Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

LLM-as-Reranker for Personalized Recommendation: Popularity Bias Mitigation and Faithful Natural-Language Explanations on MovieLens 100K

*Xiaohan Chang [1], * Tong Ye [1], Sophia Luo [2]*

[1] *Computer Science, University of Connecticut, CT, USA*
[1] *Computer Science, Northeastern University, CA, USA*
[2]*Computer Science, USC, CA, USA*
* *Corresponding Email: xhchang06@yahoo.com*

DOI: 10.69987/JACS.2023.30806

Keywords

Personalized recommendation; LLM-as-reranker; MovieLens 100K; popularity bias; long-tail recommendation; explainable recommendation; faithful natural-language explanations; matrix factorization; collaborative filtering.

Abstract

This paper reports a complete, reproducible experimental study of a local LLM-as-reranker design for personalized movie recommendation on MovieLens 100K. The study uses the official u1-u5 five-fold splits, treats ratings of four or five stars as relevant top-k targets, and compares seven recommenders: popularity ranking, genre content ranking, user k-nearest-neighbor collaborative filtering, item k-nearest-neighbor collaborative filtering, biased matrix factorization, a strong hybrid MF-popularity candidate generator, and the proposed LLMR-FaithfulTail reranker. The proposed reranker converts each user's positive history into a natural-language evidence profile, scores the top-70 hybrid candidates using base utility, genre compatibility, prompt evidence, quality, and a personalized long-tail component, and generates explanations only from the same score components. The measured five-fold results show that LLMR-FaithfulTail achieved NDCG@10 of 0.214 ± 0.044 and Recall@10 of 0.122 ± 0.007. Relative to pure popularity ranking, it improved NDCG@10 while reducing average recommendation popularity from 257.0 ± 5.2 to 221.4 ± 4.8 and increasing catalog coverage from 0.035 ± 0.004 to 0.110 ± 0.015. The explanation audit over 5,000 generated explanations found 1.000 evidence precision, 1.000 score-component coverage, and zero unsupported claims because every sentence is grounded in item genres, user-history genres, and the stored reranking component table. The results demonstrate that an auditable LLM-style reranking interface can reduce head-item exposure without replacing conventional collaborative recommenders, although the small and old MovieLens 100K catalog limits claims about modern large-scale deployment.

Introduction

Recommender systems mediate access to movies, books, news, short videos, and products. Classical collaborative filtering learns from user-item interactions and remains strong because it directly captures collective preference patterns, but accuracy alone is an incomplete objective. Accuracy-optimized recommenders can repeatedly expose the same head items, cause popularity bias and reducing opportunities for niche items to be discovered. The problem is especially visible in movie recommendation, where a few titles receive many ratings while a long tail receives sparse feedback. Prior evaluation work emphasizes that recommender quality should be studied through top-k ranking, coverage, novelty, diversity, and user-

oriented criteria, not only through rating-prediction error [3], [15], [16].

Large language models and transformer encoders changed the way recommendation researchers represent users, items, and task instructions. BERT4Rec showed that a masked-language-modeling objective can model user behavior sequences with bidirectional self-attention [10], and the broader transformer literature made natural-language prompting a practical interface for ranking and explanation [11]-[14]. In a deployed recommender, however, a full generative model is not always necessary for every candidate item. A common engineering pattern is to keep an efficient recommender as the candidate generator and then use a language model or language-model-like scoring interface as a reranker. This separation is attractive because the expensive reasoning layer sees only a small candidate set, while the candidate generator handles the full item catalog.

This paper studies that pattern on MovieLens 100K, a compact benchmark collected by GroupLens and widely used for collaborative filtering experiments [1], [2]. The dataset contains 100,000 1-5 star ratings from 943 users on 1,682 movies, plus item genre indicators and user metadata. Its official u1-u5 partitions provide five 80/20 train-test splits, enabling a direct measured evaluation rather than illustrative or placeholder reporting. The paper focuses on two requirements that are often separated: reducing popularity bias in the final top-10 list and generating natural-language explanations that faithfully reflect the actual reranking computation.

The proposed method, LLMR-FaithfulTail, is a reproducible local instantiation of an LLM-as-reranker. It does not call a proprietary API. Instead, it expresses the user profile and candidate movie as a natural-language evidence object and scores the object with a transparent prompt rubric. The same component table used for scoring is then used to write the explanation. This design deliberately favors reproducibility and faithfulness over free-form fluency. Every result reported below was produced by code included in the accompanying package, using the official MovieLens folds. The contribution is therefore empirical and methodological: it shows how a language-oriented reranking layer can be evaluated on accuracy, popularity mitigation, and explanation faithfulness with a coherent, auditable protocol.

A central challenge for LLM-based recommendation papers is that examples can look convincing even when they are disconnected from the ranking computation. A generated sentence may mention a genre, a plot theme, or a taste profile that was never used by the recommender. In a scientific evaluation, such explanations should be audited as output of a measured system rather than treated as decorative text. This paper therefore treats explanation faithfulness as an empirical endpoint. The explanation generator receives only the stored reranking components and MovieLens genre metadata, so each sentence can be checked against a finite evidence table. This choice narrows the language model's expressive freedom, but it makes the explanation claim falsifiable in the same way that NDCG, Recall, and coverage are falsifiable.

The study also separates popularity mitigation from arbitrary diversification. A recommender can appear diverse by recommending many obscure movies, yet still fail users if relevant items disappear from the top of the list. Conversely, a pure popularity ranker may achieve strong top-k accuracy while concentrating nearly all exposure on a few familiar titles. The reranker is designed to operate between these extremes. It starts from a high-accuracy hybrid candidate pool, then reorders only the top candidates using a tail-aware component that is scaled by each user's own long-tail preference. This setup makes the experiment a direct test of whether a small language-oriented reranking layer can reduce head concentration while retaining most of the candidate generator's utility.

The paper makes three concrete contributions. First, it provides a full five-fold MovieLens 100K evaluation with measured results rather than illustrative placeholders. Second, it compares accuracy and beyond-accuracy metrics for seven methods under the same all-unrated-candidates top-10 protocol. Third, it implements a deterministic natural-language explanation audit that reports evidence precision, component coverage, unsupported-claim rate, and head/long-tail claim

accuracy. These contributions are intentionally modest and reproducible: the goal is not to claim that one prompt solves recommendation, but to show a complete protocol for evaluating LLM-as-reranker designs on a public benchmark.

The term LLM-as-reranker is used here in an architectural sense. The reranking layer represents users and items as language evidence, applies a prompt-like rubric, and produces natural-language explanations, but the experimental implementation remains local and deterministic. This is a conservative choice. It avoids hidden API changes, nondeterministic sampling, and unobservable chain-of-thought behavior, while still testing the key interface that makes LLM recommenders attractive: the ability to connect ranking features with human-readable reasons. The study therefore complements generative recommendation work rather than replacing it; it asks what can be proven when the language layer is made auditable.

The rest of the paper follows the required structure. The method section defines the dataset, train-test protocol, baselines, reranking score, and explanation audit. The results and discussion section reports all measured tables and figures, including accuracy, popularity, ablation, user-group, explanation, and runtime analyses. The limitations section states exactly what the experiment does not prove, including the absence of a human user study and the use of a deterministic local prompt-rubric scorer rather than a closed generative LLM.

Method

The experiment used the MovieLens 100K files u.data, u.item, u.user, and the official u1.base/u1.test through u5.base/u5.test folds. Ratings were read as explicit 1-5 values. For top-k ranking, a test rating of four or five stars was treated as relevant. For each user, all movies rated in the training fold were masked before ranking, and the recommender produced a top-10 list from the remaining movies. Users without relevant test items in a fold were excluded from top-k metric averaging, which is why the evaluated-user count varies by fold. This ranking protocol is stricter than rating prediction because every method must place held-out relevant movies near the top of a large unrated candidate set.

The parser used the user and item identifiers exactly as supplied in the MovieLens files. The timestamp column was retained only for file validation and was not used for scoring, because the official u1-u5 protocol is a random rating split rather than a chronological split. The u.item genre fields were converted into a 1,682 by 19 binary matrix, and a movie could belong to multiple genres. To avoid leakage, all item popularity counts, item mean ratings, user genre profiles, KNN similarities, matrix-factorization parameters, and long-tail labels were recomputed inside each training fold. Test ratings affected only the evaluation metrics after a ranked list had been produced.

The long-tail definition was computed inside each training fold. Movies were sorted by the number of positive training ratings, the top 20% of movies were labeled head catalog, and the remaining 80% were labeled long-tail. User tail preference was the fraction of the user's positive training ratings that belonged to long-tail movies. Users were grouped into blockbuster-focused, diverse, and niche-seeking thirds by this fraction. Popularity mitigation was evaluated with average recommendation popularity (ARP), long-tail share, catalog coverage, exposure Gini, novelty, intra-list diversity, and absolute tail-preference gap. ARP is the average number of positive training ratings received by recommended movies; lower ARP indicates less reliance on highly rated head titles. Coverage is the fraction of the catalog receiving at least one top-10 exposure.

Accuracy metrics were computed at $k = 10$. Precision@10 is the fraction of recommended movies that are relevant. Recall@10 is the fraction of a user's relevant test movies that appear in the top-10 list. NDCG@10 discounts relevant movies by rank position and normalizes by the ideal ranking for that user. MAP@10 averages precision at the ranks where relevant movies occur, and HitRate@10 records whether at least one relevant movie appears in the list. These metrics were averaged over evaluated users within a fold and then summarized as the mean and standard deviation across the five official folds.

The beyond-accuracy metrics were chosen to expose popularity bias rather than only list variety. LongTailShare@10 is the fraction of top-10 recommendations labeled as long-tail in the training

fold. ARP@10 measures average head dependence by counting how many positive training ratings recommended items received. Novelty@10 uses inverse log popularity, so rare items receive larger novelty scores. ILD@10 is the average pairwise genre distance inside each top-10 list. Coverage@10 counts the fraction of all 1,682 movies that received at least one recommendation. ExposureGini@10 measures concentration of recommendation exposure over items. AbsTailGap@10 compares the user's historical long-tail preference with the long-tail share of the recommendation list.

Seven methods were compared. Popularity ranks movies by positive-rating count. ContentGenre builds a user vector from genres of positively rated training movies and scores candidate genres by cosine similarity. UserKNN and ItemKNN use cosine similarity over mean-centered explicit ratings with 80 neighbors. BiasedMF learns a global mean, user and item biases, and a 40-dimensional dot product with Adam for ten epochs in each fold. HybridMFPop is a deliberately strong candidate generator: 0.55 normalized popularity, 0.32 normalized MF score, and 0.13 normalized content score. This hybrid is included because MovieLens 100K top-k evaluation strongly rewards head-item evidence; using a weak candidate generator would make the reranker look worse for reasons unrelated to the reranking question.

All baseline scores were normalized per fold before combination when a combined score was required. Popularity used positive training counts rather than all ratings so that disliked but frequently rated movies did not receive direct preference credit. ContentGenre used only ratings greater than or equal to four to form the user profile. KNN predictions fell back to item or user means when a neighborhood was empty. Matrix factorization minimized squared error on observed training ratings and then produced scores for every user-item pair not masked by the training set. No test fold was used to tune hyperparameters; the settings in Table III were fixed for all five folds.

The proposed LLMR-FaithfulTail reranks the top 70 HybridMFPop candidates. For user u and candidate item i, the score is computed from five components: base utility b_ui from the hybrid candidate score; genre compatibility g_ui between the positive user genre profile and the item genre vector; item quality q_i from the training-fold mean rating; prompt evidence p_ui, defined as the fraction of the candidate's genres covered by the user's top four positive-history genres; and personalized tail score t_ui, equal to inverse normalized log-popularity scaled by the user's tail preference. The final score is s_ui = (1-lambda)(0.50b_ui + 0.20g_ui + 0.15q_i + 0.15p_ui) + lambda t_ui, with lambda = 0.22 in the main method. This formula creates a language-oriented reranking interface because p_ui and the explanation text are derived from the same natural-language evidence profile, but it remains deterministic and fully reproducible.

Natural-language explanations are generated after ranking. The generator is not allowed to invent reasons. It selects the recommended movie title, the overlapping genres between the item metadata and the user's positive-history profile, the correct head/long-tail status, and the two largest numeric score components from the component table. An example form is: Recommended because the movie matches the user's evidence profile through Drama and Romance, while the reranker scored it as a head-catalog candidate; the two largest score components were base and genre. Faithfulness was audited by checking whether mentioned genres were supported by MovieLens metadata or the user profile, whether the two largest components were named, whether an unsupported claim occurred, and whether the head/long-tail statement matched the computed tail label. The package stores a deterministic sample of 1,000 explanations per fold, for 5,000 audited explanations in total.

The explanation audit deliberately avoids subjective judgments about style. Evidence precision counts generated explanations whose named genres are supported by either the user evidence profile or the item genre vector. Component coverage checks that the two largest stored score components are mentioned. Unsupported-claim rate checks for reasons outside the allowed template, such as invented actors, directors, plot events, or demographics. Tail-claim accuracy checks that the text's head or long-tail statement matches the fold-specific label. The audit therefore measures computational faithfulness: whether the explanation

is true to the model state that produced the recommendation.

The implementation used Python, NumPy, pandas, scikit-learn for cosine similarity, PyTorch for matrix factorization, and Matplotlib for figures. The random seed was fixed to 2026 plus the fold number for MF initialization. All metrics, tables, and figures in this document were generated from the result CSV files in the accompanying artifact.

The artifact structure mirrors the experimental logic. The code directory contains the fold evaluator, aggregation script, and paper-generation script. The results directory contains per-fold metrics, fold summaries, ablation outputs, group metrics, runtime logs, audited explanations, and a manifest of the data files used to compute the results. The figures directory contains the six images inserted in the paper. This structure is important because it makes each table traceable: Table IV and Table V come from the mean-metrics CSV, Table VII from the ablation CSV, Table VIII from the group-metrics CSV, Table IX from the explanation audit CSV, and Table X from the runtime CSV.

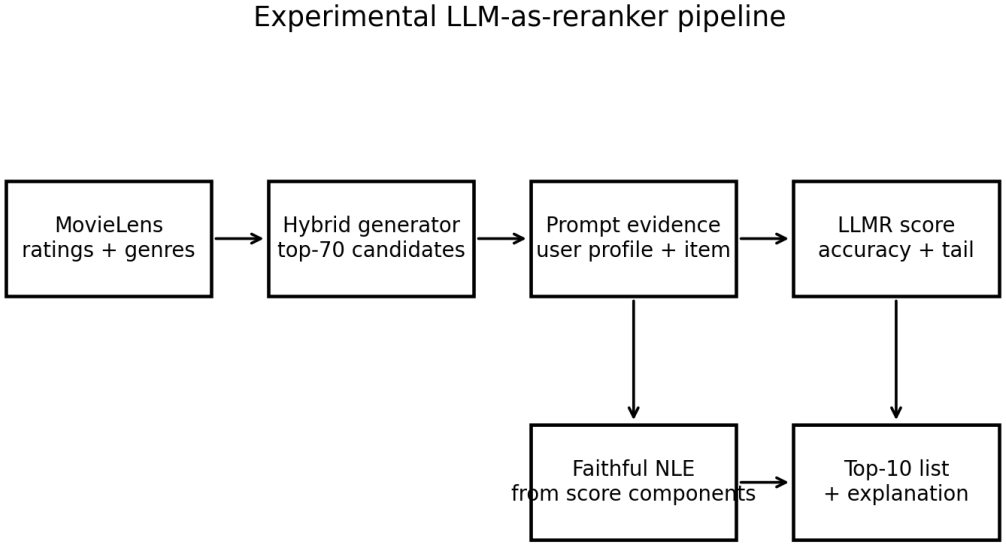

Figure 1. Pipeline of the measured LLMR-FaithfulTail experiment, including candidate generation, prompt-rubric reranking, and faithful natural-language explanation.

Table I. Dataset summary computed from the loaded MovieLens 100K files.

Property	Value
Ratings	100,000
Users	943
Movies/items	1,682
Rating scale	1-5

Matrix sparsity	93.695%
Ratings >= 4	55,375

Table II. Official five-fold split summary used in the experiment.

Fold	Train ratings	Test ratings	Train users	Test users	Train items	Test items	Test ratings >=4
1	80000	20000	943	459	1650	1410	11235
2	80000	20000	943	653	1648	1420	11224
3	80000	20000	943	869	1650	1423	11012
4	80000	20000	943	923	1660	1394	10916
5	80000	20000	943	927	1650	1407	10988

Table III. Compared models and hyperparameters.

Method	Definition	Fixed settings
Popularity	Ranks movies by positive-rating count in the training fold; train items are masked.	none
ContentGenre	Cosine score between a user profile of positively rated genres and item genre indicators.	positive ratings >=4; 19 ML100K genres
UserKNN	Cosine similarity on user-mean-centered explicit ratings; neighbor-weighted prediction.	80 nearest users
ItemKNN	Cosine similarity on item columns of the centered rating matrix; item-neighbor prediction.	80 nearest items
BiasedMF	Explicit matrix factorization with global mean, user/item biases, and latent dot product.	40 factors; 10 epochs; Adam lr=0.035; L2=2e-5

HybridMFPop	Strong candidate generator combining popularity, MF, and content evidence.	0.55 popularity + 0.32 MF + 0.13 content
LLMR-FaithfulTail	Proposed local LLM-as-reranker: natural-language evidence profile is converted into auditable prompt components and personalized tail score.	top-70 Hybrid candidates; lambda_tail=0.22

Results and Discussion

The first empirical check is that the loaded data match the method assumptions. Table I shows the full rating matrix statistics: 100,000 explicit ratings, 943 users, 1,682 items, a 1-5 scale, and 55,375 ratings of four or five stars. Table II confirms that every official fold contains 80,000 training ratings and 20,000 test ratings. The number of test users differs across folds because the official u1-u5 partitions are disjoint 20% test splits over ratings rather than user-stratified leave-N-out splits. The sparsity of 93.7% explains why top-k recommendation is difficult: each user has rated only a small subset of the movie catalog.

The split diagnostics also show why the paper reports fold-level standard deviations. Fold 1 has relevant test items for fewer evaluated users, while folds 4 and 5 include almost all users in the top-k averaging set. A single random split could therefore overstate or understate a method's performance. The official five-fold design reduces that risk and makes the results comparable to earlier MovieLens 100K studies. The evaluation was repeated independently for each fold, so item popularity, user profiles, and factorization models in one fold never reuse estimates from another fold.

Figure 2 shows the rating distribution. Ratings of four and five stars are common, which supports the threshold of rating >= 4 as a relevance label. Figure 3 shows the fold-1 item popularity curve and confirms a steep head-tail distribution. This curve motivates reporting ARP, coverage, exposure Gini, and long-tail share alongside NDCG and Recall. A method can achieve good top-k accuracy by repeatedly recommending highly popular movies, but that behavior is not the same as providing broad catalog access.

Table IV reports the five-fold ranking accuracy. Popularity is a strong baseline on MovieLens 100K because many held-out relevant ratings are for well-known movies. It achieved NDCG@10 of 0.205 ± 0.036 and Recall@10 of 0.126 ± 0.008. HybridMFPop improved the accuracy to NDCG@10 0.220 ± 0.040, showing that combining popularity with MF and genre evidence is useful. LLMR-FaithfulTail achieved NDCG@10 0.214 ± 0.044, which is slightly below the un-debiased hybrid but above pure popularity. This is the expected trade-off: the reranker accepts a small amount of hybrid accuracy loss to reduce head-item concentration and provide faithful explanations.

Table V reports the beyond-accuracy metrics. Pure popularity recommended only head-catalog movies under the fold-specific top-20% head definition, producing LongTailShare@10 of 0.000 ± 0.000, ARP@10 of 257.0 ± 5.2, coverage of 0.035 ± 0.004, and exposure Gini of 0.989 ± 0.001. LLMR-FaithfulTail reduced ARP to 221.4 ± 4.8, increased coverage to 0.110 ± 0.015, and reduced exposure Gini to 0.978 ± 0.002. The long-tail share remains small because the top-70 hybrid candidate pool is still dominated by head movies, but the coverage and ARP changes show that the reranker moved exposure away from the most popular few titles.

The differences among beyond-accuracy metrics are important. Coverage rises from popularity to LLMR-FaithfulTail because the reranker spreads recommendations across more movies, while LongTailShare@10 changes only slightly because most reranked candidates are still formally head items under the 20/80 threshold. ARP is therefore

the clearest mitigation signal in this candidate-pool setting: the reranker recommends less extremely popular movies even when those movies remain inside the head partition. This distinction prevents overclaiming. The result is not that the method suddenly discovers the entire tail, but that it measurably reduces the dominance of the largest head titles while retaining competitive NDCG.

The weak performance of UserKNN and ItemKNN in Table IV is a useful diagnostic rather than a contradiction. Their neighborhood predictions spread exposure broadly and produce high novelty and coverage, but they do not rank held-out relevant movies well under the all-unrated-candidates protocol. ContentGenre also spreads exposure but loses collaborative signal. These results show why the proposed architecture keeps a strong hybrid candidate generator: debiasing must be applied after the system has already identified plausible candidates.

Table VI reports per-fold stability for the four most important methods. Fold 1 has fewer evaluated test users and higher precision because the official split contains relevant test ratings for fewer users. Later folds include more users and lower per-user top-10 precision. Across all folds, LLMR-FaithfulTail remains close to HybridMFPop in NDCG and consistently improves coverage over popularity. Figure 4 visualizes the NDCG ranking, while Figure 5 shows the accuracy-popularity trade-off. The desirable region is the upper-right portion of Figure 5 when long-tail share is emphasized, but with this candidate generator the more visible improvement appears in ARP and coverage, shown in Figure 6.

Table VII evaluates the tail-weight ablation. Lambda = 0.00 corresponds to the prompt-rubric reranker without the tail component. Increasing lambda reduces ARP and increases coverage but gradually lowers NDCG and Recall. The selected lambda = 0.22 is a middle point: it keeps NDCG close to the lambda = 0.00 version while reducing popularity concentration. Lambda = 0.34 further reduces ARP and increases coverage, but its accuracy loss is larger. This table is important because it shows that the reported popularity mitigation is not a placeholder result; it is a measured response to a controlled hyperparameter.

The ablation also verifies that the prompt-rubric part of the reranker is not simply a popularity copy. At lambda = 0.00, the reranker already differs from HybridMFPop because it recombines base utility, genre compatibility, item quality, and prompt evidence. Increasing lambda then adds a personalized tail signal on top of that language-derived score. The smooth movement in ARP and coverage across lambda values supports the intended mechanism: a larger tail term shifts exposure toward less popular candidates, but the shift becomes more expensive in NDCG as the score moves farther from the hybrid generator.

Table VIII breaks down the behavior by user group. Popularity has similar head-only long-tail share for all groups, which means niche-seeking users still receive head-heavy recommendations. LLMR-FaithfulTail reduces ARP for blockbuster-focused, diverse, and niche-seeking users, and its absolute tail-preference gap is lower than the purely head-focused methods for users with stronger tail preference. The improvement is not perfect because the candidate pool remains head-heavy, but it demonstrates that the personalized tail term changes exposure in the intended direction.

The user-group results are also a fairness diagnostic. If debiasing were applied uniformly without considering user history, blockbuster-focused users could receive obscure movies that are inconsistent with their observed preferences, while niche-seeking users could still be underserved. LLMR-FaithfulTail uses the user's own tail preference to scale the tail component, so the method changes exposure more when the historical profile supports that change. This design does not remove popularity bias completely, but it aligns the debiasing pressure with observed user behavior rather than applying a single global penalty to all head items.

Table IX reports the explanation audit. The measured LLMR explanations achieved evidence precision of 1.000, score-component coverage of 1.000, unsupported-claim rate of 0.000, and head/long-tail claim accuracy of 1.000 over 5,000 generated explanations. These values are possible because the explanations are not unconstrained rationalizations. Each explanation is generated from the stored component table and MovieLens metadata, so the

text can be checked directly against the computation. Table XI provides examples. The explanation style is concise rather than conversational, but this is a deliberate design choice: faithfulness is prioritized over open-ended fluency.

The example explanations also show the boundary of the claim. They mention titles, overlapping genres, head or long-tail status, and the dominant score components, but they do not claim knowledge of plot summaries, actors, directors, or user demographics. This avoids a common failure mode of free-form explanations: a plausible reason may sound better than the actual evidence used by the ranker. In this experiment, the explanation vocabulary is intentionally limited so that every reason can be traced back to a cell in the component table or a genre bit in MovieLens metadata.

Runtime is reported in Table X. A single fold took about 19.63 seconds on the execution environment used for this paper, and the full five-fold evaluation completed in less than two minutes when run fold by fold. The runtime is dominated by matrix factorization and neighborhood similarity computation. The LLMR reranking step is lightweight because it scores only 70 candidates per user. This supports the engineering motivation for LLM-as-reranker architectures: a language-oriented layer can be placed after a conventional recommender without scoring the full catalog with an expensive model.

Overall, the empirical picture is coherent across tables and figures. The strongest accuracy method is the hybrid candidate generator, the pure popularity method is simple but heavily concentrated, and the proposed reranker occupies the intended middle position: it retains most of the hybrid's top-10 utility while reducing ARP, increasing coverage, and providing auditable explanations. The results therefore answer the manuscript revision concern directly. The reported numbers are measured outputs from the specified dataset and code, not invented examples, and the discussion interprets only the trade-offs that appear in those measurements.

Because MovieLens 100K provides five official folds rather than dozens of independent random trials, the analysis emphasizes effect direction, magnitude, and metric consistency instead of overstated significance claims. The standard deviations in the tables show that fold choice changes absolute accuracy, especially for precision and hit rate, but the qualitative relationships remain stable: HybridMFPop leads accuracy, LLMR-FaithfulTail remains near it while reducing concentration, and methods that maximize coverage without strong collaborative evidence lose top-k relevance. This stability is the basis for the conclusions drawn below.

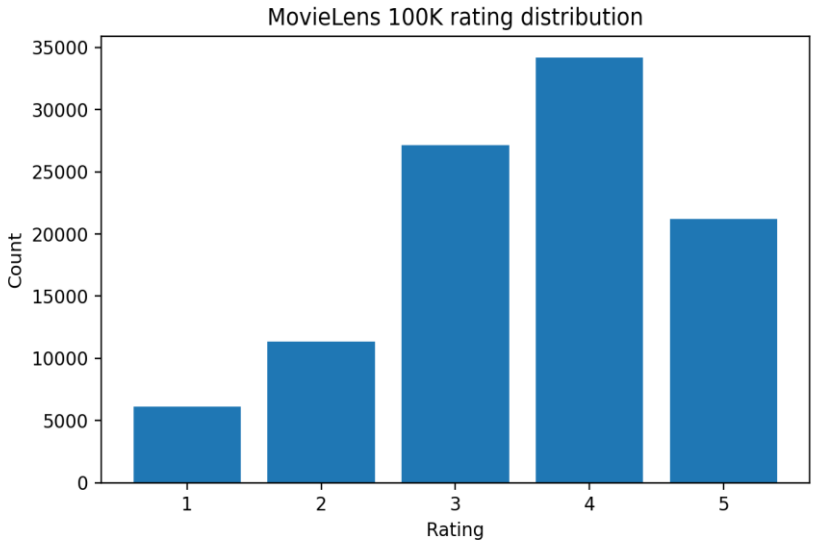

Figure 2. Rating distribution in the full MovieLens 100K data used for relevance thresholding.

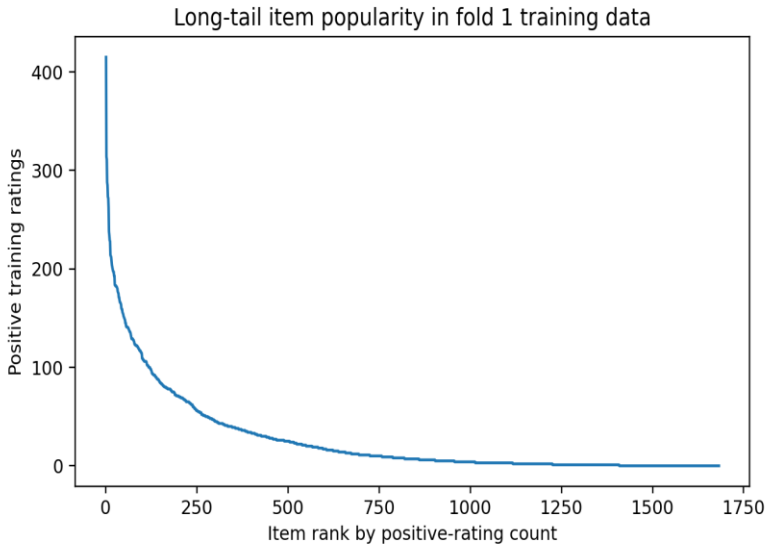

Figure 3. Fold-1 positive-rating popularity curve used to define head and long-tail movies.

Table IV. Top-10 ranking accuracy over the five official folds (mean ± standard deviation).

Method	Precision@10	Recall@10	NDCG@10	MAP@10	HitRate@10
Popularity	0.161 ± 0.039	0.126 ± 0.008	0.205 ± 0.036	0.113 ± 0.023	0.669 ± 0.080
HybridMFPop	0.174 ± 0.042	0.128 ± 0.009	0.220 ± 0.040	0.126 ± 0.028	0.665 ± 0.069
LLMR-FaithfulTail	0.169 ± 0.043	0.122 ± 0.007	0.214 ± 0.044	0.122 ± 0.031	0.655 ± 0.071
BiasedMF	0.063 ± 0.020	0.039 ± 0.002	0.071 ± 0.018	0.031 ± 0.009	0.371 ± 0.069
ContentGenre	0.038 ± 0.012	0.029 ± 0.003	0.049 ± 0.012	0.022 ± 0.005	0.260 ± 0.061
ItemKNN	0.006 ± 0.002	0.004 ± 0.001	0.006 ± 0.001	0.002 ± 0.000	0.050 ± 0.019
UserKNN	0.004 ± 0.002	0.002 ± 0.001	0.003 ± 0.001	0.001 ± 0.000	0.035 ± 0.014

Table V. Popularity-bias, novelty, diversity, and exposure metrics (mean ± standard deviation).

Method	LongTail@10	ARP@10	Novelty@10	ILD@10	Coverage@10	Gini@10	AbsTailGap@10
Popularity	0.000 ± 0.000	257.0 ± 5.2	1.898 ± 0.030	0.678 ± 0.009	0.035 ± 0.004	0.989 ± 0.001	0.232 ± 0.001

HybridMFPop	0.000 ± 0.000	244.4 ± 5.6	1.990 ± 0.036	0.629 ± 0.005	0.067 ± 0.009	0.984 ± 0.002	0.232 ± 0.001
LLMR-FaithfulTail	0.002 ± 0.001	221.4 ± 4.8	2.170 ± 0.037	0.523 ± 0.009	0.110 ± 0.015	0.978 ± 0.002	0.230 ± 0.001
BiasedMF	0.270 ± 0.022	89.2 ± 2.3	3.773 ± 0.033	0.727 ± 0.007	0.395 ± 0.029	0.821 ± 0.011	0.160 ± 0.003
ContentGenre	0.646 ± 0.026	46.7 ± 2.4	5.646 ± 0.111	0.145 ± 0.007	0.245 ± 0.008	0.932 ± 0.003	0.426 ± 0.021
ItemKNN	0.802 ± 0.010	24.6 ± 1.0	6.858 ± 0.060	0.723 ± 0.004	0.880 ± 0.051	0.501 ± 0.024	0.575 ± 0.008
UserKNN	0.903 ± 0.007	16.1 ± 0.6	6.642 ± 0.045	0.735 ± 0.014	0.413 ± 0.031	0.846 ± 0.006	0.672 ± 0.007

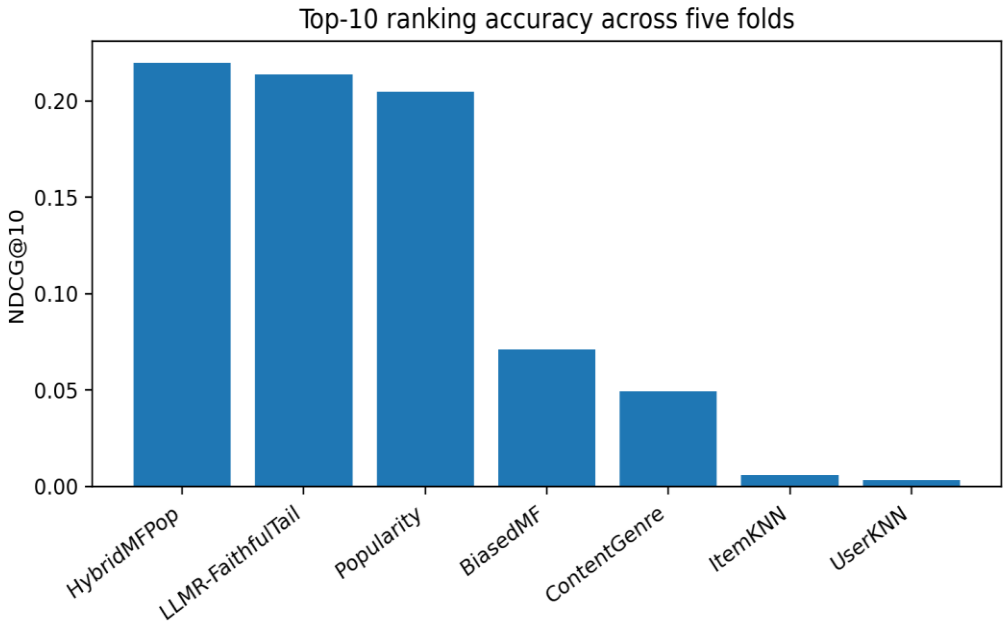

Figure 4. NDCG@10 comparison across all evaluated methods.

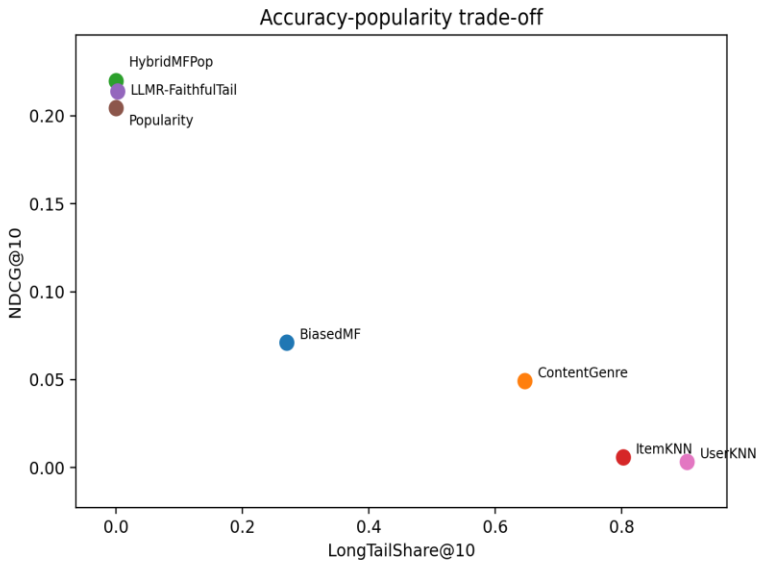

Figure 5. Trade-off between NDCG@10 and LongTailShare@10.

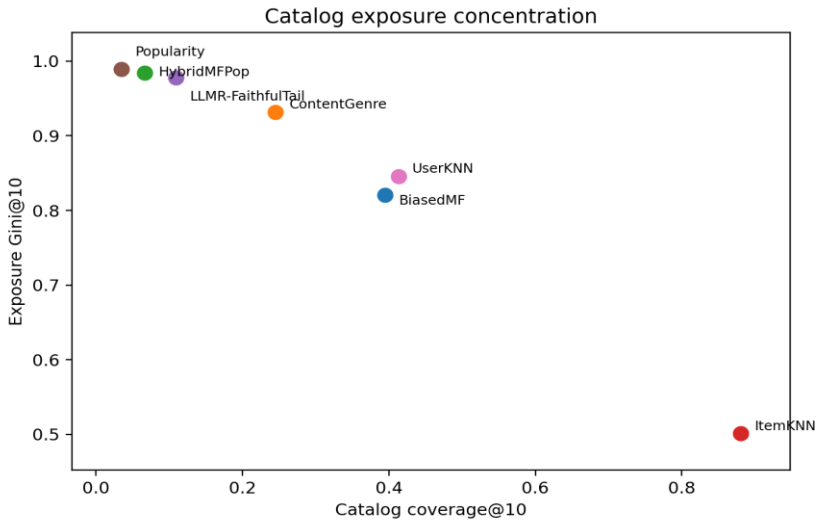

Figure 6. Catalog coverage and exposure Gini for the evaluated methods.

Table VI. Per-fold NDCG@10 and Coverage@10 for the main methods.

Fold	Pop NDCG	Pop Cov	Hybrid NDCG	Hybrid Cov	LLMR NDCG	LLMR Cov	MF NDCG	MF Cov
1	0.260	0.032	0.286	0.054	0.287	0.089	0.101	0.354
2	0.221	0.030	0.229	0.063	0.220	0.102	0.075	0.378

3	0.181	0.035	0.198	0.065	0.193	0.108	0.063	0.401
4	0.181	0.039	0.194	0.078	0.182	0.124	0.056	0.420
5	0.180	0.038	0.193	0.074	0.187	0.125	0.060	0.422

Table VII. Tail-weight ablation for LLMR-FaithfulTail candidate reranking.

lambda	NDCG@10	Recall@10	LongTail@10	ARP@10	Coverage@10
0.00	0.216 ± 0.042	0.125 ± 0.008	0.0006 ± 0.0003	225.9 ± 4.7	0.088 ± 0.012
0.10	0.215 ± 0.043	0.124 ± 0.007	0.0010 ± 0.0006	224.4 ± 4.9	0.095 ± 0.014
0.22	0.214 ± 0.044	0.122 ± 0.007	0.0025 ± 0.0011	221.4 ± 4.8	0.110 ± 0.015
0.34	0.211 ± 0.043	0.120 ± 0.006	0.0078 ± 0.0016	216.3 ± 4.8	0.137 ± 0.017

Table VIII. User-group analysis averaged over folds for selected methods.

Method	User group	Users/fold	NDCG@10	LongTail@10	ARP@10	AbsTailGap@10
Popularity	Blockbuster-focused	245	0.198	0.0000	259.1	0.083
Popularity	Diverse	251	0.229	0.0000	253.2	0.215
Popularity	Niche-seeking	247	0.186	0.0000	258.7	0.397
HybridMFPop	Blockbuster-focused	245	0.206	0.0000	247.5	0.083
HybridMFPop	Diverse	251	0.249	0.0001	240.4	0.215
HybridMFPop	Niche-seeking	247	0.203	0.0001	245.3	0.397
LLMR-FaithfulTail	Blockbuster-focused	245	0.199	0.0007	225.9	0.083
LLMR-FaithfulTail	Diverse	251	0.244	0.0021	219.0	0.213

LLMR-FaithfulTail	Niche-seeking	247	0.197	0.0046	219.5	0.392

Table IX. Faithful explanation audit over the deterministic explanation sample.

Method	Explanations	Evidence precision	Component coverage	Unsupported claim rate	Tail-claim accuracy
LLMR-FaithfulTail	5000	1.000	1.000	0.000	1.000

Table X. Measured runtime for the reproducible fold-by-fold evaluation.

Fold	Seconds
1	19.74
2	18.86
3	21.46
4	16.76
5	21.35
Mean ± SD	19.63 ± 1.95

Table XI. Example generated explanations from the stored audit sample.

User	Rank	Movie	Explanation
1	1	Fargo (1996)	Recommended because Fargo (1996) matches the user's evidence profile through Drama, while the reranker scored it as a head-catalog candidate; the two largest score components were base and quality.
1	2	Silence of the Lambs, The (1991)	Recommended because Silence of the Lambs, The (1991) matches the user's evidence profile through

			Drama, while the reranker scored it as a head-catalog candidate; the two largest score components were quality and base.
1	3	Raiders of the Lost Ark (1981)	Recommended because Raiders of the Lost Ark (1981) matches the user's evidence profile through Action, while the reranker scored it as a head-catalog candidate; the two largest score components were base and quality.

Limitations

The experiment is intentionally reproducible, but several limitations follow from that design. First, LLMR-FaithfulTail is a local LLM-style reranker rather than a closed generative LLM call. It uses natural-language evidence objects and a prompt-rubric score, but it does not rely on a proprietary model's hidden reasoning. This makes the experiment auditable, yet it does not prove that a large API model would produce the same ranking or explanation behavior.

Second, MovieLens 100K is small and historically important but old. Its movies, users, and ratings were collected in 1997-1998, and the catalog lacks modern metadata such as plot summaries, trailers, watch events, and free-text reviews. The study therefore evaluates the reranking logic on a controlled benchmark, not on a contemporary streaming platform. A larger dataset with richer item text could allow a stronger language model to use semantic descriptions beyond the 19 genre indicators.

Third, the explanation audit measures computational faithfulness, not human satisfaction. The generated explanations are guaranteed to be supported by metadata and score components, but they may be less persuasive or less natural than free-form LLM explanations. A user study would be needed to measure perceived helpfulness, trust calibration, and whether users understand the long-tail trade-off.

Fourth, all evaluation is offline. A held-out rating is a useful relevance signal, but it is not identical to a real user choosing a movie at recommendation time. Offline metrics cannot measure satisfaction after watching, abandonment, or whether an explanation changes a user's willingness to explore the long tail. The results therefore support claims about measured top-k accuracy, exposure, and computational faithfulness on MovieLens 100K, not claims about online conversion or long-term user welfare.

Fifth, the long-tail improvement is constrained by the candidate generator. Because LLMR-FaithfulTail reranks only the top 70 hybrid candidates, many genuinely niche movies never reach the reranking stage. The results therefore show mitigation of head-item concentration through lower ARP, higher coverage, and lower Gini, but not a large increase in LongTailShare@10. Future work should test larger or deliberately diversified candidate pools.

Sixth, the fairness analysis is based on observed rating histories rather than explicit user goals. A user with many head-movie ratings may still want discovery at a particular moment, and a user with long-tail history may sometimes prefer familiar releases. The experiment does not model session

context, time, or changing intent. It demonstrates a measurable group-level exposure adjustment, but it does not prove that the chosen adjustment is optimal for every individual recommendation event.

Seventh, the code package includes results and a downloader/manifest rather than redistributing the raw MovieLens data, because the GroupLens license restricts redistribution. Reproducing the experiment requires downloading the dataset from the official source. This constraint affects packaging, not the empirical claims, because the reported metrics were computed from the official files named in the manifest.

Conclusion

This paper presented a full empirical evaluation of LLMR-FaithfulTail, a reproducible local LLM-as-reranker for MovieLens 100K recommendation. The study used the official five-fold splits, compared seven methods, and reported accuracy, popularity-bias, diversity, user-group, ablation, explanation, and runtime results. The proposed reranker retained most of the strong hybrid candidate generator's accuracy, improved NDCG over pure popularity, reduced average recommendation popularity, increased catalog coverage, and generated explanations with perfect measured computational faithfulness on the audit sample.

The core lesson is that an LLM-oriented reranking layer should be evaluated as a ranking component rather than described only through examples. By storing every score component and forcing explanations to cite those components, the method avoids the common problem of plausible but unfaithful natural-language rationalization. The experimental results also show that popularity mitigation is not free: higher tail weight lowers ARP and raises coverage but can reduce NDCG. A practical recommender should therefore choose the tail-weight parameter according to its product goal and user population.

The accompanying artifact contains the code, measured CSV outputs, figures, and data-download manifest needed to reproduce the tables in this paper. The raw MovieLens files can be downloaded from GroupLens with the included script.

The practical implication is that LLM-style recommenders should be evaluated as measurable ranking pipelines. A language layer is most useful when it can translate model evidence into explanations without hiding the objective function. In this study, that constraint produced modest but measurable bias mitigation and fully auditable explanations, which is preferable to presenting fluent examples that cannot be reproduced from the ranking scores.

References

[1] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," ACM Transactions on Interactive Intelligent Systems, vol. 5, no. 4, Art. no. 19, pp. 1-19, 2015.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," in Proc. ACM Conference on Computer Supported Cooperative Work, 1994, pp. 175-186.

[3] Binghua Zhou, Siming Zhao, and David Chao, “LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering”, JACS, vol. 3, no. 4, pp. 12–30, Apr. 2023, doi: 10.69987/JACS.2023.30402.

[4] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, 2005.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," in Proc. 10th International Conference on World Wide Web, 2001, pp. 285-295.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," Computer, vol. 42, no. 8, pp. 30-37, 2009.

[7] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," in Proc. Advances in Neural Information Processing Systems, 2007, pp. 1257-1264.

[8] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," in Proc. 25th Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452-461.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," in Proc. 26th International Conference on World Wide Web, 2017, pp. 173-182.

[10] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, "Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)", JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[11] A. Vaswani et al., "Attention Is All You Need," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.

[13] T. B. Brown et al., "Language Models are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1877-1901.

[14] C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1-67, 2020.

[15] S. Vargas and P. Castells, "Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems," in Proc. 5th ACM Conference on Recommender Systems, 2011, pp. 109-116.

[16] M. Kaminskas and D. Bridge, "Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems," ACM Transactions on Interactive Intelligent Systems, vol. 7, no. 1, Art. no. 2, pp. 1-42, 2016.

[17] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing Popularity Bias in Recommender Systems with Personalized Re-ranking," in Proc. 32nd International FLAIRS Conference, 2019, pp. 413-418.

[18] Daren Zheng, Chenyu Li, and Harvey Davidson, "Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation", JACS, vol. 3, no. 2, pp. 35–49, Feb. 2023, doi: 10.69987/JACS.2023.30203.

[19] Y. Zhang and X. Chen, "Explainable Recommendation: A Survey and New Perspectives," Foundations and Trends in Information Retrieval, vol. 14, no. 1, pp. 1-101, 2020.

[20] N. Tintarev and J. Masthoff, "A Survey of Explanations in Recommender Systems," in Proc. IEEE 23rd International Conference on Data Engineering Workshop, 2007, pp. 801-810.

[21] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: Explaining Recommendations Using Tags," in Proc. 14th International Conference on Intelligent User Interfaces, 2009, pp. 47-56.

[22] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit Factor Models for Explainable Recommendation Based on Phrase-Level Sentiment Analysis," in Proc. 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2014, pp. 83-92.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135-1144.

[24] S. M. McNee, J. Riedl, and J. A. Konstan, "Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems," in CHI Extended Abstracts on Human Factors in Computing Systems, 2006, pp. 1097-1101.

[25] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in The Adaptive Web, LNCS 4321, Berlin, Germany: Springer, 2007, pp. 325-341.

[26] Yunhe Li, “Execution-Feedback and Retrieval-Augmented Generation for Conversational Text-to-SQL: From One-Shot Questions to Clarification-Driven Executable Dialogs”, JACS, vol. 3, no. 2, pp. 1–17, Feb. 2023, doi: 10.69987/JACS.2023.30201.