). arXiv: 1905.07697. URL: http://arxiv.org/abs/1905.07697.

[40] Meike Nauta et al. “From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI”. In: *ACM Comput. Surv.* 55.13s (July 2023). ISSN: 0360-0300. DOI: 10.1145/3583558. URL: https://doi.org/10.1145/3583558.

[41] Xia Ning and George Karypis. “SLIM: Sparse Linear Methods for Top-N Recommender Systems”. In: *2011 IEEE 11th International Conference on Data Mining*. 2011, pp. 497–506. DOI: 10.1109/ICDM.2011.134.

[42] Caio Nóbrega and Leandro Marinho. “Towards explaining recommendations through local surrogate models”. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC ’19. Limassol, Cyprus: Association for Computing Machinery, 2019, pp. 1671–1678. ISBN: 9781450359337. DOI: 10.1145/3297280.3297443. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/3297280.3297443.

[43] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. 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)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://aclanthology.org/D19-1410/.

[44] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[45] Milad Sabouri et al. “Towards Explainable Temporal User Profiling with LLMs”. In: *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*. UMAP Adjunct ’25. Association for Computing Machinery, 2025, pp. 219–227. ISBN: 9798400713996. DOI: 10.1145/3708319.3733655. URL: https://doi.org/10.1145/3708319.3733655.

[46] Badrul Sarwar et al. “Item-based Collaborative Filtering Recommendation Algorithms”. In: *Proceedings of ACM World Wide Web Conference* 1 (Aug. 2001). DOI: 10.1145/371920.372071.

[47] J Ben Schafer et al. “Collaborative filtering recommender systems”. In: *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 291–324.

[48] J. Ben Schafer, Joseph Konstan, and John Riedl. “Recommender systems in e-commerce”. In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. EC ’99. Denver, Colorado, USA: Association for Computing Machinery, 1999, pp. 158–166. ISBN: 1581131763. DOI: 10.1145/336992.337035. URL: https://doi.org/10.1145/336992.337035.

[49] Ruoxi Shang, K. Feng, and Chirag Shah. *Why Am I Not Seeing It? Understanding Users’ Needs for Counterfactual Explanations in Everyday Recommendations*. June 2022. DOI: 10.1145/3531146.3533189.

[50] Eyal Shulman and Lior Wolf. “Meta Decision Trees for Explainable Recommendation Systems”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 365–371. ISBN: 9781450371100. DOI: 10.1145/3375627.3375876. URL: https://doi.org/10.1145/3375627.3375876.

[51] Timo Speith. *A Review of Taxonomies of Explainable Artificial Intelligence (XAI) Methods*. June 2022. DOI: 10.1145/3531146.3534639.

[52] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[53] Annie Surla. *How to get better outputs from your large language model*. Nov. 2023. URL: https://developer.nvidia.com/blog/how-to-get-better-outputs-from-your-large-language-model/.

[54] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV]. URL: https://arxiv.org/abs/1312.6199.

[55] Juntao Tan et al. “Counterfactual Explainable Recommendation”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM ’21. Virtual Event, Queensland, Australia: Association for Computing Machinery, 2021, pp. 1784–1793. ISBN: 9781450384469. DOI: 10.1145/3459637.3482420. URL: https://doi.org/10.1145/3459637.3482420.

[56] Juntao Tan et al. “Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 1018–1027. ISBN: 9781450390965. DOI: 10.1145/3485447.3511948. URL: https://doi.org/10.1145/3485447.3511948.

[57] Gemma Team et al. *Gemma 3 Technical Report*. 2025. arXiv: 2503.19786 [cs.CL]. URL: https://arxiv.org/abs/2503.19786.

[58] Nava Tintarev and Judith Masthoff. “Effective explanations of recommendations: user-centered design”. In: *Proceedings of the 2007 ACM Conference on Recommender Systems*. RecSys ’07. Minneapolis, MN, USA: Association for Computing Machinery, 2007, pp. 153–156. ISBN: 9781595937308. DOI: 10.1145/1297231.1297259. URL: https://doi.org/10.1145/1297231.1297259.

[59] Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. “Counterfactual Explanations for Neural Recommenders”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. ACM, July 2021, pp. 1627–1631. DOI: 10.1145/3404835.3463005. URL: http://dx.doi.org/10.1145/3404835.3463005.

[60] Robin Van Meteren and Maarten Van Someren. “Using content-based filtering for recommendation”. In: *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*. Vol. 30. Barcelona. 2000, pp. 47–56.

[61] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[62] Sandra Wachter, Brent Mittelstadt, and Chris Russell. *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*. 2018. arXiv: 1711.00399 [cs.AI]. URL: https://arxiv.org/abs/1711.00399.

[63] David S. Watson et al. “Local Explanations via Necessity and Sufficiency: Unifying Theory and Practice”. In: *Minds Mach.* 32.1 (Mar. 2022), pp. 185–218. ISSN: 0924-6495. DOI: 10.1007/s11023-022-09598-7. URL: https://doi.org/10.1007/s11023-022-09598-7.

[64] Bingbing Wen et al. “ExpScore: Learning Metrics for Recommendation Explanation”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 3740–3744. ISBN: 9781450390965. DOI: 10.1145/3485447.3512269. URL: https://doi.org/10.1145/3485447.3512269.

[65] Cihang Xie et al. “Adversarial Examples Improve Image Recognition”. In: *CoRR* abs/1911.09665 (2019). arXiv: 1911.09665. URL: http://arxiv.org/abs/1911.09665.

[66] An Yang et al. *Qwen3 Technical Report*. 2025. arXiv: 2505.09388 [cs.CL]. URL: https://arxiv.org/abs/2505.09388.

[67] Yuanshun Yao, Chong Wang, and Hang Li. *Counterfactually Evaluating Explanations in Recommender Systems*. 2022. arXiv: 2203.01310 [cs.AI]. URL: https://arxiv.org/abs/2203.01310.

[68] André Levi Zanon, Marcelo Garcia Manzato, and Leonardo Rocha. *Can Offline Metrics Measure Explanation Goals? A Comparative Survey Analysis of Offline Explanation Metrics in Recommender Systems*. 2025. arXiv: 2310.14379 [cs.IR]. URL: https://arxiv.org/abs/2310.14379.

[69] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *Found. Trends Inf. Retr.* 14.1 (Mar. 2020), pp. 1–101. ISSN: 1554-0669. DOI: 10.1561/1500000066. URL: https://doi.org/10.1561/1500000066.

[70] Yue Zhang et al. “Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models”. In: *Computational Linguistics* (Sept. 2025), pp. 1–46. ISSN: 0891-2017. DOI: 10.1162/COLI.a.16. eprint: https://direct.mit.edu/coli/article-pdf/doi/10.1162/COLI.a.16/2535477/coli.a.16.pdf. URL: https://doi.org/10.1162/COLI.a.16.