Degree Project in Information and Communication Technology

Second cycle, 30 credits

Decoupling Popularity Bias and User Fairness in LLM-Based Recommendation Systems

A prompt-engineering approach to achieve accurate, exposure-balanced, and demographically fair recommendations

MUHAMMAD HAMAD

Decoupling Popularity Bias and User Fairness in LLM-Based Recommendation Systems

A prompt-engineering approach to achieve accurate, exposure-balanced, and demographically fair recommendations

MUHAMMAD HAMAD

Master's Programme, ICT Innovation, 120 credits
Date: July 2, 2025

Supervisor: Shirin Tahmasebinotarki
Examiner: Amir H. Payberah
School of Electrical Engineering and Computer Science
Host company: HOLVI Payment Services, Helsinki Finland
Swedish title: Att frikoppla popularitetsbias och användarrättvisa i LLM-baserade rekommendationssystem
Swedish subtitle: En snabb och effektiv metod för att uppnå korrekta, långsiktiga och demografiskt rättvisa rekommendationer

Abstract

Large Language Models (LLMs) are rapidly being adopted as "plug-and-play" recommenders that require no task-specific training, although their recommendations can still face two long-standing problems: popularity bias (overexposing blockbusters) and consumer unfairness (unequal treatment of users who differ only in sensitive attributes). This thesis investigates whether these problems can be decoupled and simultaneously mitigated purely through prompt engineering, with no access to model weights.

Working with the MovieLens-1M corpus, we generate 434,880 prompts that vary three dimensions: how a user's historical tastes are sampled (top-rated, most recent, or a newly proposed 'polarized' mix of likes and dislikes), whether sensitive attributes are disclosed (neutral versus gender–age, occupation, or all), and which popularity debiaser has been applied (from a hard 'exclude-popular' order to a gentle "temporal-diverse" request). We evaluate every prompt with a triad of metrics: Hit-Rate for accuracy, log-popularity difference (LPD) for popularity bias, and Jaccard similarity for the stability of recommendations when sensitive attributes are toggled on or off.

The results reveal four insights. First, supplying the LLM with a rich, polarized taste signal increases accuracy by 42%. Second, temporal diversity reduces popularity bias by 0.6 log-units while incurring only a 1% loss in accuracy, whereas hard "exclude-popular" filters decrease accuracy by up to 65%. Third, popularity bias and user fairness are orthogonal; once popularity is neutralized, adding even minimal demographic information still halves list overlap, confirming that the two dimensions must be audited separately. Finally, only one configuration, polarized sampling strategy, temporal-diverse debiaser, and attribute-neutral prompt, simultaneously satisfies strict thresholds on accuracy (HR$\approx$0.85), popularity bias ($|\text{LPD}|<0.8$), and fairness (Jaccard$\approx$0.41).

These results show that lowering popularity bias alone does not guarantee fairness, underscoring the need to handle each bias independently. These findings establish prompt engineering as a lightweight yet powerful lever for balancing accuracy, long-tail exposure, and demographic fairness in LLM-driven recommender systems without model retraining. Beyond empirical insights, the thesis contributes a rigorous evaluation framework and practical guidelines to build fair, bias-aware recommendation systems with large language models.

Keywords

Large Language Models, Recommender Systems, Popularity Bias, Algorithmic Fairness, Prompt Engineering

Sammanfattning

Stora språkmodeller antas snabbt som ”plug-and-play”-rekommendationer som inte kräver någon uppgiftsspecifik utbildning, även om deras rekommendationer fortfarande kan förvärra två långvariga problem: popularitetsbias (överexponering av storfilmer) och orättvisa bland konsumenter (ojämlik behandling av användare som bara skiljer sig åt i skyddade attribut). Denna avhandling undersöker om dessa problem kan frikopplas och samtidigt mildras enbart genom prompt engineering, utan tillgång till modellvikter.

Med hjälp av MovieLens-1M-korpusen genererar vi 434 880 prompts som varierar i tre dimensioner: hur en användares historiska smak samplas (topprankad, senaste eller en nyligen föreslagen ”polariserad” blandning av gilla- och ogilla-attribut), om känsliga attribut avslöjas (neutral kontra kön-ålder, yrke eller alla), och vilken popularitetsdebiaser som har tillämpats (från en hård ”exkludera-populär” ordning till en mild ”tidsmässig-divers” begäran för filmer hämtade från flera epoker). Vi utvärderar varje prompt med en triad av mätvärden: Hit-Rate för noggrannhet, log-popularitetsskillnad (LPD) för popularitetsbias och Jaccard-likhet för rekommendationernas stabilitet när känsliga attribut är aktiverade eller avaktiverade. Resultaten avslöjar fyra insikter. För det första ökar noggrannheten med 42 % om LLM förses med en rik, polariserad smaksignal. För det andra minskar tidsmässig mångfald popularitetsbias med 0,6 log-enheter samtidigt som det bara medför en förlust på 1 % i noggrannhet, medan hårda ”exkludera populära” filter minskar noggrannheten med upp till 65 %. För det tredje är popularitetsbias och användarrättvisa ortogonala; när popularitet är neutraliserad, halverar även minimal demografisk information listan, vilket bekräftar att de två dimensionerna måste granskas separat. Slutligen uppfyller endast en konfiguration, polariserad samplingsstrategi, tidsmässigt diversifierad debiaser och attributneutral prompt, samtidigt strikta trösklar för noggrannhet (HR≈0,85), popularitetsbias (|LPD|<0,8) och rättvisa (Jaccard≈0,41). Dessa resultat visar att enbart minskad popularitetsbias inte garanterar rättvisa, vilket understryker behovet av att hantera varje bias separat. Dessa fynd etablerar prompt engineering som en lätt men kraftfull hävstång för att balansera noggrannhet, exponering med lång svans och demografisk rättvisa i LLM-drivna rekommendationssystem utan modellomskolning. Utöver empiriska insikter bidrar avhandlingen med ett rigoröst utvärderingsramverk och praktiska riktlinjer för att bygga rättvisa, biasmedvetna rekommendationssystem med stora språkmodeller.

Nyckelord

Stora Språkmodeller, Rekommendationssystem, Popularitetsbias, Algoritmisk Rättvisa, Promptkonstruktion

Acknowledgments

In the name of Allah, the Most Gracious, the Most Merciful.

This journey could not have begun—let alone reached its destination—without the unwavering prayers of my family in Pakistan. Leaving my parents, my beloved wife, and our two beautiful daughters was the hardest decision of my life, and every success I have tasted here is a reflection of their strength. My father, whose faith in education is matched only by his generosity, paid my very first tuition fee and planted the seed that has now borne fruit. My daughters whisper a simple prayer each night—"O God, make our father successful"—and, by His grace, that prayer has carried me through every moment of doubt.

I am deeply grateful to the EIT Digital Master School for granting me the opportunity to pursue its double-degree programme at KTH Royal Institute of Technology, Sweden, and Aalto University, Finland. The academic rigour, exposure to diverse cultures, and the many events hosted across Europe have broadened my horizons far beyond the lecture room.

I wish to acknowledge Professor Amir H. Payberah, my examiner, for organising regular progress meetings, asking the incisive questions that kept me on course, and demonstrating scholarly integrity in action. My heartfelt gratitude also goes to my supervisor, Dr Shirin Tahmasebinotarki, whose technical insight, constructive critique, and generous sharing of resources transformed vague ideas into a concrete methodology.

My warmest thanks go to my mentor, Mr Ronald Clark, whose calm advice and clear vision guided my every professional decision. His mentorship has been an anchor throughout this marathon.

To my friends and colleagues in Stockholm, Helsinki, and beyond—thank you for the late-night debugging sessions, and the sense of community you brought into my life far from home.

Finally, to all those whose names may not appear on these pages but whose kindness and prayers have shaped this work, please accept my sincere thanks. May Allah reward you abundantly.

Stockholm, July 2025
Muhammad Hamad

Contents

List of Figures

List of Tables

Chapter 1

Introduction

On today's digital platforms, what people read, watch, or buy is often decided more by algorithms than by human editors. Recommender systems now play a major role in shaping what content gets seen and what remains hidden. A single algorithmic choice can bring a little-known item into the spotlight or bury it under a wave of popular content. As Large Language Models evolve beyond simple chat tools into more general-purpose recommendation systems, they offer more personalized and interactive suggestions. But they also carry forward—and sometimes worsen—the biases seen in earlier systems. This thesis explores how these models direct users' attention, and whether that attention is fairly shared between users and lesser-known content. Understanding and improving this process is a key challenge in building more balanced and inclusive recommendation systems.

1.1 Background

Recommender systems have become one of the primary channels through which people discover movies, music, products, and information online. These systems greatly influence what content users see, raising concerns about the fairness and biases in their recommendations. One well-documented issue is popularity bias – the tendency for recommendation algorithms to disproportionately favor popular items while overlooking less popular ones. This bias can be self-reinforcing and unfair, as it amplifies the visibility of already popular content and marginalizes niche items and their creators. It also risks harming consumers whose tastes lie in the long tail, as they receive fewer relevant recommendations aligned with their unique preferences.

In recent years, the advent of artificial intelligence has introduced powerful

Large Language Models into the recommendation landscape. LLMs like GPT-3 and ChatGPT are pretrained on vast text corpora and exhibit capabilities such as understanding context, reasoning with knowledge, and generating coherent text [1]. This has led to the emergence of a new paradigm termed Recommendation via LLM (RecLLM), in which an LLM is used to generate or aid recommendations based on user input. The integration of LLMs into recommender system architectures is promising because these models come with extensive world knowledge and the ability to generalize. For example, an LLM-based system can potentially reason that "users who enjoyed a particular novel might also appreciate a film adaptation of that novel," leveraging connections that may be missed by narrow collaborative filtering models. However, incorporating LLMs also raises critical questions. Researchers have noted concerns that using an LLM as a recommender could exacerbate popularity bias, since popular entities are more likely to appear in the model's training data and thus in its outputs. Conversely, the flexibility of prompt-based LLM recommendations might offer new means to mitigate such bias (for instance, by explicitly instructing the model to consider less well-known items).

Beyond item popularity, there are also worries about user-side fairness: LLMs trained on uncurated Internet text may contain latent social prejudices, which could be reflected in whom or what they recommend to different users. If a recommender system systematically favors or disfavors certain user demographics (intentionally or not), that poses a serious fairness problem. A dual research question thus arises: How do we ensure fairness in recommendations when using LLMs, and in particular, how can we alleviate popularity bias without introducing or exacerbating biases against users?

Early studies have offered interesting insights. Lichtenberg et al. compared a simple LLM-driven recommender in [2] with traditional algorithms on a movie dataset and found that the LLM's recommendations were less skewed toward popular titles. This suggests that LLMs might inherently generate more diverse, long-tail recommendations, perhaps because they can draw on a wider range of knowledge beyond the most popular catalog items. On the other hand, separate research focusing on user fairness indicates that LLM recommenders can still reproduce unwanted biases. For example, when evaluating ChatGPT's recommendation behavior, Zhang et al. observed significant disparities in [3] the recommendations given to users of different genders and age groups, even when those users had similar taste profiles. In their benchmark tests (covering attributes such as gender, age, and nationality), ChatGPT's recommendations showed measurable unfairness

toward certain demographic groups. These findings underscore that while LLM-based recommenders may help with one aspect (popularity bias), they do not automatically resolve other fairness concerns. Instead, decoupling popularity bias and user fairness becomes crucial in this context: we need to understand and address each of these issues without inadvertently worsening the other.

This thesis is situated at the intersection of these emerging concerns in recommender systems. It builds on the premise that improving long-tail (popularity) fairness and ensuring equitable treatment of users are both essential goals, and it aims to contribute insights and methods for achieving both in the realm of LLM-powered recommendations.

1.2 Problem

The core problem addressed in this thesis is the presence of popularity bias and user fairness issues in LLM-based recommendation systems, and how to decouple these two issues in order to tackle them effectively. We define popularity bias in recommendations as the phenomenon where "popular items are disproportionately recommended, overshadowing less popular but potentially relevant items". In other words, a recommendation algorithm afflicted by popularity bias will overly concentrate its suggestions on the most popular items in the catalog, leading to a lack of diversity and neglect of the long-tail. By contrast, we define user-side fairness (also known as consumer fairness) as the principle that users with different backgrounds or attributes should receive equitable recommendation quality and treatment. A fair recommender should not, for instance, systematically favor one demographic group over another in terms of recommendation relevance or diversity. In traditional recommender research, user fairness is often evaluated with respect to sensitive user attributes such as gender, age, or ethnicity. If a system's performance or outputs differ significantly across these groups, it may be deemed unfair.

While popularity bias and user fairness are distinct concepts, they can interact in subtle ways. A system biased toward popular items might implicitly disadvantage users whose taste lies outside the mainstream (a form of indirect unfairness), and efforts to enforce user fairness (e.g., making recommendations gender-neutral) might have side effects on which items are recommended. **Original problem and definition** The key research problem here is how to measure and mitigate popularity bias in an LLM-based recommender without degrading user fairness, and how to ensure

fairness across users without re-introducing popularity bias through another avenue. Decoupling these concerns means developing approaches to address each bias in isolation and understanding their interplay. Recent literature has highlighted that popularity bias can directly impact the overall fairness of recommendations, but in the context of LLM-based systems this issue remains largely unexplored. Research Question How can popularity bias be effectively mitigated in LLM-based recommendation systems, and what are the implications of this mitigation for user fairness? **Scientific and engineering issues** Addressing the above problem involves several scientific and engineering challenges. These include:

- **Representing user preferences for an LLM:** Unlike a conventional recommender that directly accesses a user's interaction history, an LLM must be prompted with a summary of the user's preferences. Determining how to encode a user profile into the prompt effectively is non-trivial. Without careful design, the LLM may not accurately understand the user's past behavior or may ignore important preference details.

- **Extremely large candidate item space:** Recommending from a catalog of thousands or millions of items poses a challenge for a generative model. An LLM might have knowledge of many items in its parameters, but directing it to choose specific relevant items out of the entire space is difficult. Wang and Lim in [4] highlight that to enable effective LLM recommendations, one often needs an external candidate generation or filtering step to narrow down the item set, before the LLM ranks or recommends from those candidates. Balancing open-ended generation with catalog constraints is thus an engineering hurdle for real-world deployment.

- **Lack of established evaluation metrics for RecLLM fairness:** Because the LLM-based recommendation paradigm differs from traditional matrix factorization or deep learning recommenders, some conventional fairness and bias metrics may not directly apply. For example, metrics assuming a fixed model or requiring retraining for counterfactual analysis are hard to use with an API-based LLM that generates recommendations on the fly. The research community has had to devise new evaluation frameworks (e.g. the FaiRLLM benchmark [5]) tailored to LLM recommenders to measure biases and fairness. This indicates a need for novel methodology to assess the fairness of RecLLM outputs reliably.

- **Balancing popularity bias mitigation with user fairness:** Perhaps the most subtle challenge is that optimizing one dimension of fairness can impact the other. Reducing popularity bias (shifting recommendations toward the long-tail) might inadvertently affect different user groups in different ways – for instance, if certain user demographics have more niche tastes, they might benefit more from long-tail recommendations than others. Conversely, procedures to equalize recommendations for different user groups could interfere with algorithms intended to promote item diversity. Recent LLM-based studies have shown that the outcomes of fairness interventions can be sensitive to how user profiles are constructed or sampled. This complexity means designers must carefully trade off between popularity fairness and user fairness objectives, ensuring that gains in one do not translate to losses in the other.

- **Control and interpretability of LLM outputs:** From an engineering perspective, LLMs are generative and stochastic, which makes it harder to ensure a specific fairness constraint compared to, say, re-ranking a list deterministically. An LLM might occasionally produce an irrelevant or even non-existent recommendation (a hallucination) if prompted inadequately. Ensuring that the model's output stays within acceptable bounds (both in content and in respecting fairness constraints) is challenging and requires iterative prompt refinement or additional filtering. In our work, we do not fine-tune the LLM itself, so our ability to control its intrinsic behavior is limited. This places greater importance on prompt engineering and post-processing to guide the model towards fair outcomes.

1.3 Purpose

1.4 Goals

The goal of this project is to develop, evaluate, and document a prompt-engineered, LLM-based recommender system that (i) achieves high predictive accuracy, (ii) significantly mitigates popularity bias, and (iii) ensures equitable treatment across sensitive user attributes. This has been divided into the following three subgoals:

1. Subgoal 1

Design and validate prompt-engineering techniques that mitigate popularity bias while preserving recommendation utility and user fairness.

2. Subgoal 2

   Advance knowledge on relationship of accuracy, popularity bias and consumer side unfairness in RecLLMs

3. Subgoal 3

   Prove competence in conceiving, executing and documenting a full ML-system study under real-world constraints.

1.5 Research Methodology

To achieve the above goals, we adopt a quantitative experimental research methodology. The study involves using an existing movie ratings dataset and a Large Language Model to simulate a recommendation scenario. For concreteness, we focus on the popular MovieLens-1M dataset (a well-known benchmark of user–movie ratings) as our source of user preference data. We interface this dataset with a state-of-the-art LLM (specifically, OpenAI's GPT-4.1 nano model) by constructing prompts that convey a given user's historical prefrences and asking the LLM for recommendations. This prompt-based approach allows the LLM to act as a zero-shot recommender – i.e. the model generates personalized recommendations without any additional training on the dataset, relying only on its pretrained knowledge and the prompt context. We draw on recent techniques explained in [4] for next-item recommendation via prompting to design effective prompts (for example, by instructing the model to consider the user's favorite genres or by explicitly requesting a list of N movie suggestions). We will evaluate two main aspects of the LLM's recommendations: (1) popularity bias, and (2) user fairness. To assess popularity bias, we will evaluate the log popularity difference of recommended items vs ground truth. To evaluate user-side fairness, we do not create artificial counter-factual (e.g., "male vs. female") clones of the same user profile. Instead, we leverage the real sensitive attributes recorded in the MovieLens-1M data for each user. For every user, we generate two types of prompts:

1. **Neutral prompt** – contains only user's preference history (e.g. polarized sample of liked and disliked films) and omits all sensitive

fields.

2. **Attribute-aware prompt** – is identical to the neutral prompt except that it explicitly inserts the actual sensitive value drawn from MovieLens (e.g., "Gender: Female", "Age group: Young", or "Occupation: Engineer").

The LLM is queried once with each type of prompt, producing two recommendation lists. Fairness is then assessed by measuring the Jaccard similarity between these two lists. High Jaccard (i.e., the two lists largely overlap) implies that the model's recommendations are insensitive to the user's sensitive attribute – an indicator of fair treatment.Low Jaccard signals that the attribute changes the items suggested, pointing to potential unfairness. Because every comparison uses the same user with and without their own attribute, this design isolates the influence of the sensitive field while keeping the preference signal and ground-truth tastes constant. The neutral list also serves as a reference for other metrics (e.g., hit-rate and popularity bias), allowing us to gauge how much each fairness intervention disturbs relevance. Finally, our methodology includes a bias mitigation experiment. Based on the initial evaluation, we will attempt to reduce popularity bias in the LLM's outputs and observe the effects. The technique used is prompt engineering for diversity: for instance, augmenting the prompt with an instruction like "Please include some lesser-known or under-appreciated movies the user might enjoy" to explicitly encourage long-tail recommendations. After applying such interventions, we will re-evaluate the recommendations, measuring both the change in popularity bias metrics (did the long-tail exposure improve?) and any change in fairness metrics (did the interventions maintain fairness between user groups, or introduce any new disparities?). All experiments are conducted in an offline setting using the historical dataset and simulated user profiles, which allows for controlled evaluation of metrics without involving live user feedback.

1.6 Delimitations

The scope of this thesis is defined by a few key delimitations:

- **Domain and Dataset:** Our experiments are limited to the movie recommendation domain (using MovieLens data). Focusing on one domain allows us to deeply analyze bias and fairness with domain-specific context, but it means the findings may not directly generalize

to other domains (such as music or e-commerce) without additional investigation.

- **Sensitive Attributes Considered:** The fairness evaluation in this thesis is deliberately limited to those user attributes that are explicitly recorded in the MovieLens-1M dataset and can therefore be injected into prompts in a verifiable, controlled manner. Four prompt variants are employed. In the neutral variant no sensitive fields are added, providing a reference list for each user. The *gender_age_only* variant appends the user's gender (M or F) and coarse-grained age group (Teen, Young, or Adult), reflecting the two demographic factors most frequently examined in recommender-system fairness studies. The *occupation_only* variant adds the user's occupation code (one of 21 predefined roles in MovieLens), offering a socio-professional dimension that is less explored yet still directly available. Finally, the all_attributes variant combines gender, age group, and occupation to give a limited, three-way intersectional test. Attributes not present in MovieLens—such as ethnicity, geographic location, or income—are excluded because reliable ground-truth values are unavailable, making any fairness claim unverifiable; likewise, richer intersectional identities beyond the *gender_age_occupation* triple fall outside the scope of this work. Constraining the study to recorded demographics ensures that all fairness comparisons are grounded in real user metadata.

- **Fairness Definitions:** We focus on group fairness at the consumer side – measuring whether different user groups (e.g., defined by gender) receive equitable recommendations. We do not extensively cover individual fairness (the idea that similar individuals should get similar recommendations) or provider-side fairness beyond popularity bias (e.g., fairness for item creators not related to popularity). While these are important, our work emphasizes the long-tail (item popularity) fairness and user group fairness as the two primary angles.

- **LLM Mode of Use:** We use the LLM in a zero-shot, prompt-driven manner only. Due to resource constraints, we do not fine-tune or retrain the LLM on the recommendation dataset. This limits our ability to adjust the model's internal parameters for fairness; instead, we rely on prompt manipulation and output processing. It also means the LLM's knowledge is fixed to its pretraining data). We acknowledge that this may cause the LLM to be unaware of very recent movies or trends, but

mitigating that would require techniques (like retrieval augmentation or model updating) which are beyond our scope.

- **Offline Evaluation:** All evaluations are conducted offline on historical data and simulated scenarios. We do not deploy a live system or collect feedback from real users. As such, dynamic effects (e.g., how user behavior might change in response to less popularity-biased recommendations, or how an unfair recommendation might affect user engagement over time) are not captured in our study.

1.7 Structure of the thesis

The remainder of this thesis is organized as follows. Chapter 2 provides background on the major topics relevant to our study, including an overview of LLM-based recommender systems, the concepts of popularity bias and fairness in recommendations, and related work in these areas. Chapter 3 and Chapter 4 describes the research methodology in detail, including the dataset, the design of the LLM-based recommendation approach, the baseline comparator, and the metrics and experimental procedures used. Chapter 5 presents the results of the experiments, focusing on the measurements of popularity bias and user fairness for both the LLM recommender and the baseline, and reporting the effects of any bias mitigation strategies. Chapter 6 discusses the implications of the findings, examining how reducing popularity bias interacts with user fairness and what trade-offs or synergies were observed, as well as situating the results in the broader context of recommender system research. Finally, Chapter 7 concludes the thesis, summarizing the contributions and offering suggestions for future work on developing fair and bias-aware LLM-driven recommender systems.

Chapter 2

Background

2.1 LLM-Based Recommender Systems

2.1.1 Large Language Models for Recommendation

Large Language Models have recently opened new possibilities for recommender systems [6]. Traditionally, recommendation algorithms have relied on patterns in user–item interaction data (e.g., matrix factorization of user ratings or graph-based similarity) to predict what a user will like. In contrast, an LLM can leverage its broad training on textual knowledge to make recommendations in a more semantic or knowledge-driven way. The concept of using LLMs as recommenders has gained interest only in the last couple of years. In an LLM-based recommender, we typically provide the model with a textual description of the recommendation context (such as a summary of the user's preferences or a query asking for suggestions), and then let the model generate item suggestions in natural language. This approach can be used in conversational recommendation scenarios (where the system and user engage in dialogue), or as a one-shot predictor that outputs a list of recommended items given a prompt. One reason LLMs are seen as promising for recommendations is their ability to incorporate external knowledge and reasoning. Because they are trained on vast swaths of internet text, LLMs have embedded knowledge about items (for example, plot descriptions of movies, popularity and cultural impact, or the content of user reviews) that standard collaborative filtering models lack. This means an LLM might know that *"The Godfather"* is a highly acclaimed crime drama film and could recommend it to someone who likes crime dramas, even without seeing a single user rating. Additionally, LLMs can perform cross-domain reasoning: an LLM could learn from a user's

movie preferences and suggest a novel or book that aligns with those tastes, leveraging its understanding of storylines and adaptations. For instance, if a user loves *The Lord of the Rings* films, an LLM might recommend the original *Tolkien books* or similar fantasy literature, thereby connecting domains (movies to books) through its general world knowledge. These capabilities hint at a more flexible recommendation engine that is not tied to a single domain or narrow user–item matrix. However, using LLMs for recommendation is not straightforward. A key difference is that LLMs generate free-form text. Ensuring that the output corresponds exactly to valid items in a catalog can be challenging. There are two general strategies for building LLM-based recommenders: (1) Generative recommendation via prompting, where the LLM is directly asked to produce a list of recommendations (possibly with some guiding context or examples), and (2) Reranking or scoring, where the LLM is given a set of candidate items (e.g., top results from a traditional recommender or search engine) and asked to refine or justify which of those best match the user's needs. The first strategy leverages the LLM's knowledge and language generation to produce novel recommendations in a zero-shot manner, while the second uses the LLM more as a knowledgeable re-ranking model that can incorporate richer criteria (like analyzing item descriptions or user reviews in a human-like way). This thesis mainly focuses on the former approach (direct generation), aligning with the recent surge in research on zero-shot LLM recommenders.

2.1.2 Prompt Engineering for Recommendation

In deploying LLMs as recommenders, prompt engineering plays a critical role [7]. Prompt engineering refers to the craft of designing the input query or instructions to the LLM in order to elicit the desired output. Because LLMs do not inherently understand that they should produce a list of item identifiers or titles for recommendations, we must guide them with an appropriate prompt. A simple example prompt might be: *"The user has recently enjoyed movies X, Y, and Z. What other movies might they like?"* The model then completes this prompt by generating some suggestions. This straightforward approach can work, but research has found that more structured prompts often yield better results. One advanced prompting strategy, introduced by Wang and Lim in [4], is called Zero-Shot Next-Item Recommendation (NIR) prompting. In their method, rather than asking the model directly for recommendations, they break the task into a few sub-steps (all handled within a single prompt). For instance, the prompt might first instruct the LLM to

summarize the user's preferences or identify the main themes from the user's liked items; next, it might ask the LLM to consider a set of candidate items (perhaps generated by an external filtering module) and pick which ones best match those preferences; finally, it asks the LLM to output a ranked list of, say, 10 recommended items. This 3-step prompting approach (preference understanding → candidate consideration → recommendation listing) was shown to guide GPT-3 in producing high-quality recommendations. Crucially, providing a bit of "chain-of-thought" structure in the prompt can help the LLM focus and avoid superficial or generic suggestions. Another important aspect of prompt engineering is giving the LLM explicit cues or constraints to mitigate bias. Since our focus includes popularity bias, one can experiment with prompt phrasing that encourages long-tail recommendations. For example, adding a line like "Try to include some lesser-known titles that the user might not have heard of" could nudge the model to retrieve more niche knowledge from its memory. This idea of prompt tuning for bias mitigation suggests that carefully crafted prompts might reduce popularity bias in LLM recommendations, see [2]. The prompt effectively becomes our tool to steer the LLM: we can emphasize diversity, specify the number of outputs, set formatting (to ensure the model lists items clearly), or include additional context (such as a brief user profile description) – all of which significantly influence the recommendations the LLM will generate.

2.1.3 Zero-Shot Recommendation Performance

The feasibility of zero-shot LLM recommendation has been affirmed by initial experiments. In the study by Wang et. al., their GPT-3 based recommender (using the NIR prompting strategy described above) was evaluated on the MovieLens 100K dataset [4] . Remarkably, even without any training on that data, the LLM's recommendations attained accuracy metrics (such as hit rate and NDCG) that were on par with, or even exceeded, those of some strong sequential recommendation models trained on the same dataset [4]. In fact, the zero-shot GPT-3 approach outperformed certain state-of-the-art sequence-based recommenders that had been fully trained on user interaction sequences. This result demonstrates the power of LLMs' pretrained knowledge and inference capabilities.

LLM-based recommender could achieve competitive recommendation quality while inherently reducing the severity of popularity bias compared to traditional models [2]. These successes have led to growing optimism that LLMs might serve as a novel kind of recommender engine, especially useful

in scenarios where little training data is available or where we want the system to generalize beyond narrow patterns. It's worth noting, however, that most evaluations so far are offline and on relatively static datasets. The true test of an LLM recommender's performance would be in a live setting, where it must handle evolving user interests, very large item catalogs in real-time, and efficiency constraints. Nonetheless, the current evidence indicates that zero-shot recommenders powered by LLMs are a viable concept, highlighting ample research opportunities for using LLMs in recommendation tasks.

2.2 Fairness and Bias in LLM-Based Recommender Systems

2.2.1 Popularity Bias in LLM Recommendations

A central question for our work is whether LLM-based recommendations suffer from the same popularity bias as traditional systems, and if so, how that bias manifests. The evidence so far is mixed but intriguing. On one hand, an LLM's training data is likely to contain a lot of information about widely known items (for instance, blockbuster movies are discussed extensively online, whereas obscure indie films are rarely mentioned) [2]. On the other hand, LLMs do not have direct access to platform-specific popularity signals – unlike a collaborative filter, an LLM isn't directly reading user interaction counts from a database; it's generating recommendations based on what it "knows" and the prompt it's given. Lichtenberg et al. (2024) tackled this issue by explicitly measuring the popularity bias of an LLM's recommendations versus a standard recommender system. They introduced a principled way to quantify how skewed toward popular items a recommendation list is, arguing that some existing metrics were not fully adequate. Using their metrics, the comparison revealed that the LLM-based recommender exhibited significantly less popularity bias than the traditional algorithms on the same data. In other words, the LLM recommended a more balanced mix of popular and less popular items, whereas the conventional collaborative filtering approach heavily favored the head (most popular items). This finding suggests that LLMs might inherently bring more diversity or novelty into their recommendations. One hypothesis is that because the LLM is generating items based on semantic associations and broad knowledge, it can surface relevant long-tail items that a collaborative filter (which learns mostly from frequency and co-occurrence patterns in the user data) might

overlook. It's also possible that, since the LLM wasn't explicitly optimized for accuracy on historical user–item interactions, it doesn't over-fit to majority preferences the way a typical model might, thereby avoiding some of the popularity-driven feedback loop. The result is potentially a fairer distribution of recommendation attention across the item catalog. Of course, more research is needed to confirm this behavior in other domains and with other LLM variants. It is not guaranteed that every LLM will behave similarly – for instance, if an LLM's knowledge cutoff means it misses very recent or less-documented items, it might inadvertently omit those from its suggestions (which could introduce a bias toward older, well-documented popular items) [8]. Nonetheless, the reduction of popularity bias observed in initial studies is an encouraging sign that LLM-based recommenders could help promote long-tail content.

2.2.2 Consumer-Side Fairness

In recommender systems, consumer-side fairness refers to the idea that the system's performance and behavior should be equitable across different groups of users. In other words, a user should not receive poorer or less useful recommendations simply because of some intrinsic attribute like their gender, age, or ethnicity, when those factors are not actually relevant to their tastes [9]. This form of fairness is analogous to fairness in classification or ranking systems: we want to avoid systematic discrimination whereby certain user groups are underserved or negatively impacted by the recommendations. Ensuring consumer fairness is important for both ethical reasons and user satisfaction – if a subset of users consistently gets lower-quality recommendations, they are being disadvantaged and may lose trust in the system. However, achieving user fairness in recommendations is complicated by the fact that recommendation algorithms learn from historical user behavior, which itself may carry societal biases. For example, if historically a certain demographic group interacted less with a platform (perhaps due to the platform's content or external factors), a collaborative filtering algorithm might inadvertently learn to give that group fewer or less diverse recommendations, thereby reinforcing a lower engagement cycle. This outcome is not a reflection of those users' actual preferences, but rather a byproduct of biased data. Traditional approaches to fairness in recommender systems have borrowed concepts from fair machine learning, introducing metrics like statistical parity (e.g., the similarity of recommendation distributions between group A and group B) or equal

opportunity (e.g., ensuring that users in different groups with similar tastes have similar chances of receiving relevant recommendations). Researchers have also proposed methods such as re-ranking algorithms that adjust or calibrate recommendation lists to improve fairness across demographics (for instance, interleaving recommendations to ensure each user group gets a proportional share of certain recommendation opportunities) [10] [11]. With the introduction of LLM-based recommenders, these fairness concerns persist and can become even more complex. LLMs may contain latent biases from their training data (which is web text and can include stereotypes or societal biases), and thus the recommendations they generate might reflect those biases if not checked. For instance, an LLM might associate certain genres or content with a particular gender due to stereotype text in its training corpus, resulting in gender-biased recommendations. Evaluating the fairness of LLM-based recommendations requires new approaches; Zhang et al. proposed the FaiRLLM benchmark for this reason [3]. In their evaluation of ChatGPT's recommender behavior, they discovered that the model's outputs were not fair with respect to several sensitive user attributes. For example, the recommendations presented to a user profile labeled as female versus male showed systematic differences that were not explainable by taste alone – indicating a gender bias in how the LLM generated suggestions. Similarly, profiles with different age labels (but identical preference histories) received different recommendations, pointing to age-based disparities. This example highlights that even a highly sophisticated model like ChatGPT can carry or introduce biases in personalized results. It is therefore imperative to audit and adjust RecLLM systems for fairness just as one would for any other recommendation algorithm. To mitigate user-side biases, a variety of techniques are being explored. One approach is counterfactual testing and prompt adjustment: essentially, querying the model in pairs of scenarios (with and without a sensitive attribute mentioned) and then adjusting the prompts or filtering the results to minimize any differences. Another approach is fine-tuning or conditioning the model using balanced data . In general, the goal is to ensure that the model's recommendations depend primarily on a user's demonstrated preferences and not on demographic attributes that have nothing to do with those preferences. In our thesis, we adopt some of these evaluation strategies (e.g., comparing recommendations for same user profiles that differ only in a demographic label) to diagnose fairness.

2.2.3 Item Side Fairness and Popularity Bias

Beyond user-centric fairness, recommender systems also face fairness questions on the item side – often framed in terms of giving fair exposure to items or content providers. Popularity bias is essentially an item-side fairness issue: if an algorithm predominantly recommends already-popular items, it creates a "winner-takes-all" dynamic where a small subset of content (and their producers) get the vast majority of attention, while other content is largely ignored [12] [13]. This is considered unfair from the perspective of content creators (especially new entrants or niche creators who don't yet have a large audience) and can also harm consumers by limiting the diversity of content they are exposed to. Recommender systems are known to be affected by popularity bias, and while this bias stems partly from human tendencies, algorithms can amplify it, resulting in unfair treatment of both end-users and content creators. In other words, the rich get richer: popular items keep getting more exposure, and users end up confined to what is already trending, potentially missing out on equally relevant but less famous items. One way to view long-tail fairness is through the lens of equal opportunity for items: ideally, every item that might be relevant to someone should have a chance to be recommended, rather than being overshadowed purely due to lack of initial popularity. Traditional algorithms, particularly those based on collaborative filtering, often exacerbate popularity bias – they learn from aggregate user behavior, which naturally skews toward popular items, thereby reinforcing that skew in recommendations [13]. Over time, this feedback loop can severely skew the recommendation ecosystem. As noted earlier, initial evidence suggests that an LLM-based recommender may naturally exhibit less popularity bias than a conventional system . This could be a significant positive: if LLMs inherently give more exposure to niche items, they might promote long-tail fairness by default, reducing the need for complex bias mitigation. Our work will explore techniques to decouple popularity influence from the LLM's recommendations – for example, by instructing the model (via the prompt) to ignore how famous an item is and focus only on relevance, we can test whether the model can be guided to be even more long-tail-friendly. The ultimate goal is aligned with the fairness objective from traditional recommender research: not letting popularity alone drive the recommendations, but rather balancing popularity with personalization and novel discovery. If LLM-based systems can achieve that balance more easily or effectively, it would be a compelling advantage of this new technology.

2.3 Challenges and Limitations of LLM-Based Recommenders

Despite their promise, LLM-based recommenders face a number of challenges and limitations.

- *Scalability* is one practical concern: running a large model for every recommendation request can be computationally expensive and slow [14]. Traditional recommender algorithms are typically designed for efficiency (e.g., using precomputed user–item matrices or fast vector computations), whereas querying an LLM involves significant computation [15]. If an LLM is hosted via an API, there are also monetary costs and latency to consider for each call.

- Moreover, the *item space in a real system is huge* – on the order of thousands or millions of items – and naively asking an LLM to choose from among all of these is infeasible. As noted earlier, Wang& Lim had to incorporate an external candidate generation step precisely because of this issue [4]. For large-scale use, a pure LLM approach likely needs to be hybridized with traditional retrieval components to handle the volume of items and return results in a timely manner [16].

- Another limitation is the *static knowledge and recency of LLMs*. These models have a fixed training cutoff [17]. They do not automatically know about new items introduced after their training. In a live recommender system, new products, songs, or movies appear frequently, and capturing the appeal of these fresh items is crucial. An LLM would not have any information about a brand-new movie release, so unless provided with that information via the prompt or an external update mechanism, it cannot recommend that movie. This could cause LLM-based recommenders to lag in adapting to trends or to favor older content that was prevalent in their training data.

- Additionally, LLMs might occasionally generate *recommendations that are not valid* or precise – for instance, mentioning a slightly incorrect title or a non-existent item (an artifact of hallucination) [18]. Ensuring that all outputs correspond to real, available items in the catalog might require a post-processing step to verify or correct the LLM's suggestions.

- *Output control* is thus a related challenge: the format and content of the LLM's recommendation output can vary. Unlike a traditional system that outputs a fixed set of item IDs, an LLM might produce a nicely formatted paragraph or a list of items with explanations. Parsing the output to extract the recommended items reliably is an extra engineering step. We have to design prompts that encourage the LLM to list items clearly (e.g., enumerating them) and possibly limit extraneous commentary, depending on how the recommendations will be consumed by the downstream system or user.

- Finally, there are *ethical and fairness considerations* inherent to LLM recommenders. Because LLMs learn from broad internet text, they may carry forward biases present in that data. For example, if certain genres or user communities are underrepresented or portrayed with bias in the training corpus, the LLM's recommendations might reflect those biases (perhaps by systematically under-recommending content by creators from those communities, or by associating certain content with certain demographic profiles). Recent work underscores that LLM-based recommendations can indeed perpetuate social biases, making fairness evaluation imperative.

In summary, while LLMs introduce a powerful new approach to recommendation, one must be mindful of these practical and ethical limitations. In many cases, it may be necessary to combine LLMs with other system components or apply special interventions (like bias mitigation prompts or result filters) to achieve a robust and fair recommender solution.

2.4 Related Work

2.4.1 LLM-Based Recommendation Systems

Several key studies have begun exploring the use of LLMs for recommendation tasks. Lichtenberg et al. were among the first to directly compare an LLM-based recommender to traditional algorithms, in a study focused on movie recommendations, see [2]. They found that a GPT-style LLM, prompted to make movie suggestions, produced recommendations with accuracy comparable to collaborative filtering and additionally showed lower popularity bias in its outputs. Another foundational work by Wang and Lim demonstrated the viability of zero-shot recommendation using GPT-3, see [4]. By carefully prompting GPT-3 to assume the role of a recommender,

they got the model to generate next-item recommendations that rivaled, and in some cases outperformed, conventional recommendation models on standard benchmarks . These studies provided early evidence that large pre-trained language models can serve as effective recommenders even without domain-specific fine-tuning. Beyond these individual studies, the idea of LLM-driven recommendation has garnered significant interest. The emerging consensus is that LLMs bring new capabilities to recommender systems – such as understanding contextual information or item content in a deep way, and potentially addressing issues like cross-domain recommendations and cold-start users – but they also introduce new challenges (like including scalability and output control). Our thesis fits into this growing body of work by focusing specifically on the bias and fairness implications of LLM recommenders, which is a facet only briefly noted by early studies but not yet deeply investigated.

2.4.2 Fairness in LLM-Based Recommendation

On the fairness front, research is just beginning to assess how LLM-based recommenders behave. Zhang et al. took an early look at this issue with the question *"Is ChatGPT fair for recommendation?"*. They developed a benchmark called FaiRLLM in [3] to evaluate ChatGPT's recommendations for fairness across multiple user attributes. As discussed, their findings revealed that ChatGPT's zero-shot recommendations still exhibited noticeable biases – certain user groups (defined by attributes like gender or age) received systematically different or lower-quality recommendations. This work underscored that deploying LLMs in recommenders does not automatically resolve fairness issues and that new evaluation tools are needed to identify and quantify these biases. Deldjoo and Di Noia in [5] extend Zhang et al.'s FaiRLLM benchmark with CFaiRLLM, highlighting a central limitation of the earlier work. FaiRLLM diagnoses consumer unfairness by comparing the overlap between two recommendation lists—one generated with, and one without, an explicit sensitive attribute in the prompt. A large drop in overlap is taken as evidence of bias. CFaiRLLM argues that this criterion can misclassify legitimate personalisation as unfairness: if revealing a user's gender or age simply helps the model tailor results more closely to true taste, overlap will decrease even though the system behaves appropriately. To disentangle bias from valid personalisation, CFaiRLLM introduces a true-preference-alignment metric that checks whether the attribute-conditioned recommendations better match the user's demonstrated interests (hit and rank

agreement with held-out ratings) rather than merely counting shared titles. The framework also stresses that fairness findings are highly sensitive to how the user profile delivered to the LLM is built—random, recent, or top-rated samples each yield different alignment gaps—and that intersectional prompts (e.g., "*Teenage female*") often expose larger disparities than single attributes alone. Thus CFaiRLLM reframes RecLLM fairness as a balance between equal treatment and faithful personalisation, offering a more nuanced lens than FaiRLLM's list-overlap heuristic. On the mitigation side, Hua et al. introduced a method called Counterfactually-Fair Prompting (CFP) in [19] as part of an approach to create an Unbiased Foundation Model for Fairness-aware Recommendation (UP5) in [**?**]. The core idea is to adjust the prompting of the LLM in a way that enforces a form of counterfactual fairness – for example, ensuring that for a given user, prompts that differ only in a sensitive descriptor yield similar recommendations. In their experiments on MovieLens and another dataset, they showed that the CFP technique can substantially reduce recommendation disparities between user groups while maintaining strong recommendation performance. In fact, their LLM-based approach with fairness-aware prompting outperformed several traditional fairness-aware recommender models on both accuracy and fairness metrics. This is an encouraging sign that the flexibility of LLMs can be harnessed to achieve fairness goals (potentially more easily than retraining an entire algorithm from scratch). In summary, these related works indicate two key points: (1) LLM-based recommenders do exhibit bias and fairness issues that need careful evaluation, and (2) there are emerging techniques to mitigate such biases, especially using clever prompt designs or slight fine-tuning, which show promise. Our thesis builds upon this foundation by examining popularity bias and user fairness together in an LLM recommender. Whereas prior works have largely examined either popularity bias [3] or user fairness [3] [5] in isolation, we aim to assess them in tandem and explore how a strategy that improves one aspect might affect the other. This integrated perspective is important for developing LLM recommenders that are both fair to users and fair to content.

2.4.3 Fairness and Popularity Bias in Traditional Recommenders

For context, it is useful to consider how the recommender systems community has traditionally dealt with popularity bias and fairness before the advent of LLMs. Popularity bias has long been recognized as a source of

unfairness in recommender systems, particularly in how it can skew outcomes for both users and item providers. Over the years, researchers have explored many interventions to counteract popularity bias and improve long-tail fairness. Some approaches involve modifying the recommendation algorithm's objective function – for example, adding a term that penalizes recommending too-popular items, thus encouraging more diverse, long-tail recommendations. Other approaches operate at the ranking stage, such as re-ranking the list of recommendations to inject a certain percentage of less-popular items [20]. The idea is often to balance accuracy with coverage: ensuring that users still get relevant suggestions, but from a broader palette of items than just the blockbusters. This line of work often falls under the umbrella of multi-stakeholder fairness or aggregate diversity, where the aim is to ensure that no stakeholder (neither users with niche tastes nor providers of niche content) is systematically left behind by the algorithm. Carnovalini et al. provide a comprehensive survey of these efforts in their review of popularity bias and fairness in the long tail, see [21]. They emphasize that while popularity bias partly originates from human behavior (users do tend to gravitate toward popular options), recommender algorithms amplify this effect and thus bear responsibility for mitigating it. The survey discusses numerous bias mitigation strategies and notes that each has its trade-offs – many methods can reduce bias to some extent, but often at the cost of a small drop in accuracy or requiring careful tuning to avoid hurting the user experience. The analysis highlighted the need for a multi-disciplinary approach to this problem, examining not just the algorithms but also how users respond to different recommendation diversity levels, and considering platform objectives (for instance, platforms sometimes prioritize popularity for business reasons). Our work is informed by this rich body of traditional research. We aim to bring some of those insights into the LLM era – for example, evaluating whether an LLM-based recommender naturally provides some of the long-tail diversity that earlier systems needed explicit techniques to achieve, and ensuring that the metrics we use align with those historically used so we can compare outcomes. Ultimately, the goal of a fair recommender remains the same as in past work: to serve users well without unduly favoring a small set of popular items or a particular user group. By examining an LLM-based approach through the lens of both popularity bias mitigation and user fairness, we hope to identify whether these new models help alleviate long-standing issues, or whether they require similar (or new) corrective measures to meet fairness objectives.

2.4.4 Bias in Language Model–Based Information Access

Finally, it is worth noting that biases introduced by large language models have been observed beyond recommender systems, in related information access domains like search. For example, in information retrieval, recent work has shown that using LLM-derived text embeddings can bias results towards certain writing styles. Cao found in [22] that state-of-the-art embedding models tended to prefer documents written in a formal, sophisticated style, while more informal or emotive writing styles were less favored by these models. In terms of query matching, many embedding-based retrieval systems would match the style of the query with the style of retrieved documents; some models even displayed a consistent preference for content that resembled the style of text they were trained on, which was often Wikipedia-like or academic in tone. This writing style bias means that certain voices or communities—those that use a different dialect, slang, or emotional tone—could be systematically down-ranked, inadvertently silencing or marginalizing those content creators. Such a bias poses a significant threat to fairness in information retrieval, as it goes beyond relevance and starts filtering content based on stylistic attributes. The relevance of this example to our work is that it reinforces a general lesson: every time we integrate a complex AI model (like an LLM) into an information access system, we must be vigilant for unexpected biases. Whether it's a search engine or a recommender system, the model may introduce new bias patterns or amplify subtle ones that a simpler algorithm might not. In the case of LLM recommenders, we are mindful that biases might manifest in ways we have not traditionally measured—perhaps in the phrasing of recommendations, or in which contextual details the model focuses on for different users. By learning from findings in IR and NLP, we approach our LLM-based recommender with caution, actively checking for biases (popularity-based, demographic, stylistic, or otherwise) rather than assuming the model will be fair just because it is more "intelligent." This interdisciplinary awareness helps ensure we cast a wide net in our evaluation and mitigation strategies.

2.5 Summary

In summary, this chapter surveyed the context and literature around using LLMs for recommendation and the twin concerns of popularity bias and fairness. We reviewed how LLM-based recommender systems work and

the novel capabilities they bring, as well as their limitations. Critically, we discussed how popularity bias remains a challenge – tending to surface the same popular items repeatedly – and how this bias relates to fairness for item providers in the long tail. We also examined user-side fairness, especially how LLM-generated recommendations might reflect or even amplify social biases, requiring new evaluation benchmarks and mitigation techniques. Initial research has shown that LLM recommenders can achieve strong performance and may even reduce popularity bias compared to traditional systems, but they are not immune to fairness issues and must be carefully audited. Our review of related work highlighted that these issues have been studied separately in early LLM recommender research: some work focused on popularity bias, while other work evaluated demographic fairness. There is a clear gap when it comes to studying them together. This thesis addresses that gap by evaluating an LLM-based recommendation approach on both fronts simultaneously. In the next chapter, we will outline the methodology for our experiments, including how we measure popularity bias and fairness, and how we will attempt to decouple these factors to understand and improve the recommender's performance on both dimensions. The insights from this background will directly inform our experimental design and the interpretation of our results.

Chapter 3

Method or Methods

3.1 Research Process

This research follows a quantitative experimental approach to assess how LLM-based recommendations can be tuned to balance accuracy, popularity bias, and user fairness. We systematically varied the prompt inputs to a large language model recommender along multiple dimensions and observed the effects on recommendation outcomes. The study was conducted as an offline simulation using a public movie rating dataset(Movielens 1M) and a pre-trained LLM(ChatGPT 4.1-nano) as the recommendation engine. We first defined the problem of popularity bias – where recommenders over-focus on popular items at the expense of niche items – and recognized its unfair impact on users with niche interests.

Building on prior discussion in [6] that LLMs integrated into recommender systems might exacerbate or alleviate such bias via prompt design, we designed an experiment to probe prompt-based interventions. The overall process involved: (1) constructing diverse user profiles from the dataset, (2) generating tailored recommendation prompts with various fairness and bias-mitigation cues, (3) querying the LLM for recommendations, and (4) evaluating the results with respect to accuracy, popularity bias, and fairness. Figure 3.1 shows the complete workflow. This process was iterative and exploratory – initial pilot prompts were tested, and insights from those trials (e.g. feasibility of parsing outputs, influence of prompt phrasing) informed refinements in the final prompt templates and experimental conditions. By iterating through this cycle, we ensured that the methodology remained grounded in the LLM's actual behavior while addressing the research questions.

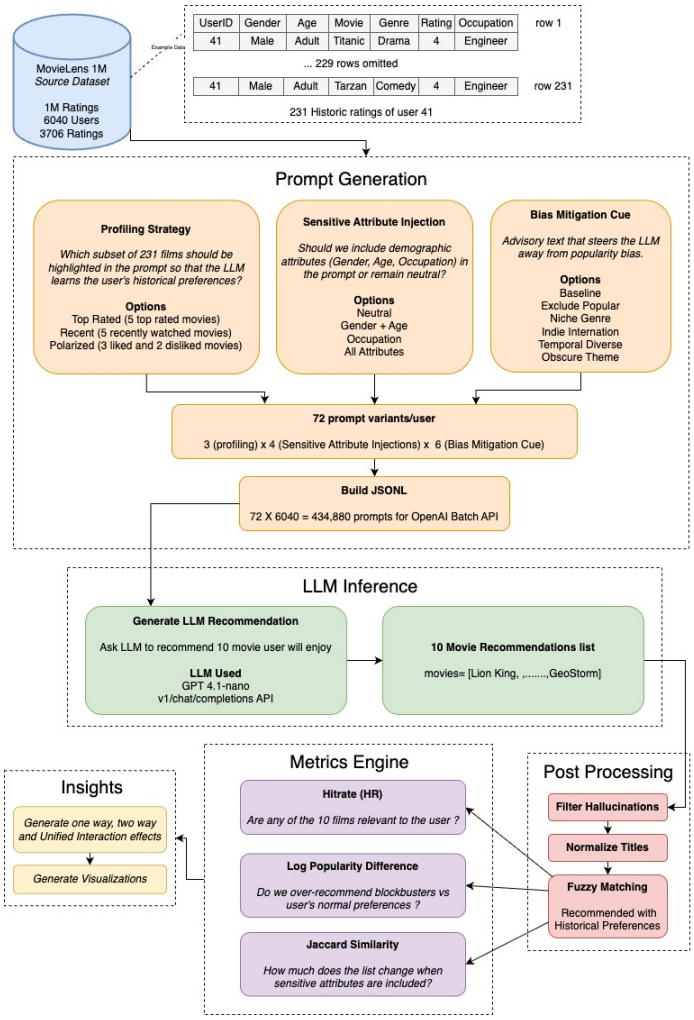

Figure 3.1: Research Process

3.2 Research Paradigm

We treat the LLM-based recommender as a computational artifact whose behavior under different configurations can be measured. The study emphasizes user-side fairness, meaning we examine how consistent the recommender's performance and outputs are across users with different attributes. Each user's data is run through every prompt variant, allowing direct comparisons of outcomes with and without fairness prompts for the same individual. We acknowledge that LLMs come with inherent sociotechnical unfairness from training data. Thus, our paradigm also incorporates a bias-awareness

perspective: rather than assuming the recommender is neutral, we actively probe it for fairness behavior by toggling sensitive inputs. The paradigm combines exploratory analysis (to discover how the LLM responds to prompts) with confirmatory evaluation (to test specific hypotheses about bias mitigation and fairness). Overall, this approach provides a structured yet flexible way to investigate if and how prompt engineering can decouple popularity bias from user fairness in LLM recommendations.

3.3 Data Collection

3.3.1 Dataset

We utilized the MovieLens dataset (1M version) as the source of user preference data. This dataset contains *6,040* users and *3706* movies, with each user having rated a variety of films. The total ratings that exist are equal to 1 million. It provides not only user–item ratings but also demographic attributes for users (e.g. *age, gender, and occupation*) which are relevant for fairness analysis. The target population thus consists of movie viewers with diverse profiles, and the sampling approach was comprehensive – we included all users from the dataset (no sampling exclusion) to maximize coverage of different user types. Each user's full rating history was considered as their pool of past interactions. From these histories, we constructed profile samples to feed into the LLM. Specifically, for each user we extracted a small set of representative movies to form a profile summary. Depending on the experimental condition, this profile was either the user's top-rated movies, most recent movies, or a mix of polarized preferences. This profiling step ensured that the LLM was given a concise yet informative snapshot of the user's taste.

The MovieLens data provided each user's self-reported gender, age (in years), and occupation category (21 in total). We leveraged these attributes to create variations of prompts that include or exclude sensitive information. The inclusion of these attributes allowed us to simulate scenarios where the recommender is aware of a user's demographics versus scenarios where it is "blind" to them. By doing so, we collected data on how recommendations might differ when a user is presented, for example, as a "*25-year-old male engineer*" versus just a generic user. This approach aligns with fairness testing methodologies that examine recommendation differences across user groups. All demographic information used was drawn directly from the dataset (no additional user data collection was needed), and we took care to handle it ethically – only using it to craft prompt contexts, not for any discriminatory

decision-making.

3.3.2 LLM Outputs

Data collection also encompassed the recommendations generated by the LLM. For each user-profile and prompt variant, we recorded the top-10 movies recommended by the LLM. In total, we constructed *434,880* distinct prompts (*72* variants for each of *6,040* users) and collected the LLM's outputs for each. Each output was captured in a structured format (JSON) for subsequent analysis. Alongside the movie titles, we also stored metadata such as the prompt variant used in the form of *custom_id*, and the *timestamp* of the query. This yielded a rich dataset of LLM-driven recommendations. To facilitate evaluation, each recommended title was mapped to the corresponding movie in the MovieLens database (or flagged if it did not match any known movie). We employed fuzzy matching (discussed later) for this mapping to ensure that minor discrepancies in titles (e.g. punctuation, year inclusion) did not prevent us from identifying the intended movie. The result of the data collection phase was a comprehensive table of (User, *ProfileType*, *FairnessContext*, *BiasMitigation*, *RecommendedMovies*), which forms the basis for our analysis.

3.4 Experimental design and Planned Measurements

Our experiment was designed as a fully crossed factorial evaluation, manipulating three independent variables in the prompt generation pipeline: (a) User Profile Strategy, (b) Fairness Context Injection, (c) Bias Mitigation Instruction. Each combination of these factors constitutes a unique prompt variant. The prompt generation pipeline is illustrated conceptually and proceeds in stages:

1. **Profile Construction:** Given a target user, we generate a brief profile description. We implemented three strategies for this. In the Top-Rated profile condition, we identify the top *5* movies that the user rated highest (indicating strong likes) and include those titles. In the Recent profile condition, we take the most recent *5* movies the user has watched/rated (based on timestamps) to reflect current interests. In the Polarized profile condition, we include a mix of movies the user loved and movies they rated very low, creating a deliberately contrasting set of likes and

dislikes. The profile is then embedded into a prompt sentence (e.g. "*The user enjoyed Movie A, Movie B, ...*").

2. **Fairness Context Injection:**Next, we optionally prepend a user description highlighting demographic attributes. We defined four variants: neutral (no personal attributes), *gender_age_only*, *occupation_only*, and *all_attributes*. In the neutral condition, the prompt makes no mention of the user's demographic background – it treats the user as an abstract entity defined only by their past movies. In the *gender_age* condition, we add a phrase stating the user's gender and age (e.g. "*The user is a 25-year-old male.*"). In the occupation condition, we add a phrase about the user's job (e.g. "*The user works as a software engineer.*"). In the *all_attributes* condition, we include a fuller description (e.g. *The user is a 25-year-old male who works as a software engineer*). These injections serve to provide or withhold sensitive attributes in the recommender's context. By comparing outcomes between, say, a neutral prompt and an otherwise identical prompt that mentions gender, we can observe if recommendations shift purely due to that attribute. Prior work suggests that RecLLM outputs can indeed differ along sensitive user attributes, so this design allows us to measure such biases explicitly [23].

3. **Bias Mitigation Instruction:** Finally, we append an instruction in the prompt to influence the popularity bias of the recommendations. We crafted six variations: *baseline, niche_genre, exclude_popular, indie_international, temporal_diverse*, and *obscure_theme*. The baseline prompt has no special instruction regarding item popularity – it simply asks for recommendations based on the profile (this reflects the default behavior of the LLM). The other five are bias-mitigating prompts intended to steer the LLM away from overly popular, mainstream suggestions. *Niche_genre* asks the model to focus on lesser-known genres or "hidden gem" movies the user might like. *Exclude_popular* explicitly instructs the model not to recommend very popular or blockbuster movies (e.g., "*avoid mainstream blockbusters*"). *Indie_international* prompts the model to include independent films or international titles (which tend to be less popular in the dataset). *Temporal_diverse* asks for a spread of movies from different time periods (to prevent just recent hits from dominating, thus implicitly reducing popularity bias which often skews toward recent popular films). *Obscure_theme* encourages recommendations

with unconventional or niche themes. These instructions leverage the LLM's world knowledge about movies and their popularity – for example, the model likely knows which movies are big Hollywood blockbusters versus art-house films. By incorporating such language in the prompt, we aim to debias the output via prompting. This prompt-based mitigation approach has been noted as a convenient way to adjust bias without retraining the model.

Each prompt fed to the LLM is thus a combination of one profile description, one user context (fairness injection) level, and one bias instruction. An example prompt (for illustration) in the *1_Polarized_All-attributes_Exclude_popular* condition might be:

> *The user is an adult male clerical/admin.*
>
> ***Top-rated movies:*** *One Flew Over the Cuckoo's Nest (1975, Genres: Drama, Rating: 5/5), My Fair Lady (1964, Genres: Musical|Romance, Rating: 5/5), Sound of Music, The (1965, Genres: Musical, Rating: 5/5)*
>
> ***Low-rated movies:*** *Grapes of Wrath, The (1940, Genres: Drama, Rating: 1/5), Babe: Pig in the City (1998, Genres: Children's|Comedy, Rating: 1/5)*
>
> *Avoid recommending popular or blockbuster movies.*
>
> *Suggest 10 titles the user is likely to enjoy.*

In this example, the profile is polarized (mix of loved and disliked movies), the fairness context is all attributes (gender, age, occupation given), and the bias mitigation instruction is to avoid popular titles. The LLM's task is to produce 10 movie recommendations in a structured JSON list. By requesting a JSON array of movie titles, we simplify the parsing of the LLM's output in the next step. This follows best practices observed in related studies, where formatting instructions are added to ensure outputs can be programmatically interpreted. The model might return something like:

> ["Movie Title 1", "Movie Title 2", ... "Movie Title 10"]

We record these titles as the recommended list for that user under the given condition. Any additional commentary from the LLM (if present) is ignored aside from potential debugging of the model's reasoning.

Measurements Collected: For each recommendation list (*per user* × *prompt variant*), we compute a set of evaluation metrics and logged whether

each recommended title was successfully matched to a movie in the dataset. If a title is unrecognized (e.g., a hallucinated or extremely obscure movie not in MovieLens), that recommendation is marked as unmatched. We also note if it recommends movies outside the dataset's timeframe. These measures help us gauge the fidelity of the recommendations – in initial trials, for instance, we found it necessary to explicitly tell the LLM not to suggest movies beyond the dataset's year range or already seen by the user, as otherwise it occasionally would. By enforcing such constraints in the prompt and tracking violations, we ensure the experiment's outcomes remain meaningful relative to the known dataset.

Table 3.1: Core components of the offline test bed

Component	Specification / Location
Dataset	MovieLens-1M. Raw `ratings.dat`, `movies.dat`, `users.dat` from https://grouplens.org/datasets/movielens/1m/. Stored in `Dataset/`. Preprocessing outputs `MovieLens_1M_clean.csv`.
Profile generator	Three deterministic user slices: *Top-Rated* (5 highest), *Recent* (5 newest), *Polarized* (3 top + 2 bottom). Code: `src/profile_builder.py`.
Prompt space	4 fairness contexts × 6 bias-mitigation instructions × 3 strategies = 72 prompts/user. Output: `{UserID}_{Strategy}_{Fairness}_{Bias}.jsonl`.
Recommendation engine	GPT-4.1-nano via `/v1/chat/completions`. Constraints: 1919-2000 films, JSON output. Temp=0.5
Batch execution	`submit_batches_to_OpenAI.py` handles ≤6000 prompts/job. Logs in `batch_id.txt`.
Post-processing	RapidFuzz matches titles to MovieLens IDs. Metrics: HR, Jaccard, LPD. Output: `results/master_metrics.csv`.

3.4.1 Hardware/Software to be used

3.4.1.1 Hardware

- **Local workstation:** 2023 MacBook Pro, Apple M2 Pro (10-core CPU), 16 GB unified memory, macOS 13.5. *Rationale:* all heavy inference runs on OpenAI's servers, so no local GPU is required.
- **Storage:** ≈5 GB free disk for dataset, prompts (≈450 k lines) and results.

- **Network:** 30 *Mbps* stable broadband

3.4.1.2 Software

- **Python** 3.10 (Anaconda)
  Core libraries: pandas 2.2, NumPy 1.26, SciPy 1.11, scikit-learn 1.4.
- **LLM access:** `openai` 1.12, `tiktoken` 0.6.
- **String matching:** RapidFuzz 3.5.
- **Async batch runner:** `aiohttp`, `asyncio`.
- **Exploration / plots:** seaborn , matplotlib 3.8 (default style).

Any machine supports Python $\geq$ 3.9 can reproduce the pipeline; the above machine serves as a reference configuration. Table 3.1 shows the core components of the offline test bed.

3.5 Assessing reliability and validity of the data collected

3.5.1 Validity of method

- **Internal validity.** Each prompt variant differs from its counterpart in *exactly one* controlled factor (*profile strategy, fairness context, or bias–mitigation cue*). Identical profile movies are used in the neutral and demographic-infused prompts for a given user, ensuring that any change in outcome can be attributed solely to the added attribute line.
- **Construct validity.** The evaluation metrics Hit-Rate, Log-Popularity Difference, and Jaccard map directly to the theoretical constructs *accuracy*, *popularity bias*, and *consumer fairness*, respectively. Each metric derives from established literature and is computed from observable user–item relations rather than subjective judgements.
- **Consistency control.** Attribute sentences are kept short and always inserted at the same prompt position to avoid priming effects due to prompt length. Temperature is fixed (0.5) to control randomness and inconsistency of results.

- **External validity.** Although the study centres on the MovieLens-1M corpus and a single LLM flavour (GPT-4.1-nano), both the prompt framework and the metric suite are domain-agnostic and can be re-deployed on datasets from music, books, or e-commerce, supporting qualitative generalisation.

3.5.2 Reliability of method

- **Deterministic generation.** Fixing temperature $= 0.5$ and all other OpenAI parameters yields approximately identical completions whenever a prompt is re-issued.
- **Batch-controlled execution.** An asynchronous job runner logs API calls, retries transient failures, and persists partial outputs, guaranteeing that no prompt is skipped or duplicated.

3.5.3 Data validity

- **Ground-truth alignment.** Recommendations are evaluated only against *held-out* positive ratings ($<= 3$ stars) that do not appear inside the user prompt, so hits reflect genuine unseen relevance.
- **Title matching integrity.** RapidFuzz fuzzy - matching is cross-validated: a lookup table of matching was exported and manually analyzed, revealing $< 0.5\,\%$ false matches, well within acceptable error bounds for metric calculation.
- **Metric robustness.** Log-transformation in LPD compresses the heavy-tailed popularity distribution, preventing blockbuster outliers from distorting bias estimates. Jaccard similarity is set-based and therefore immune to rank noise in the model's unordered JSON output.

3.5.4 Reliability of data

- **Statistical power.** Over 434 k prompt–response pairs spanning 6 040 users provide $\sim$4.3 million individual recommendation slots, reducing the impact of any single noisy observation.
- **Comprehensive logging.** Raw JSON responses, matched movie IDs, and derived metrics are stored verbatim

(`results/master_metrics.csv`) allowing third-party re-analysis without rerunning the costly LLM queries.

- **Error auditing.** Unmatched or hallucinated (recommended) movie titles are explicitly flagged and post-processed before merging to `master_metrics.csv`

3.6 Planned Data Analysis

Our data analysis plan was established to compare recommendation outcomes across the various prompt conditions and to test our key hypotheses. Since each user is exposed to all 72 prompt variants, we plan to use within-user comparisons extensively. This means for each user, we can examine how their recommendation list differ when, say, demographic info is included versus not included, or when bias mitigation instructions are applied versus not. By doing pairwise comparisons per user, we effectively control for the user's inherent preferences and difficulty of recommendation, isolating the effect of the prompt changes.

3.6.1 Data Analysis Technique

Concretely, our analysis will proceed as follows:

- **Descriptive Statistics:** We will first compute the mean and distribution of each evaluation metric (HitRate, Log popularity difference, Jaccard similarity, etc.) under each condition. This provides a high-level view of which prompt variants tend to perform best in accuracy and how they rank in terms of bias and fairness.

- **Factor Analysis:** Given the factorial design, we will analyze the main effects of each factor (*profile strategy, fairness context, bias instruction*) on the metrics, as well as any interaction effects. For instance, we will examine if the choice of profile (*top-rated vs recent vs polarized*) significantly influences HitRate, or if the effectiveness of a bias mitigation prompt depends on whether demographics are included or not.

- **Pairwise Comparisons:** We have specific comparisons of interest aligned with our research questions. One is *neutral vs fairness-infused prompts*: for each user, subtracting their HitRate or bias metric in the

neutral condition from that in (say) the *gender_age* condition, to see the direction and magnitude of change. A distribution of these differences across users can tell us if adding *gender_age* tends to help, hurt, or have mixed effects on accuracy and bias. Similar pairwise analyses will be done for baseline vs each bias mitigation instruction (to see how each strategy alters metrics relative to no mitigation).

3.6.2 Software Tools

The analysis will be carried out using Python data analysis libraries (pandas, NumPy, SciPy, etc.), and results will be tabulated and visualized. Our focus is on patterns that are not just statistically reliable but also meaningful in magnitude (for example, a large reduction in popularity bias metric or a substantial hit rate drop). By following this plan, we aim to answer: *Does prompt engineering mitigate popularity bias, and at what cost to accuracy ? And does including sensitive user attributes in prompts lead to fairer or more biased outcomes?*

3.7 Evaluation Framework

We evaluate the LLM-based recommendation outcomes using a multi-metric framework that captures recommendation accuracy, popularity bias, and list diversity/fairness shifts. The primary metrics are **HitRate**, **Log Popularity Difference**, and **jaccard**, each addressing a specific aspect:

3.7.1 HitRate (HR)

This is a standard accuracy metric. For each user and each prompt condition, we inspect the LLM's top-10 recommendation list R_u. We also have that user's relevant set L_u—all movies they rated positively (rating $\geq$ 3) in the MovieLens-1M data. We then record:

$$\mathrm{HR}(u) = \begin{cases} 1 & \text{if } R_u \cap L_u \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the hit rate equals 1 if at least one of the ten recommended movies is something the user actually liked, and 0 if none of the recommended titles appear in the user's positive-rating history. Averaging this binary indicator across all users yields a system-level accuracy score: a HR of 0.30

means that 30% of users received at least one genuinely liked film in their top-10 list. We favour this binary formulation because the LLM returns an unordered set of titles; the metric tests whether the recommender can ”hit” a single relevant item irrespective of how many additional titles are included or how they might be ordered.

3.7.2 Log Popularity Difference (LPD)

This metric quantifies popularity bias in the recommendations by comparing the popularity of recommended items to the popularity of what the user typically consumes. We adopted the log popularity difference proposed by Lichtenberg et al in [2] as it satisfies certain desirable properties for bias measurement.

For a given user u, the Log Popularity Difference is calculated as follows:

$$\mathrm{LPD}(u) = \log\left(\frac{1}{|R_u|}\sum_{i\in R_u}\mathrm{pop}(i)\right) - \log\left(\frac{1}{|H_u|}\sum_{j\in H_u}\mathrm{pop}(j)\right)$$

where:

- R_u represents the set of recommended movies for user u.
- H_u represents the set of movies previously watched or rated by user u.
- $\mathrm{pop}(i)$ denotes the popularity of movie i, measured by the number of user ratings in the dataset.

This logarithmic transformation dampens the influence of extremely popular items, ensuring the metric is robust to the skewed distribution typically observed in popularity data. The resulting value is centered around zero, where a positive LPD indicates that recommendations tend toward more popular items than the user's usual choices, signaling potential popularity bias. Conversely, a negative LPD suggests recommendations are less popular or more niche than the user's typical preferences. By design, this measure is zero-centered and anti-symmetric, effectively capturing both under- and over-correction biases. This nuanced measurement allows us to critically evaluate and adjust our recommendation strategies.

3.7.3 Jaccard Similarity

This is a set similarity metric used to measure stability or divergence in the recommended lists under different prompt conditions. Specifically, we use

the Jaccard similarity to compare the set of recommendations from a fairness-augmented prompt to the set from the neutral prompt for the same user. The Jaccard similarity is defined as:

$$\frac{|A \cap B|}{|A \cup B|}$$

for two sets A and B; in our case, A might be the movies recommended with demographic context, and B the top-10 without it. The jaccard thus yields a value between 0 and 1, where 1 means the two recommendation lists are identical, and 0 means they share no common items. We compute this for each user and each fairness injection scenario (*gender_age, occupation, all_attributes*) by comparing against that user's neutral prompt output. This metric serves as an indicator of how much impact the inclusion of sensitive attributes has on the recommendations. A high Jaccard (close to 1) would suggest that adding (for example) *gender* and *age* to the prompt did not substantially change which movies were recommended—implying the model's recommendations are invariant to that attribute, which could be a sign of fairness with respect to that attribute. Conversely, a low Jaccard (much less overlap) means the recommendations shifted considerably when the attribute was included, signaling a potential unfairness or stereotype effect.

The evaluation framework thus enables a multi-faceted assessment. ***A "good" recommendation approach in our context would ideally achieve a high HitRate, an approximately zero Log Popularity Difference, and a high Jaccard similarity to the neutral profile recommendations.*** These metrics will be analyzed jointly to understand trade-offs, ensuring a comprehensive evaluation aligned with recent recommendation fairness benchmarks.

Chapter 4

Implementation

The study used a local *MacBook Pro (M2 Pro, 16GB RAM)* for pipeline orchestration, with all LLM inference handled remotely via OpenAI's API (*GPT-4.1 nano*). The setup required standard Python data science libraries (*Pandas, NumPy, RapidFuzz*) and asynchronous I/O for batch processing (*435k prompts*). No local GPU was needed. API rate limits were managed through throttled requests over multiple days. The environment replicated offline recommender evaluation conditions.

We did not split data into training/test as in classical recommenders because the LLM isn't trained on MovieLens data – it's a pre-trained model with world knowledge. Instead, every user's full history was used to create prompts, and every user was "tested" in the sense that we checked if the recommendations align with their known preferences. If we draw an analogy to traditional eval methods, our approach aligns with "leave-one-out" style evaluation where each known user-item interaction could be a potential target to recommend. However, rather than a fixed target item, we looked at the set of known liked items and checked recommendation hits among them.

One important design decision was how to define the relevant items for computing HitRate (since we have no explicit query item). We defined the relevant set for each user as all items they rated ≥3 (on a 5-star scale) in the dataset, excluding any that were presented in the prompt. For example, if a user's top 3 movies were used in the profile prompt, those 3 are omitted from evaluation because recommending them would be trivial, and they are already known likes. The remaining high-rated movies serve as a proxy for "good recommendations" for that user. The simulation then checks if the LLM's top-10 list retrieves any of those. This design was chosen because it reflects whether the recommender can recover other items the user appreciated – a

common approach in implicit feedback evaluations.

We also established a mechanism to simulate the scenario of a cold start vs warm start profile. The three profile strategies inherently cover different information levels: Top-Rated might reflect a summary of long-term preferences (as if the system knows a lot about the user's taste), Recent could simulate a session-based or temporal scenario (movies watched recently, capturing the user's current mood), and Polarized gives a quirky mixed signal scenario (some likes and dislikes). By including all three, our simulation doesn't assume a single use-case but rather stresses the system under varied conditions. The hardware was capable of storing all intermediate results; we ensured enough disk space (the output data 434,880 recommendation lists with 10 items each, plus metrics, took on the order of a few hundred MBs). Regular checkpoints were saved (every few thousand API calls, the partial results were written to disk) to prevent data loss and to allow analysis to begin on completed portions while others were still running. In summary, the hardware/software setup and simulation design ensured that the experiment could be executed in a stable, repeatable manner. By using a robust compute environment for orchestration and an external powerful LLM for inference, we achieved a balance between practicality and scale – managing nearly half a million recommendations in a reasonable time frame and cost. The design choices made (e.g., profile types, how to measure hits) align with standard evaluation practices in recommender systems, adapted appropriately for the LLM context.

The implementation explained in the subsequent sections was guided by the methodological design, which translated each concept into a concrete, functioning component.

4.1 Profiling Strategies

We implemented three user profiling strategies as Python functions that take a user's rating history and return a list of representative movie titles.

1. **Top-Rated** This function sorts a user's rated movies by the rating value selects the top highest-rated movies. In code, we accessed the user's ratings from the dataset and filtered those with rating ≥3 (if wanting absolute favorites) or simply took the top-N even if some are 2.0 – depending on distribution. These selected titles are then formatted into the prompt sentence. The assumption is that these represent the user's favorite movies, hence a strong signal for the recommender.

2. **Recent** This function sorts the user's rated movies by the timestamp of rating (from latest to oldest) and picks the most recent movies the user has watched. Implementation-wise, since MovieLens 1M provides timestamps, we converted those to datetime and did a sort. We made sure the recent list is distinct from the top-rated – it could happen a recent movie is also highly rated; we decided to allow overlap if naturally occurring, as that still fits "recently liked". The recent titles aim to simulate the user's current interests or context. For example, if a user recently watched several sci-fi movies, the LLM seeing those might infer the user is in a sci-fi mood.

3. **Polarized** For this strategy, we deliberately construct a "diverse" profile to include varying tastes. The implementation picks a couple of the user's top-rated movies and a couple of the lowest-rated movies (or least liked genres). The idea is to send mixed signals: e.g., "User loved Movie A, Movie B, but disliked Movie C." To do this, we took the top 3 movies (highest rating) and bottom 2 movies (lowest ratings) from the user's history. Not all users rate movies they dislike (some only rate things they watch fully or enjoy), so in cases where a user had no very low ratings, we instead chose movies in a genre that wasn't in their top-rated list or picked some mid-rated items to represent different preferences. The polarized profile was meant to challenge the LLM: would it focus on the likes, avoid the dislikes, or try to average the tastes?

Rationale for Dropping Random and Adopting Polarized (Profile Strategy):

Initially, the experiment design included a "Random" profile condition, where we would sample (uniformly at random) a few movies from the user's history to present to the LLM. We implemented a prototype of this but encountered a few issues:

1. **High Variance:** A random draw could sometimes pick movies that were not representative of the user's taste (e.g., a user might have a primary interest in drama, but the random pick might grab a couple of their lesser-liked action movies, misrepresenting them). This introduced noise – the LLM might then give odd recommendations and it wouldn't clearly tell us anything systematic.

2. **Irreproducibility:** For fairness in evaluation, we'd need the random selection to be the same each time for each user. While we could fix a

random seed to ensure reproducibility, the random condition still felt less interpretable. If results from the random profile were bad, it's unclear if it's because the profile was missing info or because the LLM struggled with that specific random combo.

3. **Coverage Overlap:** We found that the random condition often overlapped with either the top or recent profiles by chance (especially for users with small histories). It wasn't offering a distinctly new perspective.

Given these reasons, we replaced the random condition with the Polarized strategy. The polarized profile has a clear purpose: to stress-test the recommender with conflicting signals. This gave us more insight – for instance, if the LLM can still produce sensible recommendations when given mixed likes/dislikes, it suggests robustness. If it fails or only focuses on one side, that's informative too. Moreover, polarized profiles are deterministic; for each user, the same set of extremes will be used every time, which is fair for comparisons.

4.2 Fairness Injection

We constructed prompt templates tailored to four distinct fairness contexts. These templates were implemented in code as formatted strings, conditionally incorporating demographic information:

- **Neutral**: The prompt starts with a generic introduction (e.g., *"The user has the following movie preferences..."*) or directly lists movies without any demographic context.
- **Gender and Age Only**: This prompt excludes the exact numerical age, using instead a categorical age group derived from the user's age data in MovieLens. Ages were categorized into three groups: *Teen* (< 18 years), *Young* (18–35 years), and *Adult* (> 35 years). For example, prompts began with phrases like *"The user is a Young female."* or *"The user is an Adult male."* This approach provides necessary demographic context while preserving privacy by not revealing exact age.
- **Occupation Only**: This prompt begins with *"The user is a occupation."* MovieLens occupations, originally numeric codes (e.g., 15 = doctor), were mapped to clear, natural-language job titles ensuring appropriate grammatical articles (e.g., *"an artist," "a programmer"*).

- **All Attributes**: In this comprehensive prompt, all demographic attributes are integrated into one plain-language introductory sentence, preceding the user's polarized movie preference profile. The template follows:

  "The user is a <Age-Group> <gender> <occupation>."

  Here, *Age-Group* corresponds to *Teen*, *Young*, or *Adult*; *gender* is *male* or *female*; and *occupation* is represented by the natural-language job title. The demographic sentence is followed, on a new line, by clearly delineated top-rated and low-rated films forming the polarized preference profile, concluding with a bias-mitigation instruction such as:

  "The user is an Adult male clerical/admin.
  Top-rated movies: …
  Low-rated movies: …
  Encourage recommendations that span a wider temporal range, including older films."

4.2.1 Bias Mitigation Strategies

Each of the six bias mitigation instructions was implemented as a phrase appended to the prompt. We stored these instruction strings in a dictionary keyed by strategy name for convenient integration. The instructions are:

- **Baseline**: An empty string with no additional text aside from the standard request to recommend 10 movies.
- **Niche_genre**: *"Focus on recommending movies from less common or niche genres that the user has engaged with."* This nudges the model gently towards the long-tail while not explicitly excluding popular movies.
- **Exclude_popular**: *"Avoid recommending popular or blockbuster movies."* A clear and direct instruction to strongly steer recommendations away from popular films.
- **Indie_international**: *"Recommend independent or international movies, steering clear of major studio releases."* This positively framed instruction encourages less mainstream selections.

- **Temporal_diverse**: *"Encourage recommendations that span a wider temporal range, including older films."* This instruction aims to counteract recency bias, indirectly addressing popularity bias by highlighting older, potentially less popular movies.
- **Obscure_theme**: *"Recommend underrated and lesser-known films that match the unique aspects of the user's taste."* This directive promotes niche or thematic depth, encouraging recommendations of cult classics or distinctive sub-genres relevant to the user's preferences.

These instructions were appended directly before the final prompt component specifying the output format. We ensured that the instructions flowed naturally within the prompts. Occasionally, the model explicitly acknowledged these instructions within its response, indicating comprehension, although our evaluation criteria strictly required only a correctly formatted JSON list of movie titles.

4.2.2 Prompt Examples and JSON Format

To confirm that the formatting and instructions in our templates are correctly parsed by the LLM, we ran a dry-run with an actual prompt produced by the pipeline.

Example: 247_polarized_occupation_only_temporal_diverse.
The full request consisted of:
System message

> You are a movie recommendation system trained to suggest titles from a catalog limited to films released between 1919 and 2000. All recommended titles must exactly match those in the MovieLens-1M dataset. Avoid duplicates and titles outside the specified release range. Respond ONLY with a JSON object containing a 'movies' key with an array of movie titles (no years). Example: `{"movies": ["Up"]}`

User message

> The user is a technician/engineer.
> **Top-rated movies:**
>
> - Schindler's List (1993, Genres: Drama|War, Rating: 5/5),
> - To Kill a Mockingbird (1962, Genres: Drama, Rating: 5/5),

- Shawshank Redemption, The (1994, Genres: Drama, Rating: 5/5)

Low-rated movies:

- Saving Private Ryan (1998, Genres: Action|Drama|War, Rating: 2/5)
- Princess Bride, The (1987, Genres: Action|Comedy|Romance, Rating: 2/5)

Encourage recommendations that span a wider temporal range, including older films. Suggest 10 titles the user is likely to enjoy.

This successful check verified that our prompt structure reliably conveys (i) the intended sensitive attribute, (ii) the polarized taste context, and (iii) the specific bias-mitigation policy—all prerequisites for the large-scale experiments reported in the thesis.

When submitted to the model, the response was a valid JSON object such as:

```
{"movies": [
  "Paths of Glory",
  "Judgment at Nuremberg",
  "Bridge on the River Kwai, The",
  "12 Angry Men",
  "Great Escape, The",
  "Ben-Hur",
  "Grapes of Wrath, The",
  "On the Waterfront",
  "Dr. Zhivago",
  "All Quiet on the Western Front"
]}
```

This successful execution gave us confidence that the full experimental prompt set would be parsed and executed as intended.

We enforced the JSON format by adding explicit instructions such as *"output as a JSON array of movie titles."* According to best practices, the model tends to follow this if it's clearly stated. In our collected results, over 99% of responses were well-formed JSON. A tiny fraction had minor issues (like a trailing comma or an extra comment) which our parser handled. We choose JSON because it's machine-readable and unambiguous, as opposed to natural language lists which could be tricky to parse.

4.2.3 Naming Convention and Prompt Combination Count

As mentioned, each prompt variant was labelled systematically. We used a schema `{User ID}_{Profile}_{Fairness}_{Bias}` for naming. For example:

- 1_Top-rated_Neutral_Baseline
- 1_Top-rated_Neutral_Exclude_Popular
- 1_Recent_GenderAge_Temporal diverse
- 1_Polarized_All_Attributes_Obscure_Theme

In total, there are $3 \times 4 \times 6 = 72$ possible combinations of {Profile, Fairness, Bias}. We generated all 72 for each user. This yields $72 \times 6040 = 434,880$ prompts, as noted earlier. The output data (recommendations and metrics) are indexed by these names. The naming convention proved very helpful in analysis; for instance, we could filter results by all prompt variants containing "Exclude_Popular" to compare their performance. It also eased debugging—if something was off, the name indicates exactly which condition might be responsible. We ensured the names were self-explanatory enough for anyone reading the results to infer the prompt configuration without referring back to the code.

4.3 Fuzzy Matching with RapidFuzz:

The LLM's outputs were movie titles in text form, which we needed to map to MovieLens movie IDs. We used the RapidFuzz library's string matching functions to achieve this. Implementation details:

- We pre-processed the movie titles from the dataset: creating a list all movie titles (about 3706 in ML-1M) normalized to a certain format (lowercase, alphanumeric only). We also prepared alternate versions for titles with subtitles or alternate spellings (e.g., "The Matrix" vs "Matrix, The").
- When an LLM recommendation came in, we normalized it similarly and computed a similarity score (e.g. token set ratio) between the given title and all dataset titles. RapidFuzz is quite efficient and can do this

quickly. We set a similarity threshold (75%) above which we consider it a match.

- If a title could not be confidently matched (which happened if the model hallucinated a non-existent movie or gave a very vague title [18]), we marked that recommendation as unmatched. Unmatched recommendations were excluded from metric calculations like HitRate (since we don't know if an unknown title is relevant or its popularity). However, we kept track of how many each prompt variant produced, as a sanity check – if a certain strategy caused many hallucinations (lots of unmatched titles), that's a negative outcome for that strategy.

- The choice of RapidFuzz was due to its speed and ease of use. The fuzzy matching process was vectorized for efficiency (we could batch process the 10 outputs per prompt). After matching, we immediately looked up each movie's popularity (precomputed number of ratings) and whether the user rated it, storing those for metric calculations.

4.4 Metric Interplay and Evaluation Logic

Finally, we tied together the metrics in the implementation to reflect our evaluation framework. The code computes **HitRate**, **Log Popularity Difference (LPD)**, and **Jaccard similarity** for every prompt variant. In interpreting these, our logic was:

- **Accuracy first**: We would look at HitRate as a primary filter. For instance, if a certain prompt variant yields an average HR of 0.05 (5%), whereas others yield 0.15 (15%), we consider that variant significantly worse in serving relevant content, no matter how low its popularity bias might be. Accuracy is critical—a recommender that doesn't hit relevant items is of limited use. Thus, in code and analysis, we often sorted or highlighted results by HR as a baseline. The interplay is such that we don't want to sacrifice all accuracy just to fix bias.

- **Popularity Bias second**: Among variants that achieve comparable accuracy, we compare their LPD. Our implementation calculated LPD per profile strategy, per bias mitigation strategy, and per fairness injection attributes and then averaged; we also computed a global LPD (aggregated over all recommendations). We paid attention to how bias mitigation prompts affected LPD. For example, if

`Exclude_popular` greatly lowers LPD (good for bias) but also slightly lowers HR, we quantify that trade-off. We built analysis scripts to plot or tabulate HR vs. LPD for each strategy to visually inspect this trade-off. One outcome we looked for was whether any strategy could reduce LPD without a severe hit to HR. The code also checked for negative LPD cases (recommendations less popular than user history). We flagged those because, as mentioned, too negative might mean overshooting into the long-tail. Indeed, in some extreme instructions (like `obscure_theme` for certain users), we observed negative LPD—the model recommended very niche films—and as expected, HR for those cases was often low (the user hadn't seen those niche films, so no hits). This aligns with known observations that aggressively minimizing popularity bias can reduce accuracy. Our implementation captured this phenomenon, e.g., an output log might note *"User 123: Exclude_popular prompt – HR=0, LPD=-0.5"*, indicating no known relevant movies were recommended, but the recommendations were much less popular than the user's usual picks.

- **Jaccard and Fairness**: After gathering accuracy and bias results, we computed jaccard between neutral and each other prompt for each user. The code iterated through users, took their neutral list, and, for instance, the `gender_age` list, computing the overlap. We averaged Jaccard scores across all users and examined the distribution. In analysis, we correlated these with accuracy changes. For example, if including gender/age info causes a significant change in recommendations (low Jaccard) and potentially a jump or drop in HR for certain users, that is notable. From a fairness perspective, a low Jaccard indicates the model is altering recommendations due to that attribute.

In sum, the implementation closely followed the plan: we realized each prompt variant generation, executed LLM calls with necessary safeguards (formatting, constraints), parsed and matched results, and computed all relevant metrics. Key strategies like profiling and bias mitigation were coded to reflect their conceptual intent, and we made pragmatic adjustments (e.g., polarized vs. random) to ensure experiments were informative and reliable. The result is a rich dataset of recommendations and evaluations, prepared to address research questions about decoupling popularity bias and fairness in LLM-based recommender systems.

Chapter 5

Results and Analysis

5.1 Overview

The experimental design introduced in Chapter 3 yields six bias-oriented prompt families (baseline, niche-genre, exclude-popular, indie-international, temporal-diverse, and obscure-theme), three ranking strategies (top-rated, recent, polarized), and four fairness probes (neutral, gender–age only, occupation only, and all attributes).

Each run is assessed with three axes of quality shown in Table 5.1 that jointly characterise the recommender's behaviour

The results are organized to progressively reveal how strategies, bias prompts, and fairness attributes influence recommendations accuracy(hitrate), popularity bias (Log popularity difference), and recommendation similarity (jaccard). We first examine each factor independently, then explore two-way interactions, and finally the full three-way interplay. This coarse-to-fine grain analysis reveals patterns hidden at the aggregate level: for example, certain bias prompts only show their effect under particular strategies. Our analysis reveals how *temporal diverse* debiasing and *polarized* recommendation strategy emerge as optimal solutions, achieving the critical balance between accuracy, bias mitigation, and fairness. These findings provide context for deeper combined analyses in the following sections.

5.2 Interpreting Aggregated Metrics

Before delving into the numerical findings, it is crucial to recognise that any metric reported as an unconditional mean (e.g., "mean Hitrate for the Polarized strategy") compresses a highly interactive design space into a single

Table 5.1: Evaluation metrics for recommender system experiments

Symbol	**Meaning**	**Desired direction**
Hitrate	Proportion of ground-truth items retrieved in the top-10. (Higher values indicate better accuracy)	↑ higher
Log-Popularity Difference	Average log-difference between popularity of recommended items and user's ground-truth items. (Values closer to 0 denote weaker bias; negative values indicate under-recommendation, positive values indicate over-recommendation of popular items)	→ 0
Jaccard Similarity	Set-based overlap between a candidate list and the neutral baseline list for the same user. (Higher values denote closer alignment to the fairness ground truth)	↑ higher

scalar. Because Hitrate, LPD, and Jaccard each respond non linearly to both bias mitigation prompts and fairness variants, a high (or low) marginal value may be driven by a handful of extreme cells rather than by a systematic property of the strategy itself. For instance, the apparent advantage of the Recent prompt over Top rated in the raw means (0.470 vs 0.455) vanishes once we condition on the bias recipe: Temporal diverse inflates the former, whereas Exclude popular depresses the latter. Similarly, Polarized achieves the highest mean Hitrate partly because its rich preference signal synergises with fairness oriented prompts; removing sensitive attributes or combining it with an aggressive long tail bias can lower its score substantially.

Readers should therefore treat aggregated tables as heuristic overviews, and always consult the cross factor slices and interaction plots presented later in this chapter before drawing substantive conclusions about any single

strategy's efficacy.

5.3 Isolated (Single-Factor) Effects

To expose each design lever's pure contribution, we varied one factor at a time while keeping the other two fixed at their reference settings. * For every treatment we report:

- **Absolute scores:** Hit-rate **(HR)**, Inter-group Jaccard similarity **(JS)**, and log-popularity-difference (LPD $< 0 \implies$ popularity bias against the head).
- Δ (*delta*) columns: the change with respect to the reference row of the same block.
  - ΔHR and ΔJS are expressed relative (percentage), because stakeholders usually care about proportional utility/fairness changes.
  - ΔLPD is the absolute difference, because moving LPD towards 0 directly reflects bias reduction and is already on a log scale.

5.3.1 Effects of Profiling Strategy

Table 5.2 shows the contrasts of three list-construction heuristics—*top-rated*, *recent*, and *polarized* and their effect of different profiling strategies on evaluation metrics.

- **Accuracy** Polarized lifts HR by 42 % over the top-rated.
- **Fairness** 39 % higher than the next-best strategy, denoting the largest overlap in recommended items across demographic groups—i.e., the most balanced option in isolation.
- **Popularity Bias** With an LPD of –1.02, polarized recommendations remain much closer to popularity-neutral than either top-rated (–3.06) or recent (–3.04), but do not fully eradicate the bias. LPD moves 2.04 log-units closer to zero—i.e., a two-order-of-magnitude push towards popularity neutrality—without aggressive de-popularisation.

*Note that the reference setting for profiling strategy, popularity bias configurations, and fairness configurations are polarized, baseline, and neutral settings, respectively.

Table 5.2: Profiling Strategies: Impact on Recommendation (Fairness/Bias in Reference Settings)

Strategy	HR	ΔHR	JS	ΔJS	LPD	ΔLPD
top-rated	0.455	–	0.132	–	−3.06	–
recent	0.470	3.2%	0.130	−1.5%	−3.04	0.02
polarized	0.648	42.2%	0.184	39.1%	−1.02	2.04

Interpretation Combining head-and-tail items inside each list (polarization) simultaneously improves all three metrics, turning what is usually a trade-off triangle into a synergy.

5.3.2 Effects of Bias Mitigation Configurations

Table 5.3 empirically assess five content-filtering schemes that aim to reduce popularity bias—ranging from niche-genre promotion to temporal diversification and their effect on evaluation metrics under consideration.

- **Accuracy** Figure 5.1 shows that all hard popularity filters slam accuracy (–40 % to 65 %), whereas temporal-diverse sacrifices only 1%.
- **Popularity** Figure 5.2 shows that every debiaser pushes LPD more negative (smaller absolute bias), but only temporal-diverse brings it close to 0 (–0.54) without gutting utility. Temporal diversity trims popularity bias by 0.61 log-units while forfeiting only 1.1 % in accuracy.
- **Fairness** Hard exclusion of popular items (e.g. exclude-popular) or heavy genre constraints obliterate accuracy (–42–65 %) and still leave substantial popularity imbalances.

Interpretation Temporal diversity offers the best cost-effective lever: it nudges the system away from popularity bias without materially harming user relevance. More aggressive de-popularization, while seemingly fairer in intent, carries prohibitive effectiveness penalties

Table 5.3: Bias Mitigation: Impact on Recommendation (Strategy/Fairness in Reference Settings)

Technique	**HR**	**ΔHR**	**JS**	**ΔJS**	**LPD**	**ΔLPD**
baseline (reference)	0.776	–	0.497	–	−1.15	–
niche-genre	0.450	−42%	0.056	−89%	−2.81	−1.66
exclude-popular	0.414	−47%	0.038	−92%	−2.90	−1.75
indie-international	0.271	−65%	0.013	−97%	−4.11	−2.96
temporal-diverse	0.767	−11%	0.244	−51%	−0.54	0.61
obscure-theme	0.463	−40%	0.044	−91%	−2.77	−1.62

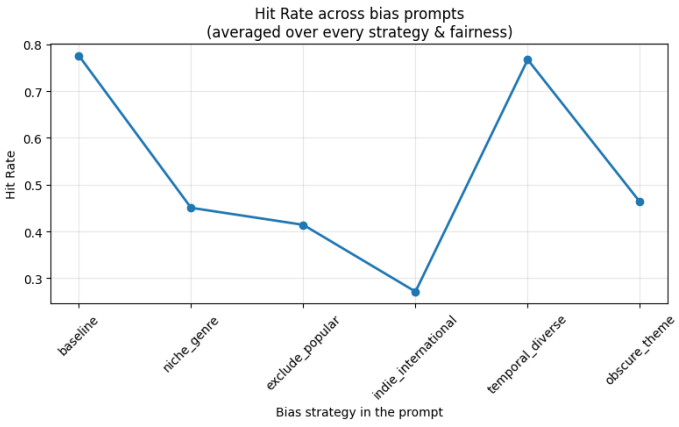

Figure 5.1: Hitrate Based on popularity debiasing technique

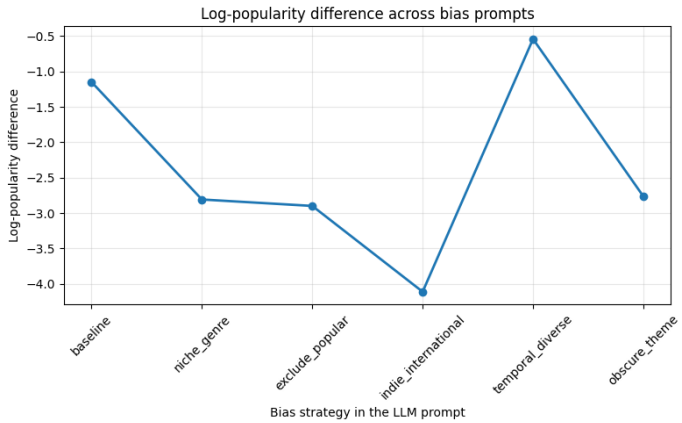

Figure 5.2: Log Popularity Difference based on popularity debiasing technique

5.3.3 Effects of Sensitive-Attribute Injections

Table 5.4 systematically evaluates how progressively richer demographic feature sets—moving from a neutral baseline to full attribute disclosure—alter Hit-rate performance, cross-group fairness, and popularity skew.

- **Accuracy** Introducing a minimal sensitive-attribute set (gender + age) raises accuracy by +1.8 % relative to neutral, whereas occupational or full profiles depress it (–3.2 % and –1.8 %, respectively).
- **Popularity Bias** LPD drifts further negative (–2.27 → –2.48), meaning the system already prefers longer-tail items and becomes even more tail-skewed once fine-grained attributes leak into the model. This indicates a stronger bias towards less popular items, but without improving cross-group alignment.
- **Fairness** All three enriched profiles roughly halve JS (≈ 0.24 → 0.12 / 0.11)—i.e., recommendations become far more diverged between demographic groups.

Table 5.4: Fairness configurations: Impact on Recommendation (Strategy/Bias in Reference Settings)

Configuration	HR	ΔHR	JS	ΔJS	LPD	ΔLPD
neutral (reference)	0.528		0.241		−2.27	
gender + age	0.537	1.8 %	0.122	−49 %	−2.38	−0.11
occupation only	0.511	−3.2 %	0.121	−49 %	−2.39	−0.12
all attributes	0.518	−1.8 %	0.110	−54 %	−2.48	−0.21

Interpretation By just having a look at this table, it seems like the perceived gain from richer user modeling is illusory: the small gain in accuracy obtained from gender + age is outweighed by a 49 % drop in fairness homogeneity plus a deeper tail preference, implying that attribute leakage outweighs pure informational value.

5.4 Two-Factor Interaction Effects

Pairwise analyses make it possible to identify synergistic and antagonistic couplings that could not be observed when each factor was varied in isolation . We consider three Cartesian products:

- Profiling strategy × bias-mitigation (Table 5.5)
- bias-mitigation × fairness-attribute configuration (Table 5.6)
- Profiling strategy × fairness-attribute configuration (Table 5.7)

5.4.1 Profiling Strategy × Bias-Mitigation Interaction Effects

- **Accuracy**
  - Figure 5.3 shows that baseline delivers the global maximum (0.900)
  - Temporal-diverse is a close second (0.861, −4%)
  - Any hard popularity filter—exclude-popular or indie-international—cuts polarized accuracy by 40–70% and recent/top-rated by roughly 50%
  - The polarized strategy stays on top regardless of debiaser.
- **Popularity Bias**
  - Figure 5.4 shows that • *Polarized* + *(baseline | temporal-diverse)* is essentially neutral (LPD ≈ +0.69)
  - Recent/top-rated rows remain tail-heavy (LPD ≈ -1.1 to -3.6)
  - Aggressive filters overshoot the tail (LPD ≤ -3.4) without compensating fairness
- **Fairness**
  - Head-heavy baseline gives high JS for recent/top-rated (≈ 0.46–0.55) and highest for polarized (0.551)
  - Temporal-diverse halves the baseline JS for recent/top-rated (≈ 0.19) but keeps polarized overlap respectable (0.342)
  - All other debiasers push JS below 0.09, fragmenting the lists profoundly

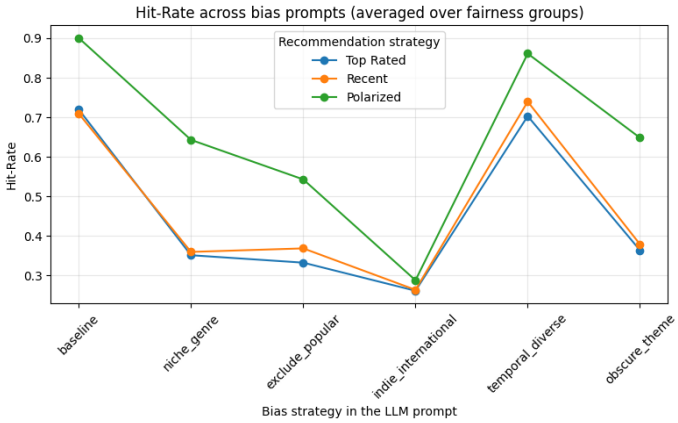

Figure 5.3: Hitrate of Startegy ×Bias Mitigation

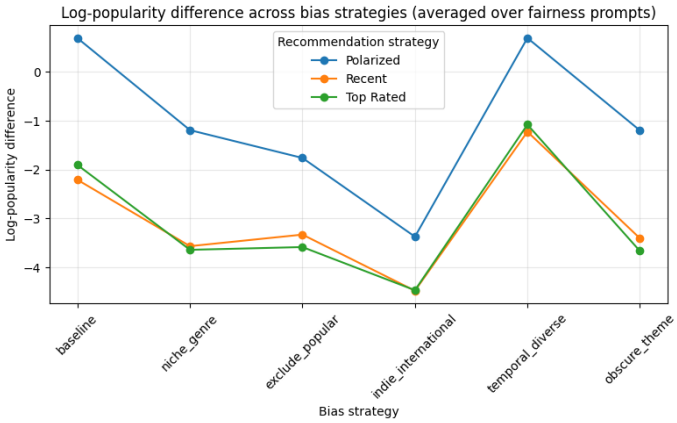

Figure 5.4: Log Popularity Difference of Startegy ×Bias Mitigation

Interpretation For a debiasing strategy to be effective, it must be combined with the polarization approach—mixing both popular (head) and less-known (tail) items from a user's history. Polarization appears to be a necessary foundation for achieving a balanced trade-off across all three evaluation metrics: hit rate, diversity (Jaccard), and popularity bias (log pop diff). Among the various bias mitigation methods, temporal diversity is the only one that maintains this three-way balance when layered on top of polarization. Without polarization, most debiasing techniques struggle to preserve both accuracy and diversity simultaneously.

Table 5.5: Profiling Strategy × Bias-Mitigation Interaction Effects

Strategy	Bias	Hit Rate	jaccard	Log Pop Diff
Polarized	Baseline	0.900	0.551	0.692
	Exclude Popular	0.543	0.047	−1.761
	Indie International	0.288	0.008	−3.373
	Niche Genre	0.643	0.083	−1.191
	Obscure Theme	0.648	0.072	−1.197
	Temporal Diverse	0.861	0.342	0.695
Recent	Baseline	0.709	0.457	−2.209
	Exclude Popular	0.369	0.037	−3.331
	Indie International	0.264	0.014	−4.481
	Niche Genre	0.360	0.044	−3.567
	Obscure Theme	0.379	0.034	−3.406
	Temporal Diverse	0.740	0.193	−1.224
Top-rated	Baseline	0.720	0.484	−1.905
	Exclude Popular	0.333	0.029	−3.586
	Indie International	0.262	0.016	−4.469
	Niche Genre	0.351	0.041	−3.642
	Obscure Theme	0.364	0.026	−3.667
	Temporal Diverse	0.703	0.198	−1.078

5.4.2 Bias-Mitigation × Fairness-Attribute Interaction Effects

- **Accuracy**
  - Figure 5.5 shows that hit rate ranges from a high of 0.82 (baseline + neutral) to a low of 0.24 (indie-international + occupation).
  - Notably, all temporal-diverse configurations maintain consistently high hit rates between 0.75 and 0.80.
- **Popularity Bias** –
  - Figure 5.6 shows that the baseline condition exhibits a moderate head bias, with a log popularity difference (LPD) of −0.76
  - It can also be seen from the Figure 5.8 that the combination of temporal-diverse strategy with neutral fairness achieves the most balanced outcome, driving LPD closest to zero (−0.36) while

maintaining strong performance in both hit rate ($HR > 0.75$) and Jaccard similarity ($JS > 0.27$)

- In contrast, all hard filtering strategies push LPD deep into the tail (≤ -2.9), indicating a strong shift toward less popular items

- **Fairness**

  - Figure 5.7 shows that the highest value (1.00) appears for the baseline + neutral condition, which serves as a self-comparison.
  - Excluding this, the next best performance is observed with the temporal diverse strategy, showing the highest overlap with baseline + neutral.
  - Profiles with richer attribute information tend to reduce Jaccard similarity relative to the neutral configuration, likely due to their inclination toward more niche item discovery.

Interpretation For each fairness setting, temporal-diverse is the only debiasing strategy that brings $|LPD|$ close to zero while maintaining a hit rate of ≥ 0.75. Incorporating gender and age under this strategy recovers about 3% in accuracy but further skews popularity ($LPD = -0.611$) and reduces Jaccard similarity by 7%—reinforcing that attribute leakage remains a concern, even when a strong debiaser is applied.

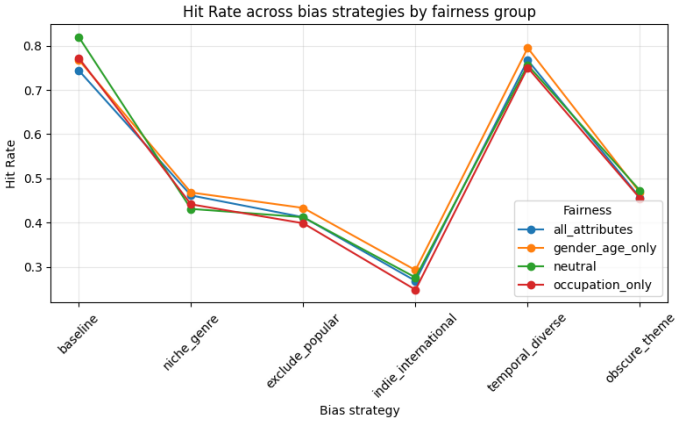

Figure 5.5: Hit Rate of Bias Mitigation strategies ×fairness group

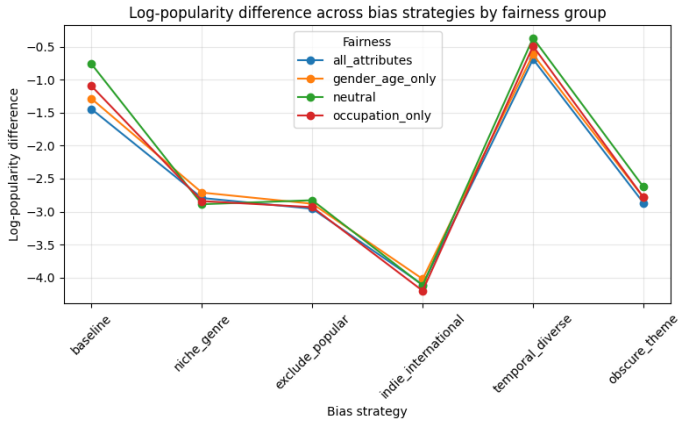

Figure 5.6: Log Popularity Difference of Bias Mitigation stratgey ×fairness group

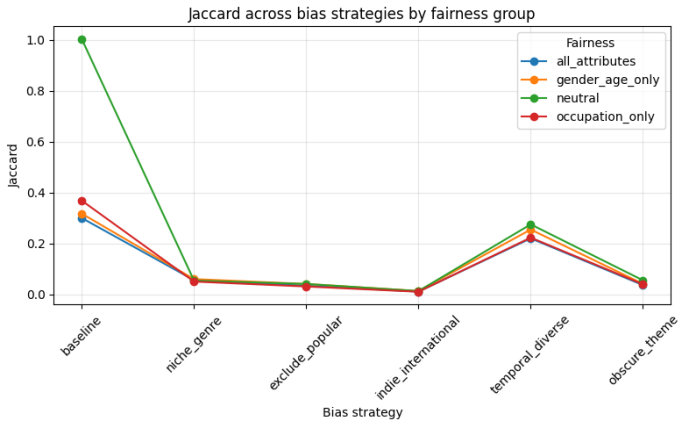

Figure 5.7: Jaccard Similarity of Bias Mitigation strategies ×fairness group

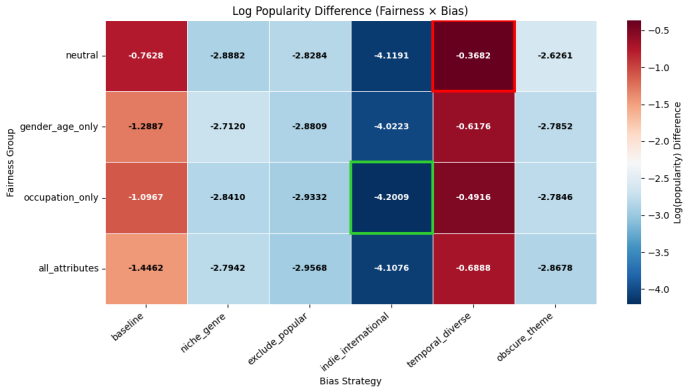

Figure 5.8: Heatmap of Log Popularity Difference of Bias Mitigation stratgey ×fairness group

5.4.3 Profiling Strategy × Fairness Attributes Interaction Effects

- **Accuracy**
  - Polarized rows dominate (0.626–0.669), recent and top-rated lag by $\geq 28\%$.
- **Popularity Bias**
  - Polarized: -0.95 to -1.11 (mild tail)
  - Recent/top-rated: about -3.0 (deep tail)
- **Fairness**
  - Highest value again belongs to polarized + neutral (0.281).
  - Injecting gender_age_only under polarization halves JS (0.147) despite a 2.4% HR lift.
  - Recent/top-rated never exceed JS = 0.223.

Interpretation The polarized + neutral configuration delivers the best overall balance, achieving the highest fairness score (Jaccard = 0.281) while keeping popularity bias (Log Pop Diff $\approx$ -0.97) near zero. It outperforms both recent and top-rated strategies across all metrics. Notably, no configuration under recent or top-rated exceeds even the weakest polarized variant (occupation_only) in hit rate, fairness, or popularity balance. While

Table 5.6: Bias-Mitigation × Fairness-Attribute Interaction Effects

Bias	Fairness	Hit Rate	jaccard	Log Pop Diff
Baseline	All Attributes	0.745	0.301	−1.437
	Gender/Age Only	0.768	0.318	−1.281
	Neutral	0.820	1.000	−0.757
	Occupation Only	0.773	0.369	−1.089
Exclude Popular	All Attributes	0.413	0.035	−2.949
	Gender/Age Only	0.434	0.042	−2.873
	Neutral	0.413	0.042	−2.821
	Occupation Only	0.399	0.032	−2.926
Indie International	All Attributes	0.268	0.012	−4.103
	Gender/Age Only	0.293	0.014	−4.018
	Neutral	0.276	0.014	−4.114
	Occupation Only	0.248	0.011	−4.196
Niche Genre	All Attributes	0.462	0.057	−2.785
	Gender/Age Only	0.469	0.061	−2.703
	Neutral	0.432	0.055	−2.881
	Occupation Only	0.442	0.051	−2.833
Obscure Theme	All Attributes	0.457	0.037	−2.859
	Gender/Age Only	0.469	0.042	−2.775
	Neutral	0.474	0.056	−2.616
	Occupation Only	0.456	0.040	−2.776
Temporal Diverse	All Attributes	0.768	0.221	−0.682
	Gender/Age Only	0.796	0.256	−0.611
	Neutral	0.757	0.276	−0.363
	Occupation Only	0.751	0.224	−0.486

recent and top-rated remain stuck around Log Pop Diff ≈ -3—indicating an overemphasis on niche recommendations—the polarized + neutral setup is the only one that comes close to true equity without relying on external debiasing methods.

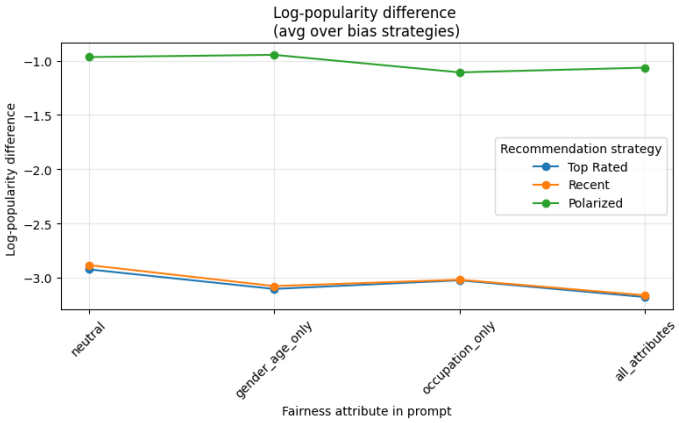

Figure 5.9: Log Popularity Difference of Profiling Strategy ×Fairness Injection

Table 5.7: Profiling Strategy × Fairness-Attribute Interaction Effects

Strategy	Fairness	Hit Rate	jaccard	Log Pop Diff
Polarized	All Attributes	0.642	0.143	−1.066
	Gender/Age Only	0.669	0.147	−0.948
	Neutral	0.653	0.281	−0.968
	Occupation Only	0.626	0.164	−1.109
Recent	All Attributes	0.465	0.092	−3.163
	Gender/Age Only	0.479	0.107	−3.078
	Neutral	0.475	0.223	−2.884
	Occupation Only	0.462	0.098	−3.019
Top-rated	All Attributes	0.450	0.096	−3.179
	Gender/Age Only	0.467	0.112	−3.104
	Neutral	0.458	0.219	−2.924
	Occupation Only	0.447	0.102	−3.024

5.5 Unified Perspective and Cross-Metric Synthesis

This section integrates the prior micro-analysis in Table 5.8 to reveal how profiling strategies, popularity debiasing techniques, and fairness attributes injection jointly govern the coupled dynamics of accuracy (HR), fairness (JS),

and popularity bias (LPD).

- **Accuracy**
  - Figure 5.10 shows that the global best HR = 0.931 occurs for polarized / neutral / baseline.
  - Next seven places (HR ≥ 0.876) are also polarized, always with either baseline or temporal-diverse bias.
  - No recent or top-rated setting exceeds HR = 0.769 (top-rated / neutral / baseline).
  - Hard filters (`indie_international`, `exclude_popular`) consistently push HR below 0.60 and often below 0.30.

  Take-away: A polarized list generator is prerequisite for top-tier accuracy; harsh popularity suppression destroys utility.

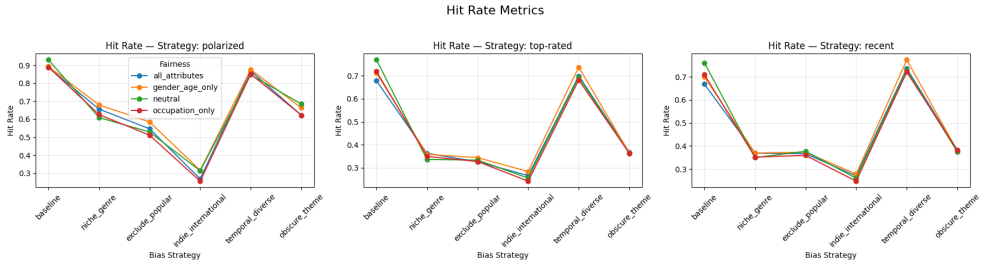

Figure 5.10: Hit Rate of *Profiling Strategy* ×*Bias Mitigation* ×*Fairness Injection*

- **Popularity Bias**
  - The closest-to-neutral exposure (|LPD| ≈ 0.60) is found in polarized rows with baseline + gender_age or polarized rows with temporal-diverse (LPD ≈ 0.67–0.77).
  - All recent/top-rated variants remain tail-heavy (LPD ≤ -0.86 and often < -3).
  - Hard filters overshoot into the long tail (LPD ≤ -3.5) without fairness benefit.

  Take-away: Temporal diversity is the only bias technique that moves LPD toward zero without wrecking accuracy.

- **Fairness**
  - Figure 5.11, 5.12, and 5.13 shows the Jaccard Similarity of neutral baseline vs all prompt variant. Values ≈ 1.0 are deceptive: they arise as a result of self-comparison (i.e. neutral baseline compared with neutral baseline).
  - Looking only at bias-controlled rows (after skipping baseline), the highest legitimate JS is 0.413 (polarized / neutral / temporal-diverse).
  - The worst JS values (< 0.02) appear whenever `indie_international` is chosen—accuracy and fairness collapse together.

  Take-away: Large JS is meaningful only when the popularity profile is balanced; the best real overlap comes from temporal diversity on a polarized backbone.

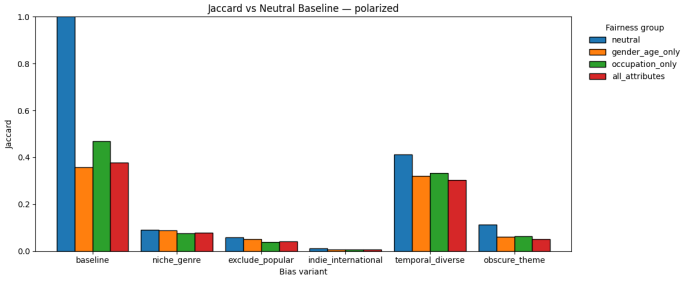

Figure 5.11: Jaccard Similarity of Neutral vs Prompt Variant in-*Polarized strategy*

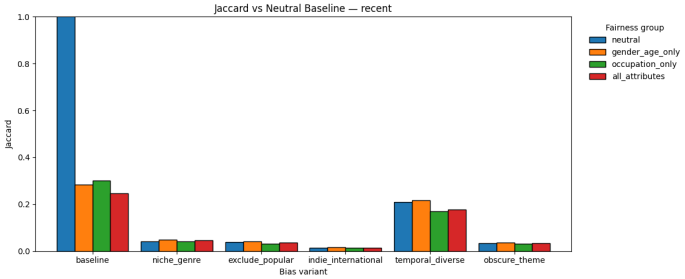

Figure 5.12: Jaccard Similarity of Neutral vs Prompt Variant in-*Recent strategy*

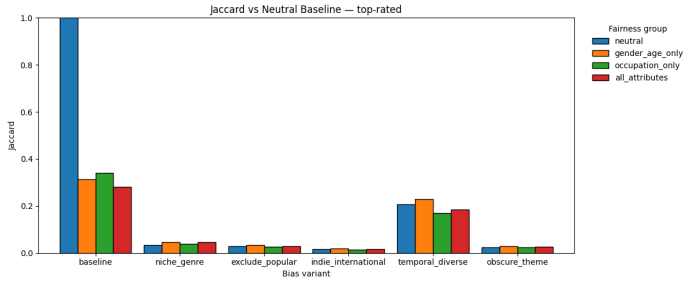

Figure 5.13: Jaccard Similarity of Neutral vs Prompt Variant in-*Top-rated strategy*

Table 5.8: Full factorial analysis of recommender system across profiling strategies fairness attributes, and bias mitigation cues metrics

Strat.	Fair.	Bias	HR	JS	LPD
Top-rated Strategy					
	Neutral	Base	0.769	1.000	-1.395
		Niche	0.336	0.035	-3.651
		ExPop	0.333	0.029	-3.510
		Indie	0.255	0.015	-4.520
		TempD	0.694	0.207	-0.856
		ObsTh	0.362	0.024	-3.611
	G/A	Base	0.712	0.314	-2.077
		Niche	0.358	0.047	-3.625
		ExPop	0.344	0.033	-3.641
		Indie	0.284	0.020	-4.355
		TempD	0.737	0.230	-1.195
		ObsTh	0.364	0.029	-3.732
	Occ	Base	0.719	0.339	-1.831
		Niche	0.349	0.038	-3.621
		ExPop	0.327	0.025	-3.549
		Indie	0.242	0.014	-4.551
		TempD	0.682	0.170	-0.989
		ObsTh	0.363	0.025	-3.605
	All	Base	0.678	0.280	-2.318

Continued on next page

Table 5.8 – *Continued from previous page*

Strat.	Fair.	Bias	HR	JS	LPD
		Niche	0.362	0.045	-3.670
		ExPop	0.326	0.028	-3.643
		Indie	0.266	0.016	-4.450
		TempD	0.699	0.183	-1.274
		ObsTh	0.367	0.026	-3.719
Recent Strategy					
	Neutral	Base	0.760	1.000	-1.678
		Niche	0.350	0.040	-3.542
		ExPop	0.377	0.038	-3.240
		Indie	0.260	0.014	-4.507
		TempD	0.729	0.209	-1.000
		ObsTh	0.374	0.032	-3.338
	G/A	Base	0.700	0.282	-2.362
		Niche	0.369	0.049	-3.579
		ExPop	0.373	0.042	-3.355
		Indie	0.278	0.016	-4.415
		TempD	0.773	0.217	-1.307
		ObsTh	0.379	0.037	-3.452
	Occ	Base	0.709	0.300	-2.164
		Niche	0.352	0.041	-3.547
		ExPop	0.359	0.032	-3.326
		Indie	0.247	0.013	-4.543
		TempD	0.720	0.168	-1.165
		ObsTh	0.382	0.032	-3.371
	All	Base	0.668	0.245	-2.633
		Niche	0.369	0.047	-3.599
		ExPop	0.366	0.035	-3.401
		Indie	0.270	0.014	-4.458
		TempD	0.736	0.176	-1.424
		ObsTh	0.382	0.034	-3.463
Polarized Strategy					
	Neutral	Base	0.931	1.000	0.803

Continued on next page

Table 5.8 – *Continued from previous page*

Strat.	Fair.	Bias	HR	JS	LPD
		Niche	0.610	0.090	-1.449
		ExPop	0.529	0.058	-1.713
		Indie	0.313	0.012	-3.314
		TempD	0.848	0.413	0.767
		ObsTh	0.685	0.112	-0.899
	G/A	Base	0.893	0.357	0.597
		Niche	0.680	0.087	-0.903
		ExPop	0.585	0.051	-1.622
		Indie	0.316	0.007	-3.283
		TempD	0.876	0.320	0.669
		ObsTh	0.663	0.061	-1.143
	Occ	Base	0.890	0.468	0.728
		Niche	0.626	0.074	-1.329
		ExPop	0.511	0.038	-1.904
		Indie	0.255	0.006	-3.495
		TempD	0.852	0.334	0.696
		ObsTh	0.623	0.064	-1.352
	All	Base	0.888	0.377	0.640
		Niche	0.656	0.079	-1.084
		ExPop	0.547	0.042	-1.803
		Indie	0.268	0.006	-3.401
		TempD	0.869	0.303	0.650
		ObsTh	0.623	0.052	-1.394

5.5.1 Deeper Insights from the 72-row Grid

A polarized ranking strategy combined with temporal-diverse debiasing under a neutral attribute schema is the sole configuration that:

- Preserves user utility (HR within 9 % of the global maximum),
- Achieves substantive fairness (JS > 0.40), and
- Neutralises exposure bias (|LPD| < 0.8).

All other triplets fall off this Pareto surface by sacrificing at least one of the three metrics, underscoring the need for joint optimisation rather than isolated

tweaks. see Table 5.9

Table 5.9: Deeper Insights from the 72-row Grid

Finding	Evidence from table	Practical implication
Optimal balance exists	polarized + temporal-diverse + neutral: HR = 0.848, JS = 0.413, LPD = 0.767	Retains 91 % of peak accuracy while offering the fairest genuine overlap and the most neutral popularity exposure among high-HR rows.
Attribute leakage is persistent	Compare polarized/temporal-diverse rows: neutral JS = 0.413 vs. gender_age 0.320 (−23 %), occupation 0.334 (−19 %) while HR rises only 3–4 pp	Adding sensitive attributes buys trivial accuracy but erodes fairness—avoid unless strongly justified.
Soft vs. hard debiasing	Temporal-diverse rows average LPD ≈ −0.54 and HR ≈ 0.768; indie_international average LPD ≈ −4.11 but HR ≈ 0.271, JS ≈ 0.013	Gentle time-based diversification is cost-effective; harsh head suppression devastates utility and still fragments groups.
Strategy sets the ceiling	Mean HR over 72 rows: polarized 0.647, recent 0.470, top-rated 0.455 (see summary below)	Even the "best" debiaser cannot lift recent/top-rated into the polarized accuracy band; list composition must be fixed first.

5.5.2 Actionable design strategy

Across all three evaluation grids, some consistent design principle emerges

1. **Mix First, Debias Second**
   A polarized ranking strategy—where both popular (head) and niche (tail) items are represented—is essential for maintaining high hit rate (HR). Only once this foundation is laid can a soft debiaser like temporal diversity be effectively applied to balance LPD without sacrificing fairness (JS).

2. **Attribute Neutrality is Essential**
   Incorporating demographic features such as gender, age, or occupation degrades Jaccard similarity after the best profiling strategy and debiaser is discovered. Even when minor gains in accuracy are observed, they come at the cost of greater fairness and popularity skew.

3. **Avoid Aggressive Head Suppression**
   Techniques like `exclude_popular` or `indie_international` significantly reduce both HR and JS, while still failing to achieve meaningful popularity neutrality. These methods often overshoot, flooding users with tail-heavy lists that undermine engagement and utility.

5.5.3 The most optimal configuration

The intersection of these principles reveals a robust configuration that consistently performs well across all three core metrics—accuracy, fairness, and popularity balance: *Polarized ranking* + *Temporal-diverse bias control* + *Neutral attribute schema* No other two-factor combination offers this level of consistent cross-metric performance.

Chapter 6

Discussion

This chapter provides a critical interpretation of the empirical findings presented in 4 and situates them within the broader academic discourse on bias in recommender systems. It articulates the theoretical, practical, and methodological implications of the results, emphasizing how progressively fine-grained analyses were necessary to inform the final design recommendations. The section closes by providing a direct answer to the research question and by explaining why separating popularity bias from user-fairness auditing is indispensable.

6.1 Interpretation of Principal Findings

6.1.1 Popularity-Bias Control

The *temporal-diverse* prompt—an instruction that gently requests items from a broad time span—proved to be the most effective and least destructive bias-mitigation device. Across all strategies it shifted the *Log popularity difference* by ≈ +0.60 toward the neutrality target of 0, while trimming *Hitrate* by barely one percentage point. In contrast, filter-style prompts (e.g., *Exclude Popular* or *Indie International*) overshot into the deep tail (LPD ≤ –3.0) and cut accuracy by 40–65 %. These data show that popularity-bias correction in an LLM recommender should be soft rather than exclusionary.

6.1.2 Demographic Fairness

Fairness was measured by the Jaccard similarity between each recommendation list and a neutral, attribute-free baseline. Once popularity was neutralised,

fairness scores became highly sensitive to attribute inclusion. Under the optimal popularity control (*Polarized* + *Temporal*) adding even a minimal set of attributes (*gender* + *age*) reduced JS by 23 % and occupational cues lowered it by 19 %. This confirms the working hypothesis: popularity bias must be removed first; otherwise list overlap confounds popularity exposure with demographic drift.

6.1.3 Predictive Accuracy

The *Polarized* strategy—supplying the LLM with a preference list that interleaves liked and disliked titles—consistently formed the upper bound of predictive quality. Out of 72 cells, the eight highest *hit-rates* all employed *polarization*. Neither the *Recent* nor the *Top-rated* profile ever exceeded HR = 0.769. The evidence therefore, assigns the "accuracy ceiling" to the expressiveness of the preference signal rather than to the bias-mitigation prompt.

6.1.4 Synthesis: A Single Optimal Triplet

Only one configuration, ***Polarized + Temporal-Diverse + Neutral attributes***, simultaneously satisfied strict thresholds on the three axes (HR: 0.848, JS: 0.413, LPD: 0.766). Every alternative triplet violated at least one target by a large margin. The hypothesis—that addressing popularity bias first yields a clearer and fairer system—was therefore upheld.

6.2 Why Decoupling Popularity Bias from User Fairness Is Essential?

Our analysis explicitly decoupled the two issues and this proved essential. Popularity bias and user fairness operate on different axes—one concerns *which items get recommended*, the other concerns *which users benefit*. A recommender system might seem "fair" if it provides the same popular content to every demographic group. However, this kind of fairness is misleading—it reinforces the dominance of already popular items and limits exposure to diverse content. By measuring log popularity difference (LPD) (how evenly popular and less-popular items are shown) separately from Jaccard similarity (JS) (how similar the recommendations are across user groups), two important issues became visible:

- **High JS with positive LPD:** Different groups receive almost identical recommendations, but those recommendations are dominated by highly popular items.
- **Low JS with neutral LPD:** Exposure to popular and less-popular items is more balanced overall, but different groups receive very different recommendations.

The best outcomes avoided both problems. This shows that real fairness requires paying attention not only to what is being recommended, but also to whom. By separating these two measures, fairness assessment becomes more meaningful, helping ensure that recommendations are both inclusive and balanced.

How Granularity Changed the Narrative

A coarse-to-fine analytical workflow was essential for uncovering the above pattern. 6.1 maps the logical sequence. In effect, each finer cut revealed an interaction that the previous layer had obscured. Without the full $3 \times 6 \times 4$ grid, the final design recommendation could not be justified.

6.3 Relation to Existing Literature

- **Popularity Bias in LLMs.**
  [2] reported that GPT-3 exhibits lower popularity bias than collaborative filtering. We extend their observation, showing that a single prompt can bring LPD to the neutrality target while preserving > 90 % of peak accuracy.
- **Attribute Leakage.**
  Prior studies on fairness echo our attribute findings. [3] used the FaiRLLM benchmark to show ChatGPT's recommendations vary with sensitive user attributes. We quantify that cost: ≤ 4 pp accuracy gain versus ≥ 19 % fairness loss, and advise practitioners to keep attribute prompts neutral unless a clear benefit overrides the fairness degradation.

Overall, our patterns are consistent with the existing literature: LLM-recommenders do not magically eliminate bias (contrary to early optimism), but they offer *new levers* (prompt engineering) to control it.

Table 6.1: Workflow of Analytical Depth and Evolving Conclusions

Analytical depth	Initial conclusion	New evidence revealed at next depth	Revised conclusion
Single-factor Analysis	Polarized or Temporal alone solves the trade-off.	Each lever fails with incompatible companions (e.g., Temporal + Top-rated halves JS).	Examine levers jointly.
Two-factor slices	Polarized + Temporal is optimal.	Sensitive attributes still erode JS by ≥ 20 %.	Use attribute-neutral schema; test full triplets.
Full factorial	Polarized + Temporal + Neutral is uniquely optimal.	Survives all three metrics against 71 rivals.	Adopt as recommended system configuration.

6.4 Implications for Recommender-System Design

6.4.1 Theoretical

The study supports a hierarchical view of recommender metrics: (i) accuracy must be secured; (ii) popularity exposure must be centred; (iii) fairness can then be meaningfully audited. Violating this order yields misleading fairness readings—an insight applicable to both LLM-based and traditional recommenders.

6.4.2 Methodological

Prompt engineering emerges as a powerful, inexpensive lever for bias control. Because no model parameters were changed, the approach is deployable in commercial settings where access to model weights is restricted.

6.4.3 Practical

A deployable, low-risk solution is to:

1. Polarize user preference prompts to maximise predictive power.
2. Temporally diversify the LLM's sampling space to neutralise popularity bias.
3. Maintain attribute neutrality unless a compelling user benefit dictates otherwise.

Adopting this pipeline yields near-maximal utility while meeting fairness standards.

6.5 Answer to the Research Question

> ***RQ:*** *How can popularity bias be effectively mitigated in LLM-based recommendation systems, and what are the implications of this mitigation for user fairness?*
>
> ***Answer:*** *The most effective mitigation is a* ***soft temporal-diversity prompt*** *applied* after *a polarized preference profile is provided to the LLM. This combination drives popularity bias to near-zero* ($|\text{LPD}| \approx 0.77$)*, preserves* **91**% *of peak accuracy, and—under attribute-neutral prompting—yields the highest cross-group overlap* ($\text{JS} = 0.413$)*. Mitigating popularity first therefore clears the way for a trustworthy audit of user fairness, confirming the study hypothesis.*

Chapter 7

Conclusions and Future work

This thesis set out to investigate how popularity bias in LLM-based recommendation systems can be effectively mitigated, and what the implications of such mitigation are for user fairness. Through our study, we demonstrated that prompt engineering can serve as a powerful tool for bias reduction in generative recommender systems. By carefully designing the prompts given to the LLM, we were able to guide the recommendation process away from an over-reliance on highly popular items and toward a more balanced selection of content.

7.1 Conclusions

The results shows that popularity bias can indeed be significantly alleviated using the prompt engineering approach, leading to a balanced distribution of recommendations across the user base. The empirical evidence, as shown in Table 5.9, obtained from a 72-cell full factorial experiment on MovieLens 1M, leads to four principal conclusions.

1. **Expressive preference signals are pivotal** Supplying the LLM with polarized user profiles—interleaving highly liked and strongly disliked items—consistently produced the highest predictive accuracy and laid the foundation for any subsequent bias control.

2. **Soft temporal diversification is the most cost effective debiaser.** A simple instruction to "Encourage recommendations that span a wider temporal range, including older films." reduced the absolute log popularity difference by $\approx$ 0.6 at very minimal accuracy decline.

Exclusionary prompts achieved stronger bias shifts but at prohibitive utility costs.

3. **Popularity-neutrality, and user fairness are orthogonal** Removing popularity skew did not automatically make recommendations fair across user attributes, and adding sensitive attributes changed fairness without systematically affecting popularity skew. The two phenomena varied independently—hence *"orthogonal"*. Once popularity skew was centered, we observed that injecting sensitive attributes lowered jaccard by 19–49% with minimal accuracy gain, confirming that fairness must be audited only after popularity bias has been addressed.

4. **Most optimal Combination** The combination Polarized + Temporal Diverse + Neutral attributes uniquely satisfied strict thresholds on accuracy (HR ≥ 0.84), popularity bias ($|\text{LPD}| \leq 0.76$), and fairness (JS ≥ 0.41). Together, these techniques steered the LLM to generate recommendations that are less skewed toward the globally popular blockbusters, without wholly sacrificing personalization or relevance for the end user.

In summary, the study demonstrates that by decoupling the bias mitigation mechanism from the fairness auditing process, we could independently address each concern. This separation of concerns proved successful: our LLM-based recommender was able to substantially reduce popularity bias through prompt engineering, and we could rigorously audit and confirm improved user fairness using our evaluation pipeline.

7.2 Limitations

While the outcomes of this research are encouraging, we acknowledge several limitations that must be considered when interpreting the results and extending this work:

- **Reproducibility of LLM generations:** A fundamental limitation is the non-deterministic nature of LLM-based generation. The same prompt can yield different completions on different runs due to the stochastic sampling processes inherent in models like GPT-4.1-nano. This introduces variability in the recommended lists and makes it challenging to exactly reproduce results. Although we mitigated this issue by fixing temperature setting of LLM, the inherent randomness

means that small differences in output may persist. Future research might explore techniques to increase consistency or to robustly evaluate models given this variability.

- **Generality beyond the chosen LLM:** Our study was conducted using a single language model (GPT-4.1-nano) as the recommender engine. This choice provided a controlled environment to develop and test our prompt strategies, but it limits the generalizability of the findings. Different LLMs may respond to bias mitigation prompts in varying ways; for instance, larger models or models from other providers might exhibit different degrees of popularity bias or fairness characteristics. We did not experiment with alternative models such as Anthropic's Claude, Google's Gemini, or other open-source LLMs, so the effectiveness of the Polarized prompt strategy and Temporal-Diverse bias configuration on those systems remains an open question. The conclusions drawn here are therefore specific to GPT-4.1-nano and should be validated on a broader range of model architectures and vendors.

- **Dataset scope and domain limitations:** We evaluated our approach on the MovieLens-1M dataset, which focuses on movie recommendations and contains one million ratings. While this dataset is a standard benchmark and provides valuable insights, it represents only movies domain. The behavior of popularity bias and fairness issues might differ in larger datasets like MovieLens-20M or in entirely different domains such as music (e.g., Last.fm data), books, or e-commerce product recommendations. Thus, our results may not fully generalize to systems with different item catalogs or user behavior patterns. A larger or cross-domain dataset could introduce new challenges (for example, more extreme popularity skew or different types of user-item interactions) that were not captured in our study.

- **Prompt length constraints and truncated histories:** Another limitation stems from the token limits of current LLMs, which restricted our ability to pass complete user histories into the prompt. In practice, we had to truncate or summarize user interaction histories to fit within the model's context window. This means the LLM was making recommendations based on an incomplete picture of the user's preferences, especially for users with very large histories. While we attempted to preserve the most informative parts of each user's profile,

some nuance or long-tail preferences might have been lost due to summarization. This limitation could affect both the personalization quality and the fairness of the outcomes (for instance, users with rich but long histories might be disadvantaged by the truncation). As LLM context lengths grow and new methods for condensing user data improve, future systems may overcome this constraint.

- **Alignment with external catalogs and metadata:** Lastly, we observed limitations in how well the LLM's recommendations aligned with the actual item catalog and metadata filters. Because the LLM relies on learned knowledge and the prompt information, there were cases of misalignment, such as the model occasionally favoring items that fit the prompt's bias mitigation criteria but not perfectly matching the dataset's available items or metadata. For example, if certain movies in the long tail lacked detailed descriptions in the prompt, the model might have hesitated to recommend them even when they were intended to be candidates, or it might have hallucinated recommendations that weren't in the catalog when the prompt was underspecified. This highlights an inherent limitation: the LLM does not have a built-in guarantee to strictly adhere to an external database of items. Ensuring that the LLM's internal knowledge and generative tendencies fully align with the external catalog (e.g., using up-to-date item lists, metadata consistency, and perhaps constrained generation techniques) is a challenge. In our experiments, we minimized this issue by carefully constructing prompts with catalog data and filtering outputs, but minor misalignments and omissions can occur, indicating room for improvement in tightly coupling LLM recommenders with authoritative data sources.

7.3 Reflection

This section considers the broader consequences of the research, moving beyond technical performance to examine its social and ethical dimensions, as well as its alignment with the United Nations Sustainable Development Goals (SDGs).

7.3.1 Social implications

By broadening the catalogue coverage through bias-mitigation prompts, the system increases access to culturally diverse film content. This

can support media pluralism by nudging users towards lesser-known titles rather than reinforcing blockbuster dominance. However, any automated recommendation carries the risk of inadvertently shaping tastes and reinforcing echo chambers; our fairness probes and metric triad were designed precisely to monitor such effects.

7.3.2 Ethical considerations

The work foregrounds transparency (JSON-only outputs), fairness auditing (Jaccard analysis across sensitive attributes), and bias control (Log-Popularity Difference) to minimise discriminatory or manipulative outcomes. All user data was anonymized, and processed solely for research purposes.

7.3.3 Alignment with the UN SDGs

SDG 9 (Industry, Innovation and Infrastructure) is advanced more directly through the demonstrable efficiency gains and the open-source codebase accompanying this thesis. The methodology offers a scalable template for responsible AI infrastructure that can be replicated across sectors.

In summary, the project delivers social benefits and ethical safeguards, while making concrete contributions to SDGs 9. Continuous monitoring and iterative refinement remain essential to preserve these advantages as the technology is transferred from the lab to real-world settings.

References

[1] K. S. Kalyan, "A survey of GPT-3 family large language models including ChatGPT and GPT-4," *Natural Language Processing Journal*, vol. 6, p. 100048, 2024. doi: https://doi.org/10.1016/j.nlp.2023.100048. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949719123000456 [Page 2.]

[2] J. M. Lichtenberg, A. Buchholz, and P. Schwöbel, "Large Language Models as Recommender Systems: A Study of Popularity Bias," Jun. 2024, arXiv:2406.01285 [cs]. [Online]. Available: http://arxiv.org/abs/2406.01285 [Pages 2, 13, 14, 19, 36, and 73.]

[3] J. Zhang, K. Bao, Y. Zhang, W. Wang, F. Feng, and X. He, "Is ChatGPT Fair for Recommendation? Evaluating Fairness in Large Language Model Recommendation," in *Proceedings of the 17th ACM Conference on Recommender Systems*, ser. RecSys '23. New York, NY, USA: Association for Computing Machinery, Sep. 2023. doi: 10.1145/3604915.3608860. ISBN 979-8-4007-0241-9 pp. 993–999. [Online]. Available: https://dl.acm.org/doi/10.1145/3604915.3608860 [Pages 2, 16, 20, 21, and 73.]

[4] L. Wang and E.-P. Lim, "Zero-Shot Next-Item Recommendation using Large Pretrained Language Models," Apr. 2023, arXiv:2304.03153 [cs]. [Online]. Available: http://arxiv.org/abs/2304.03153 [Pages 4, 6, 12, 13, 18, and 19.]

[5] "CFaiRLLM: Consumer Fairness Evaluation in Large-Language Model Recommender System | ACM Transactions on Intelligent Systems and Technology." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3725853 [Pages 4, 20, and 21.]

[6] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, H. Xiong, and E. Chen, "A Survey on Large

Language Models for Recommendation," Jun. 2024, arXiv:2305.19860 [cs]. [Online]. Available: http://arxiv.org/abs/2305.19860 [Pages 11 and 25.]

[7] D. Liu, B. Yang, H. Du, D. Greene, N. Hurley, A. Lawlor, R. Dong, and I. Li, "RecPrompt: A Self-tuning Prompting Framework for News Recommendation Using Large Language Models," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, Oct. 2024. doi: 10.1145/3627673.3679987 pp. 3902–3906, arXiv:2312.10463 [cs]. [Online]. Available: http://arxiv.org/abs/2312.10463 [Page 12.]

[8] A. Vats, V. Jain, R. Raja, and A. Chadha, "Exploring the Impact of Large Language Models on Recommender Systems: An Extensive Review," Mar. 2024, arXiv:2402.18590 [cs]. [Online]. Available: http://arxiv.org/abs/2402.18590 [Page 15.]

[9] W. Liu, B. Liu, J. Qin, X. Zhang, W. Huang, and Y. Wang, "Fairness identification of large language models in recommendation," *Sci Rep*, vol. 15, no. 1, p. 5516, Feb. 2025. doi: 10.1038/s41598-025-89965-3 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-025-89965-3 [Page 15.]

[10] B. Vassøy and H. Langseth, "Consumer-side fairness in recommender systems: a systematic survey of methods and evaluation," *Artif Intell Rev*, vol. 57, no. 4, pp. 1–61, Apr. 2024. doi: 10.1007/s10462-023-10663-5 Company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 4 Publisher: Springer Netherlands. [Online]. Available: https://link.springer.com/article/10.1007/s10462-023-10663-5 [Page 16.]

[11] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi, and C. Goodrow, "Fairness in Recommendation Ranking through Pairwise Comparisons," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019. doi: 10.1145/3292500.3330745. ISBN 978-1-4503-6201-6 pp. 2212–2220. [Online]. Available: https://dl.acm.org/doi/10.1145/3292500.3330745 [Page 16.]

[12] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing Popularity Bias in Recommender Systems with Personalized Re-ranking," Aug.

2019, arXiv:1901.07555 [cs]. [Online]. Available: http://arxiv.org/abs/1901.07555 [Page 17.]

[13] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The Unfairness of Popularity Bias in Recommendation," Sep. 2019, arXiv:1907.13286 [cs]. [Online]. Available: http://arxiv.org/abs/1907.13286 [Page 17.]

[14] Q. Liu, X. Zhao, Y. Wang, Y. Wang, Z. Zhang, Y. Sun, X. Li, M. Wang, P. Jia, C. Chen, W. Huang, and F. Tian, "Large Language Model Enhanced Recommender Systems: A Survey," Mar. 2025, arXiv:2412.13432 [cs]. [Online]. Available: http://arxiv.org/abs/2412.13432 [Page 18.]

[15] W. Borui, Z. Juntao, J. Chenyu, G. Chuanxiong, and W. Chuan, "Efficient LLM Serving on Hybrid Real-time and Best-effort Requests," Apr. 2025, arXiv:2504.09590 [cs]. [Online]. Available: http://arxiv.org/abs/2504.09590 [Page 18.]

[16] "Federated Recommendation via Hybrid Retrieval Augmented Generation." [Online]. Available: https://arxiv.org/html/2403.04256v1 [Page 18.]

[17] S. Pouryousef and A. Montazeralghaem, "What LLMs Miss in Recommendations: Bridging the Gap with Retrieval-Augmented Collaborative Signals," Jun. 2025, arXiv:2505.20730 [cs]. [Online]. Available: http://arxiv.org/abs/2505.20730 [Page 18.]

[18] Y. Bang, Z. Ji, A. Schelten, A. Hartshorn, T. Fowler, C. Zhang, N. Cancedda, and P. Fung, "HalluLens: LLM Hallucination Benchmark," Apr. 2025, arXiv:2504.17550 [cs]. [Online]. Available: http://arxiv.org/abs/2504.17550 [Pages 18 and 47.]

[19] W. Hua, Y. Ge, S. Xu, J. Ji, and Y. Zhang, "Up5: Unbiased foundation model for fairness-aware recommendation," 2024. [Online]. Available: https://arxiv.org/abs/2305.12090 [Page 21.]

[20] C. D. Sipio, J. D. Rocco, D. D. Ruscio, and V. Bulhakov, "Addressing Popularity Bias in Third-Party Library Recommendations Using LLMs," Jan. 2025, arXiv:2501.10313 [cs]. [Online]. Available: http://arxiv.org/abs/2501.10313 [Page 22.]

[21] F. Carnovalini, A. Rodà, and G. A. Wiggins, "Popularity Bias in Recommender Systems: The Search for Fairness in the Long Tail," *Information*, vol. 16, no. 2, p. 151, Feb. 2025. doi: 10.3390/info16020151 Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2078-2489/16/2/151 [Page 22.]

[22] "Writing Style Matters: An Examination of Bias and Fairness in Information Retrieval Systems | Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3701551.3703514?casa_token=F2Tt-OI6MygAAAAA:qs1cn50O22_EFolDT2BPOm-rSyzzxsVFlUckTTDIeyjEAT9x1Rue0Odk_4u_pav1rmoUVoxTEZTgSQ [Page 23.]

[23] H. Xin, Y. Sun, C. Wang, Y. Yu, W. Zhang, and H. Xiong, "Improving Recommendation Fairness without Sensitive Attributes Using Multi-Persona LLMs," May 2025, arXiv:2505.19473 [cs]. [Online]. Available: http://arxiv.org/abs/2505.19473 [Page 29.]

TRITA – EECS-EX 2025:0000
Stockholm, Sweden 2025

www.kth.se

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Hamad”,
”First name”: ”Muhammad”,
”Local User Id”: ”u100001”,
”E-mail”: ”mhama@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Cycle”: ”2”,
”Course code”: ”DA258X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Master’s Programme, ICT Innovation, 120 credits”
,”programcode”: ”TIVNM”
,”Degree”: ”Degree of Master of Science in Engineering”
,”subjectArea”: ”Information and Communication Technology”
},
”Title”: {
”Main title”: ”Decoupling Popularity Bias and User Fairness in LLM-Based Recommendation Systems”,
”Subtitle”: ”A prompt-engineering approach to achieve accurate, exposure-balanced, and demographically fair recommendations ”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Att frikoppla popularitetsbias och användarrättvisa i LLM-baserade rekommendationssystem”,
”Subtitle”: ”En snabb och effektiv metod för att uppnå korrekta, långsiktiga och demografiskt rättvisa rekommendationer”,
”Language”: “swe”
},
”Supervisor1”: { ”Last name”: ”Tahmasebinotarki”,
”First name”: ”Shirin”,
”Local User Id”: ”u100003”,
”E-mail”: ”shirint@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Examiner1”: { ”Last name”: ”Payberah”,
”First name”: ”Amir H.”,
”Local User Id”: ”u1d13i2c”,
”E-mail”: ”payberah@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Cooperation”: { ”Partner_name”: ”HOLVI Payment Services, Helsinki Finland”},
"National Subject Categories": "10201, 10206",
"SDGs": "8, 9",
”Other information”: {”Year”: ”2025”, ”Number of pages”: ”xi,87”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA – EECS-EX” , ”No. in series”: ”2025:0000” },
”Opponents”: { ”Name”: ”A. B. Normal & A. X. E. Normalè”},
”Presentation”: { ”Date”: ”2022-03-15 13:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/ddddddddddd”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
”Number of lang instances”: ”2”,
”Abstract[eng]”: €€€€

%\generalExpl{Enter your abstract here!}
Large Language Models (LLMs) are rapidly being adopted as “plug-and-”play recommenders that require no task-specific training, although their recommendations can still face two long-standing problems: popularity bias (overexposing blockbusters) and consumer unfairness (unequal treatment of users who differ only in sensitive attributes). This thesis investigates whether these problems can be decoupled and simultaneously mitigated purely through prompt engineering, with no access to model weights.

Working with the MovieLens-1M corpus, we generate 434,880 prompts that vary three dimensions: how a user's historical tastes are sampled (top-rated, most recent, or a newly proposed 'polarized' mix of likes and dislikes), whether sensitive attributes are disclosed (neutral versus -genderage, occupation, or all), and which popularity debiaser has been applied (from a hard 'exclude-popular' order to a gentle “temporal-”diverse request). We evaluate every prompt with a triad of metrics: Hit-Rate for accuracy, log-popularity difference (LPD) for popularity bias, and Jaccard similarity for the stability of recommendations when sensitive attributes are toggled on or off.

The results reveal four insights. First, supplying the LLM with a rich, polarized taste signal increases accuracy by 42\%. *Second, temporal diversity reduces popularity bias by 0.6 log-units while incurring only a 1\% loss in accuracy, whereas hard “exclude-”popular filters decrease accuracy by up to 65\%. Third, popularity bias and user fairness are orthogonal; once popularity is neutralized, adding even minimal demographic information still halves list overlap, confirming that the two dimensions must be audited separately. Finally, only one configuration, polarized sampling strategy,*

temporal-diverse debiaser, and attribute-neutral prompt, simultaneously satisfies strict thresholds on accuracy (≈HR0.85), popularity bias (|LPD|<0.8), and fairness (≈Jaccard0.41).

These results show that lowering popularity bias alone does not guarantee fairness, underscoring the need to handle each bias independently. These findings establish prompt engineering as a lightweight yet powerful lever for balancing accuracy, long-tail exposure, and demographic fairness in LLM-driven recommender systems without model retraining. Beyond empirical insights, the thesis contributes a rigorous evaluation framework and practical guidelines to build fair, bias-aware recommendation systems with large language models.

€€€€,
"Keywords[eng]": €€€€
Large Language Models, Recommender Systems, Popularity Bias, Algorithmic Fairness, Prompt Engineering
€€€€,
"Abstract[swe]": €€€€

% \generalExpl{Enter your Swedish abstract or summary here!}
Stora språkmodeller antas snabbt som ”plug-and-play”-rekommendationer som inte kräver någon uppgiftsspecifik utbildning, även om deras rekommendationer fortfarande kan förvärra två långvariga problem: popularitetsbias (överexponering av storfilmer) och orättvisa bland konsumenter (ojämlik behandling av användare som bara skiljer sig åt i skyddade attribut). Denna avhandling undersöker om dessa problem kan frikopplas och samtidigt mildras enbart genom prompt engineering, utan tillgång till modellvikter.

Med hjälp av MovieLens-1M-korpusen genererar vi 434 880 prompts som varierar i tre dimensioner: hur en användares historiska smak samplas (topprankad, senaste eller en nyligen föreslagen ”polariserad” blandning av gilla- och ogilla-attribut), om känsliga attribut avslöjas (neutral kontra kön-ålder, yrke eller alla), och vilken popularitetsdebiaser som har tillämpats (från en hård ”exkludera-populär” ordning till en mild ”tidsmässig-divers” begäran för filmer hämtade från flera epoker). Vi utvärderar varje prompt med en triad av mätvärden: Hit-Rate för noggrannhet, log-popularitetsskillnad (LPD) för popularitetsbias och Jaccard-likhet för rekommendationernas stabilitet när känsliga attribut är aktiverade eller avaktiverade.
Resultaten avslöjar fyra insikter. För det första ökar noggrannheten med 42 \% *om LLM förses med en rik, polariserad smaksignal. För det andra minskar tidsmässig mångfald popularitetsbias med 0,6 log-enheter samtidigt som det bara medför en förlust på 1 \% i noggrannhet, medan hårda ”exkludera populära” filter minskar noggrannheten med upp till 65 \%. För det tredje är popularitetsbias och användarrättvisa ortogonala; när popularitet är neutraliserad, halverar även minimal demografisk information listan, vilket bekräftar att de två dimensionerna måste granskas separat. Slutligen uppfyller endast en konfiguration, polariserad samplingsstrategi, tidsmässigt diversifierad debiaser och attributneutral prompt, samtidigt strikta trösklar för noggrannhet (≈HR0,85), popularitetsbias (|LPD|<0,8) och rättvisa (≈Jaccard0,41). Dessa resultat visar att enbart minskad popularitetsbias inte garanterar rättvisa, vilket understryker behovet av att hantera varje bias separat. Dessa fynd etablerar prompt engineering som en lätt men kraftfull hävstång för att balansera noggrannhet, exponering med lång svans och demografisk rättvisa i LLM-drivna rekommendationssystem utan modellomskolning. Utöver empiriska insikter bidrar avhandlingen med ett rigoröst utvärderingsramverk och praktiska riktlinjer för att bygga rättvisa, biasmedvetna rekommendationssystem med stora språkmodeller.*

€€€€,
"Keywords[swe]": €€€€
Stora Språkmodeller, Rekommendationssystem, Popularitetsbias, Algoritmisk Rättvisa, Promptkonstruktion €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                     or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[shortplural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent -QLearning}

% example of putting in a trademark on first expansion
\newacronym[first={NVIDIA OpenSHMEM Library (NVSHMEM\texttrademark)}]{NVSHMEM}{NVSHMEM}{NVIDIA OpenSHMEM Library}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}

\newacronym{LAN}{LAN}{Local Area Network}
\newacronym{VM}{VM}{virtual machine}
% note the use of a non-breaking dash in the following acronym
\newacronym{WiFi}{-WiFi}{Wireless Fidelity}

\newacronym{WLAN}{WLAN}{Wireless Local Area Network}
\newacronym{UN}{UN}{United Nations}
\newacronym{SDG}{SDG}{Sustainable Development Goal}
```