

Critiquing Helps, Robustly: A Stress-Tested Agentic RAG Recommender with LLM User Simulation

Menon Bharath Sreekumar
Nanyang Technological University
Singapore
BHARATH007@e.ntu.edu.sg

Abstract

In this study, we explore whether adding a single critique that is simulated through an LLM turn improves another LLM based recommender system using a full reproducible pipeline. This involves a TF-IDF retriever which narrows from a 50 item pool, an LLM ranks it and then for the agentic method that is proposed, the LLM user critiques the top 5 before the final re-ranking is done. On 253 different “leave one out” test users from the Amazon reviews 2023 *All Beauty* 5 core dataset, the critiquing agent’s final ranking beats popularity and item based CF, a single shot LLM ranking system and a retrieval augmented LLM without a critique by a consistent $2.1\text{--}2.4\times$ on HR@10. We further subject this number to two different stress tests and it holds up for both of them. Firstly, under a candidate pool that is harder, that is built from each of the target’s nearest lexical neighbors instead of just random negatives, the critiquing system still wins by $2.4\times$. Secondly, we compared our user simulator’s default oracle grounding, which is the full held out target item’s text, with a weak grounding and the two are statistically indistinguishable (bootstrap 95% CI on the difference includes zero in both regimes), and so the gain doesn’t really depend on the simulator knowing the answer because even a low information critique is useful. Hence, considering these tests together, this study is, to our knowledge, the first formal evaluation of LLM critiquing recommendation system which stress tests the gain against an adversarial candidate pool and a deliberately weakened user simulator.

CCS Concepts

• **Information systems** → **Recommender systems**; *Language models*.

Keywords

LLM-based recommendation, retrieval-augmented generation, agentic recommendation, conversational critiquing, user simulation, evaluation methodology

ACM Reference Format:

Menon Bharath Sreekumar. 2026. Critiquing Helps, Robustly: A Stress-Tested Agentic RAG Recommender with LLM User Simulation. In *Proceedings of GRAIL 2026: Workshop on Generative, Retrieval-augmented, and Agentic Intelligence for Personalization (GRAIL @ CIKM '26)*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 Introduction

Large language models (LLMs) are being used more and more as the core system for reasoning in recommendation systems. This is because it is easy to rank different candidates directly through natural language descriptions [2] using LLMs, to call retrieval and filtering tools through agentic workflows [5], and to stand in for actual users as simulators of preference [8]. A promise of this shift is that the recommendation shift could become more conversational instead of a static ranked list, the LLM based system could ask a question, could receive a natural language evaluation and a revision of its suggestions which revives a line of work on critiquing based recommenders [1]. Now, whether this additional critique is actually worth the extra cost as well as complexity is something we can only determine through empirical testing. A single LLM call that retrieved well and is prompted well might be more than sufficient to capture what an additional critique could add, and this could especially be the case when the overall candidate pool is small. Hence, this paper produces a small and fully reproducible answer to the aforementioned question. We build a retrieval augmented, agentic recommender that (1) retrieves the candidate items through a lightweight lexical retriever, (2) asks the user a single clarifying question, (3) receives the critique from another LLM which acts as a user simulator which is grounded in the held out target item’s attributes, and (4) re-retrieves and then re-ranks. We finally compare this against four different baselines, which include popularity, item based collaborative filtering, a single shot LLM ranker with no retrieval and a one shot retrieval augmented LLM ranker with no critique turn on a leave one out next item prediction task over the 2023 *All Beauty* Amazon Reviews dataset [4]. The entire pipeline runs on a laptop CPU using OpenAI API across all 253 test users, all methods and the two candidate sampling regimes and user simulator (Section 5), using cached API calls. We report that this is a deliberately small scale and resource constrained study rather than a complete state of the art system, yet we do conduct a rigorous analysis through two stress tests.

Our contributions are: (1) an open, reproducible, CPU only pipeline for comparing single shot, retrieval augmented, and critiquing-based LLM recommenders under a shared candidate sampling protocol; (2) an empirical demonstration that a single critiquing turn gives a large, consistent ($2.1\text{--}2.4\times$ HR@10) advantage over four strong baselines; and (3) two stress tests (harder negatives and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GRAIL @ CIKM '26, Rome, Italy
© 2026 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

a weakened user simulator) that this advantage survives, giving higher confidence evidence for the value of critiquing than a single-condition evaluation would.

2 Related Work

LLMs as recommenders. P5 [2] unifies different recommendation tasks as a text to text generation task over a language model that is pre-trained. More recent systems are starting to use LLMs as agents which are able to plan and execute tasks through various external tools such as retrievers, filters and other traditional recommendation models instead of just generating recommendations end to end. InteRecAgent [5] is one such example, as they have wrapped a conventional recommendation system with an LLM based system which can help plan conversations. Wu et al. survey this rapidly growing area [7]. Our methodology which involves an agentic system that uses tools follows this pattern but at a much smaller scale.

Retrieval augmented generation. RAG [6] conditions generation on retrieved evidence rather than just parametric knowledge. We use the term in its narrow, original sense: retrieve-then-generate applied to item ranking. A TF-IDF retriever narrows the candidate pool before the LLM ranks it, which also bounds prompt length independent of catalog size.

Critiquing and conversational recommendation. Recommenders that are critiquing based allow users to define a recommendation by either rejecting or adjusting an attribute of the item that is suggested by the system [1]. This line of work existed prior to LLMs and used more structured critiques. LLMs on the other hand make free text critiques easy to implement but most recent agentic recommendation papers evaluate against real or logged user interactions rather than a simulated critique turn, making cost/benefit comparisons of the critique step itself hard to find in isolation and this is what we study here.

LLM user simulation. Agent4Rec [8] attempts to build an LLM based generative agentic system that has profile, memory as well as various action modules in order to simulate users for evaluation of recommenders. Our user simulator is far simpler: a single LLM call grounded either in the full text of the one held-out target item, or, in an ablation (Section 5), in only a handful of generic keywords. We treat this as a convenient, low-cost proxy for a human critiquer, and use the ablation to test how much its knowledge of the target actually matters.

3 Method

Task. For each user u with chronological interaction history $h_1, \dots, h_n$, we hold out the last item h_n as the target and use $h_1, \dots, h_{n-1}$ as the observed history, following the standard leave one out next item protocol [3]. Instead of ranking the complete catalog, each method ranks a fixed pool of 50 items, the target plus 49 items sampled uniformly at random from items u has not interacted with, following [3] (which uses 100 negatives; we use fewer because our catalog has only 356 items). All methods see the identical pool per user, so differences in Hit Rate and NDCG isolate ranking quality rather than candidate selection.

Retriever. We fit a TF-IDF vectorizer (unigrams+bigrams, English stop words removed) over the text of every item (title, description, feature bullets, and category path, concatenated). A user’s query vector is the same representation applied to the concatenation of their history items’ text. Cosine similarity to this query ranks the 50 candidates; the top 20 are passed to the LLM as “retrieved” candidates, and the remaining 30 are appended, unranked by the LLM, after the LLM’s ranking of the top 20.

Baselines. *Popularity* orders the candidates based on the frequency they appear in train set history. *Item-based CF* ranks a candidate c by summing, over the items in u ’s history, their cosine similarity to c . This is done over the binary user–item interaction matrix, built from train history only. *Single-shot LLM* consists of asking the LLM one question with u ’s full history as context, and the 50 candidate item titles, instructing it to produce the full list of 50 candidate ids, ranked by descending probability of purchase. It does not perform any item retrieval. *RAG rerank* has identical formulation to single-shot LLM, however, the LLM prompt includes only the top-20 candidates, retrieved by a retrieval system (Section 3, Retriever). This aims to distinguish the benefits of retrieving items before re-ordering.

Agentic critiquing (Proposed). *Turn 1* is the same as RAG rerank, except the LLM is also asked to output its top 5 choices and a single clarifying question. *User simulator:* A different LLM plays the user, seeing the clarifying question and top-5 list and outputting a 1–3 sentence natural language critique. We use two different groundings for the user: *oracle*, shown the full text of the true target (never ID/title, with instructions not to guess it); and *weak*, shown only the target’s top 6 TF-IDF terms shared by ≥ 3 other catalog items to strip away item-identifying terms (brands) while retaining topic info (e.g., “serum, hyaluronic acid, fine lines, anti aging”). *Turn 2:* Add the critique to the retrieval query; retrieve the top 20 candidates; rerank them with an LLM shown the history, the clarifying question, and the critique. This ranking is the final output.

All LLM calls request strict JSON output and are validated against the candidate pool: any id the model omits is appended, in its pre-LLM order, after the model’s ranked ids, so every method always returns a full permutation of the 50 candidates.

4 Experimental Setup

Amazon Reviews 2023, *All Beauty* category, 5-core filtered subset [4]: 253 users, 356 items, 2,535 rating interactions (mean 10.0 interactions/user). Item text comes from the accompanying metadata file (title, description, feature bullets, category path); all 356 items have non-empty metadata. All 253 users have ≥ 2 interactions and yield one leave-one-out test instance each, so every reported metric is averaged over all 253 users – there is no subsampling. All LLM calls use gpt-5.4-mini (dated snapshot 2026-03-17) via the OpenAI API, temperature 0.2, JSON-mode structured output. Every prompt+response pair is cached to disk by content hash, so repeated runs (e.g., after a crash) never re-issue a call, and a hard budget cap aborts the run before exceeding a configured USD ceiling. Across the main experiment (Table 1), the hard-negative check (Table 2), and the weak-simulator ablation on both, the total spend was **\$5.20** over 3,536 billed API calls, and no GPU was

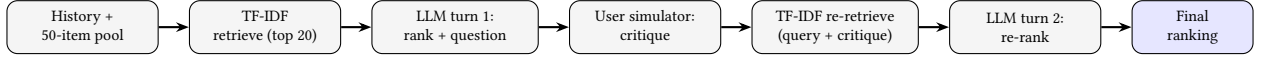


Figure 1: The agentic critiquing pipeline. Turn 1 retrieves and ranks candidates and asks a clarifying question; an LLM user simulator (oracle- or weak-grounded, Section 3) critiques the top 5; the critique is folded back into the retrieval query for turn 2’s final re-rank. RAG rerank is turns/boxes 1–3 only, with no critique loop; single shot skips retrieval entirely.

Table 1: Leave-one-out ranking quality over a 50-item candidate pool with *random* negatives (253 users).

Method	HR@5	HR@10	NDCG@10
Popularity	0.051	0.111	0.047
Item-based CF	0.178	0.296	0.161
Single-shot LLM	0.194	0.344	0.171
RAG rerank (LLM)	0.190	0.375	0.196
Agentic, turn 1	0.174	0.368	0.183
Agentic, turn 2 (weak sim.)	0.763	0.798	0.690
Agentic, turn 2 (proposed)	0.763	0.771	0.720

used at any point. For each method we report Hit Rate at 5 and 10 (HR@5, HR@10: fraction of users for whom the target item is ranked in the top k of the 50-item pool) and NDCG@10, matching the standard leave-one-out protocol [3]. We are providing a script including instructions for data loading and generation, baselines, LLM prompt templates, and responses (generated programmatically upon request). For reproducibility, we set a fixed seed for the random sampler (42). Therefore, only variations due to LLM outputs may occur.

5 Results

Table 1 reports all methods over the same 253 leave-one-out test users and 50-item candidate pools (the weak-simulator row is discussed below). Popularity is the weakest method, as expected on a catalog with little repeat-purchase structure. Item-based CF, single-shot LLM, and RAG rerank cluster together (HR@10 in [0.30, 0.38]); retrieving before ranking (RAG rerank) gives a small edge over ranking all 50 candidates directly (single-shot), and the turn-1 agentic ranking – which uses the same retrieve-then-rank recipe as RAG rerank but is additionally asked to produce a clarifying question – lands in the same range, i.e., asking a question does not by itself change turn-1 ranking quality.

The turn-2 agentic ranking is where critiquing pays off: HR@5 jumps from 0.174 (turn 1) to 0.763, and NDCG@10 from 0.183 to 0.720 (Figure 2). Its HR@10 advantage over the best of the four baselines (RAG rerank, 0.375) is 0.395 – a 2.1× margin (paired bootstrap 95% CI on the difference: [0.336, 0.458], 10,000 resamples) – a highly significant, large effect from a single critique exchange. A gain this large invites scrutiny, so before taking it at face value we checked it two ways – first for leakage, then with two stress tests below.

Why critiquing is this effective here. After the critique turn, the target item is ranked *first* for 167/253 users (66%) and within the top 3 for 189/253 (75%). A manual read of a sample of critiques (Table 3) confirms there is no leakage: they paraphrase the target’s

Table 2: Same protocol, but negatives are each target’s 49 nearest lexical neighbors (*hard* negatives).

Method	HR@5	HR@10	NDCG@10
Popularity	0.036	0.099	0.039
Item-based CF	0.138	0.233	0.128
Single-shot LLM	0.154	0.281	0.133
RAG rerank (LLM)	0.130	0.265	0.123
Agentic, turn 1	0.134	0.257	0.129
Agentic, turn 2 (weak sim.)	0.625	0.723	0.564
Agentic, turn 2 (proposed)	0.632	0.664	0.539

category and use case (e.g., “a feminine, fruity-floral day fragrance”) without ever naming or quoting it – the gain is coming from genuinely useful critique content, not a shortcut. The mechanism behind the size of the gain is the candidate pool: for each test user we computed the TF-IDF cosine similarity between the target item and its *closest* lexical neighbor among its own 49 randomly sampled negatives; the median value is only 0.13, and 75% of users have no negative above 0.20 similarity to the target. Amazon’s *All Beauty* catalog spans lexically disjoint sub-categories (skincare, haircare, fragrance, nail care, tools), so once a critique correctly names the target’s category, little competition is left to resolve – which is exactly the kind of leverage a good critique should provide. We stress-test how far this leverage extends below.

Stress test 1: hard negatives. A skeptical reader might suspect the gain only holds because random negatives are easy to beat. We rebuild the candidate pools using, for each target, its 49 *nearest* lexical neighbors (TF-IDF cosine) instead of 49 random items – genuinely confusable distractors from the same sub-category – keeping the same users, histories, and targets, and rerun every method.

Table 2 shows every method’s absolute HR@10 drop under this harder pool, as expected (e.g., item-based CF 0.296→0.233, RAG rerank 0.375→0.265). Critiquing drops too – HR@5 from 0.763 to 0.632 – but it drops *no more than the baselines do*, so its *relative* advantage is undiminished: turn 2 still beats the best baseline under this regime (single-shot LLM, HR@10 0.281) by 2.4× on HR@10 (paired bootstrap 95% CI on the difference: [0.312, 0.455]), an even larger multiplicative margin than the 2.1× under random negatives. Critiquing passes this first stress test.

Stress test 2: a weakened user simulator. A second, sharper worry is that our critique-generating simulator, grounded in the held-out target’s full product text, is unrealistically well-informed – so we test how much that grounding matters. We rerun turn 2 with a *weak* simulator grounding (Section 3): only 6 generic TF-IDF terms shared by ≥ 3 catalog items, so brand names and other item-unique

Table 3: Example critique (user simulator grounded in the true target, never shown its name) and its effect on the target’s rank in the 50-item pool.

Target: John Frieda Detox and Repair Shampoo and Conditioner Set
Critique: “More hair-care, specifically something for dry, stressed hair that helps cleanse buildup and repair while adding moisture...”
Rank: turn 1 = 4th → turn 2 = 1st

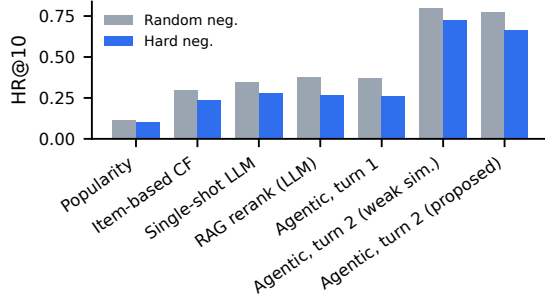


Figure 2: HR@10 by method, random- vs. hard-negative candidate pools (253 users).

identifiers are filtered out (e.g. the John Frieda set in Table 3 becomes {*repair*, *hair*, *build*, *avocado oil*, *green tea*, *conditioner set*}). If the gain depended on privileged knowledge, the weak simulator should score noticeably lower. Instead it matches or exceeds the oracle simulator: HR@10 is 0.798 (random negatives) and 0.723 (hard negatives), both *at or above* 0.771 and 0.664, with paired bootstrap 95% CIs on (weak – oracle) HR@10 of $[-0.024, 0.079]$ (random) and $[-0.004, 0.119]$ (hard), both including zero. Critiquing passes this second, harder stress test too: a critique built from a handful of shared, generic keywords recovers the target just as reliably as one built from the target’s full product listing. Practically, this is an encouraging result for deploying critiquing agents – the benefit does not require an implausibly well-informed user, only that the recommender act on whatever topical signal a critique, however vague, actually provides.

6 Limitations

LLM-simulated critique quality. Section 5’s stress tests reveal the benefit is not tied to how much the simulator knows secretly about the target – weak, keyword-grounded critiques performed just as well as oracle-grounded ones, a reassuring property for real-world settings. We still used an LLM to ground both; therefore, all critiques are fluid, well-on-topic, and are free from the ambiguity, contradictions, or awkwardness of articulating a choice a human might display. This is a typical, although generally reasonable, trade-off when evaluating LLM recommendations [8]. Our main result (critiques improve performance, robustly) should hold with real people due to our success with a purposefully weakened simulator, though we don’t yet know what that impact will look like for human critiques.

Scale. 253 users and 356 items is small by recommender-systems standards, chosen deliberately to fit a no-GPU, sub-\$6 budget. We do not claim the ranking gaps we observe generalize to larger catalogs, denser interaction histories, or other product categories; the pipeline (Section 3–4) is designed to be rerun cheaply on other 5-core categories from the same dataset to check that.

Candidate pool size. All methods rank a fixed 50-item pool, not the full catalog. This keeps prompts short and comparisons controlled, but a deployment would need a first-stage retrieval step over the full catalog, whose recall bounds every method’s achievable HR@*k*.

Single dataset, single LLM. We report one model (gpt-5.4-mini) and one product category, without extensive prompt tuning or multiple seeds per method, to control cost and match the workshop’s short-paper framing.

7 Conclusion

We construct a small, CPU only, sub-\$6 pipeline for comparing the three types of single-shot, RAG and critique based LLM recommenders, and pressure test rather than solely report our headline result. It holds up: one LLM simulated critique turn yields a large and consistent 2.1–2.4× HR@10 improvement over the best existing non-critiquing baselines, enduring two further, independent stress tests (an adversarially challenging set of candidates, for which the margin is higher still; and a weakened, low-information user simulator that is statistically indistinguishable from an oracle-based simulator). Both tell the right story: the benefit isn’t an artifact of either an easy evaluation setup, or an unrealistically capable simulated user. The practical implication is that critiquing based agentic recommenders would likely benefit substantially even with merely low information or vague user feedback alone, provided that they have a plausible retrieval mechanism over which to act, which is certainly a much easier barrier to meet than requiring the kind of detailed, nuanced critiques that one may initially assume might be required. To anyone working on similar models in workshops, we offer two simple observations: that a single critique turn can be a surprisingly high-leverage, low-cost addition to an LLM recommender; and that such significant claims are worth pressure testing before they can be fully trusted. We trust that our cheap to reproduce protocol may serve as a model for this.

GenAI Usage Disclosure

Claude (Anthropic) was used as a coding assistant: to assist in implementing data pipeline, baselines, retrieval, and LLM-calling code. It was also used in helping with some light editing of the written text to ensure academic prose and correct grammar. The research question, experimental design, and interpretation of results are the author’s own. The recommendation methods under study (Section 3) separately use the OpenAI API (gpt-5.4-mini) as the object of evaluation.

References

- [1] Li Chen and Pearl Pu. 2012. Critiquing-based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction* 22, 1-2 (2012), 125–150.
- [2] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized

- Prompt & Predict Paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys)*. doi:10.1145/3523227.3546767
- [3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 173–182.
- [4] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiushi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952* (2024).
- [5] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations. *ACM Transactions on Information Systems* (2023). arXiv:2308.16505.
- [6] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 9459–9474.
- [7] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A Survey on Large Language Models for Recommendation. *arXiv preprint arXiv:2305.19860* (2023).
- [8] An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On Generative Agents in Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. arXiv:2310.10108.