Fact vs. Fiction: Are the Reportedly "Magical" LLM-Based Recommenders Reproducible?

Shirin Tahmasebi[1], Narjes Nikzad[2], Amir H. Payberah[1], Meysam Asgari-chenaghlu[3], and Mihhail Matskin[1]

[1] Royal Institute of Technology, Sweden {shirint, payberah, misha}@kth.se
[2] Gisma University of Applied Sciences, Germany narjes.khasmakhi@gisma.com
[3] Zendesk, Germany meysam.asgarichenaghlou@zendesk.com

Abstract. Reproducibility is a cornerstone of scientific progress, allowing researchers to validate findings and build upon previous work. While reproducibility has been an important issue in traditional recommender systems, the rise of Large Language Model (LLM)-based recommendation systems introduces new challenges, particularly in top-N recommendation tasks. In this study, we investigate the reproducibility of state-of-the-art LLM-based recommendation systems. We categorize key factors affecting the reproducibility of recommendation performance into four groups: *code*, *data*, *methodological details*, and *evaluation*. Our findings highlight significant performance fluctuations based on these factors, emphasizing the need for these factors to be clearly documented and considered during evaluations. To enhance reproducibility, we propose LLMReClarify, a comprehensive set of guidelines adapted from the NeurIPS reproducibility checklist, tailored specifically for LLM-based recommendation systems.

Keywords: Large Language Models · Recommendation Systems · Top-N Recommendation · Reproducibility

1 Introduction

Reproducibility is a cornerstone of scientific progress [1,23], ensuring that findings can be consistently replicated and validated by other researchers. It allows researchers to build on previous work, verify findings, and push the boundaries of knowledge. Without reproducibility, reported findings may be subject to doubt. To support reproducibility, various guidelines and checklists have been developed to help researchers follow best practices, outlining common pitfalls and ways to avoid them. Academic venues like NeurIPS [18] and ACM Transactions on Recommender Systems (TORS) [22] now require adherence to these checklists, enforcing stricter standards to ensure that research can be reliably reproduced.

Recently, several research domains have faced an ongoing reproducibility crisis [23], and Recommendation Systems (RecSys) [3,4] are no exception. Studies addressing this issue in RecSys highlight factors such as the lack of standardized evaluation protocols, the unavailability of code and data, and the variability

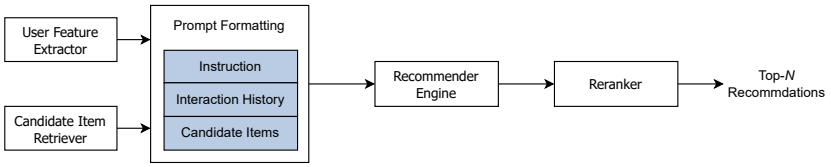


Fig. 1: Architectural Modules of the LLM-based Recommendation Systems

in experimental setups [3,4]. These studies suggest that achieving reproducible results requires researchers to report more details about their methods.

Apart from that, in recent years, Large Language Models (LLMs) have gained popularity in RecSys due to their remarkable reasoning capabilities [11,12]. A typical architecture of an LLM-based RecSys is comprised of a subset of the following key modules [12] (as depicted in Figure 1):

- *User Feature Extractor*, to extract features from a user's past interactions.
- *Candidate Item Retriever*, to select a limited number of candidate items from which the final recommendation will be chosen.
- *Prompt Formatting*, to consolidate the input information and format it appropriately for the next module.
- *Recommendation Engine*, to recommend items from the candidate list.
- *Re-ranker*, to re-rank recommendations to prioritize the most relevant ones.

Research has shown that LLMs can be leveraged within any of the architectural modules illustrated in Figure 1 [11,12]. However, while integrating LLMs offers many advantages, it also introduces new challenges. These challenges include the complexities of fine-tuning LLMs, hallucinations, and the limitations of context windows. These factors all affect the performance of the RecSys. Given the growing prominence of LLM-based RecSys, it is essential to assess their reproducibility, as has been done for other non-LLM-based RecSys. Furthermore, since LLMs can be integrated into various modules of the LLM-based RecSys (Figure 1), it is important to consider the reproducibility of each of these modules. That is because incorporating LLMs into each of these modules can introduce unique reproducibility challenges that must be investigated.

Scope. In this paper, we aim to investigate the reproducibility of several recently-proposed LLM-based RecSys, with a particular emphasis on top-N recommendation tasks. In such tasks, the system selects N items as recommendations for a given user. We focus on assessing the reproducibility of recommendation results when LLMs are incorporated into any of the modules depicted in Figure 1.

Objective. By examining the various factors that can potentially affect the recommendation performance, we seek to identify the specific details that should be reported to enhance reproducibility in designing new LLM-based RecSys. To achieve this, we introduce a framework called LLMReClarify. This framework serves as a comprehensive guide, outlining the essential components and practices necessary to improve reproducibility. Through this work, we aim to fill the

gap in reproducibility research for LLM-based RecSys by providing guidelines that promote transparency and reliability in their development.

Contributions. This paper presents several key contributions, including:
- Analyzing the reproducibility challenges specific to LLM-based RecSys, identifying critical factors such as *code availability*, *data handling*, *methodological details*, and *evaluation protocols*.
- Proposing LLMReClarify, a detailed guideline adapted from the NeurIPS reproducibility checklist [18], tailored specifically for LLM-based RecSys.
- Validating the importance of the identified factors through extensive experiments, revealing significant performance fluctuations based on these factors.

Outline. The structure is as follows: Section 2 reviews related work and identifies research gaps. Section 3 introduces LLMReClarify and key reproducibility factors in LLM-based RecSys. Sections 4 and 5 present our experimental findings, followed by a discussion in Section 6 and the potential future works and conclusions in Sections 7 and 8.

2 Related Work

In this section, we first discuss the current efforts on reproducibility in RecSys in Section 2.1. Next, we review LLM-based RecSys in Section 2.2 and conclude with research gaps in Section 2.3.

2.1 Reproducibility in Recommendation Systems

Reproducibility issues in RecSys have attracted significant attention, with numerous studies addressing this challenge. This section reviews their findings. In [3,4], the authors highlight reproducibility concerns in RecSys, especially with neural network-based methods. After attempting to reproduce results from 26 papers (2015-2018), they found that many neural approaches do not consistently outperform optimized baselines, with only 12 studies being reproducible with reasonable effort. Issues include unavailable code, inconsistent data splitting, and unoptimized baselines, all of which motivated them to advocate for rigorous research practices, improved baseline optimization, and practical validation.

In [1], the authors emphasize the importance of reproducibility for verifying results in RecSys, identifying key challenges like inconsistent data-splitting and algorithm implementation. To address these, they propose a comprehensive framework covering dataset collection, data splitting, algorithm implementation, candidate filtering, evaluation protocols, and statistical testing.

Reproducibility challenges in *transformer-based* RecSys have also been studies. For instance, [21] highlights inconsistencies in BERT4Rec's [24] reported performance over SASRec [9,24], revealing deviations from original configurations for reproducibility. They introduce a new implementation using the HuggingFace Transformers library, successfully replicating results on most datasets, aiming to improve BERT4Rec's reproducibility and reliability with new implementations and thorough documentation.

Building on this, [15] focuses on reproducibility challenges unique to LLM-based RecSys by introducing LaikaLLM, a framework designed to standardize

processes such as training, fine-tuning, and evaluation. This framework facilitates consistent experimentation and enables researchers to replicate results effectively. Using LaikaLLM, the authors evaluate the reproducibility of the P5 paradigm [5], addressing issues like inconsistent repository organization, insufficient documentation, and variability in results. Moreover, they demonstrate how prompt design and backbone LLM selection significantly influence performance, emphasizing the need for standardized approaches in LLM-based RecSys.

2.2 LLM-based Recommendation Systems

LLMs have demonstrated strong potential in RecSys, leveraging their reasoning abilities for personalized and context-aware recommendations. NIR [25] generates candidate items via user and item filtering, while RecMind [26] uses memory networks and dynamic prompting to improve recommendation relevance in multi-turn dialogues. LlamaRec [27] employs a two-stage approach: an external model selects candidates, followed by a prompt-based re-ranker to refine recommendations, fine-tuned on Amazon Beauty [7,16], Amazon Games [7,16], and MovieLens [6]. GenRec [8], focused on sequential recommendations, relies on fine-tuning and is tested on Amazon Toys and MovieLens. LLMRec [14] benchmarks tasks like rating prediction and review summarization, with gains from supervised fine-tuning. PALR [2] combines user profiling, item retrieval, and ranking for personalized recommendations on Amazon Beauty and MovieLens.

2.3 Research Gap

Considerable efforts have been made to address reproducibility in non-LLM-based RecSys. However, integrating LLMs introduces unique complexities absent in non-LLM-based RecSys, necessitating a focused examination of reproducibility in this specific context. Examples of these complexities include:

- **Hallucinations:** LLMs may recommend non-existent items, such as recommending a movie name that does not exist. Reporting the approach for managing these hallucinations is crucial for having a reproducible system. For example, one potential approach to address this issue is to provide the LLM with a predefined list of candidate items from which to choose. However, this introduces additional considerations, such as the number and ordering of candidate items, which greatly affect system performance.
- **Limited Context Window:** Due to limitations in input size, including a user's entire interaction history in a prompt is often infeasible. Summarizing or selecting key interactions is crucial, and clearly reporting this strategy is essential, as it can heavily affect performance.

These are just a few examples of the unique challenges faced by LLM-based RecSys, which are not present in non-LLM-based systems. This highlights a research gap and the need for a reproducibility checklist tailored specifically to LLM-based RecSys. With LLMReClarify, we aim to address this gap by providing comprehensive guidelines. Notably, our focus is on LLM-based RecSys that leverage *item features*—such as titles, descriptions, or categories—rather than solely relying on item IDs, as is common in models like BERT4Rec [24].

3 LLMReClarify

In response to the challenges highlighted in Section 2.3, we introduce LLMReClarify, a framework to enhance reproducibility in LLM-based RecSys, particularly for top-N recommendation tasks. Built upon the NeurIPS reproducibility checklist for machine learning systems [17,18], LLMReClarify categorizes the reproducibility metrics into four groups: code-related factors (**R1**), data-related factors (**R2**), methodological details (**R3**), and evaluation details (**R4**), each of which is detailed in the following subsections.

R1: Code-related Factors. Ensuring reproducibility in LLM-based RecSys requires public access to the following two critical components, with detailed setup and execution instructions to enable effective replication and validation:

- **R1.1 Fine-tuning Code:** Code for adapting the LLM to different tasks.
- **R1.2 Evaluation Code:** Scripts for evaluating the model performance.

R2: Data-related Factors. The data-related factors can be divided into several key items, including:

- **R2.1 Data Availability:** The leveraged dataset should be publicly accessible.
- **R2.2 Data Splitting Strategy:** The method for dividing data into training, validation, and test sets should be clearly defined. Common approaches include *holding out* data by time cutoffs, *leave-one-out* (LOO), which reserves each user's latest interaction for testing and the second latest for validation, and *user-based* splits that assign earlier users to training, a middle group to validation, and the latest to testing.
- **R2.3 Data Preprocessing Approaches:** Documenting data preprocessing steps, including any filtering policies, is crucial. Criteria and rationale for excluding records should be clearly stated.

R3: Methodological Details. The methodological details can be divided into several key factors, including:

- **R3.1 Type of LLM:** The leveraged LLM variant (e.g., GPT-3.5 or Llama).
- **R3.2 Fine-tuning Strategy:** The fine-tuning approach used, if applicable.
- **R3.3 Prompt Formatting:** Given its critical role in interpreting user intent, it is essential to document the prompt structure, task instructions, the format of user-item interaction history, and the number of historical items provided as input to the recommendation engine.
- **R3.4 Handling Hallucination:** This refers to the mechanisms in place, if any, to address the generation of irrelevant or non-existent content by LLMs. For example, one potential approach is to constrain the output space by integrating external knowledge sources, such as knowledge graphs, to ensure that recommendations are grounded in valid data.
- **R3.5 Item Retriever:** This component generates a pool of candidate items from which recommendations will be made. Key elements in this component which may impact the model performance include: (1) the retrieval strategy (e.g., random selection, the most or least similar items to ground truth), (2) the number of retrieved candidates, and (3) the position of the ground truth in the candidate list.

R4: Evaluation Details. Accurate and transparent reporting of evaluation details is crucial for reproducibility and reliability in LLM-based RecSys. Specific assumptions or record exclusions before evaluation must be clearly documented, as they can significantly impact results. Applying consistent assumptions and criteria across baselines is essential for fair comparisons, ensuring accurate performance assessments and preventing misleading conclusions.

4 Experimental Evaluation

In this section, we describe our experimental setup and methodology. We first provide an overview of the dataset used in our experiments in Section 4.1, followed by a detailed description of the baselines for our study in Section 4.3.

4.1 Datasets

We conducted our experiments using the MovieLens dataset [6] across all baselines. This dataset provides versions of various sizes, including 100K, 1M, and 20M ratings, with users rating movies on a scale of 1 to 5. To ensure consistency, we used the specific version referenced in each original baseline paper.

4.2 Metrics

To assess performance, we use Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG). HR@k checks if the correct item appears in the top-k items of the recommendation list, calculated as the ratio of cases where the correct item is included within the top-k to the total cases. NDCG@k evaluates ranking quality by considering the position of relevant items in the top-k recommendations. It is calculated by normalizing the Discounted Cumulative Gain (DCG) against the ideal DCG, where DCG gives higher scores to relevant items ranked near the top.

4.3 Baseline

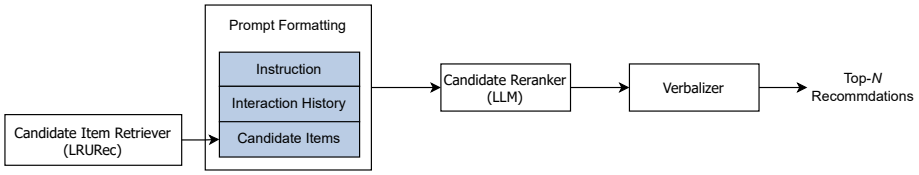
As mentioned in Section 2.2, numerous LLM-based top-N RecSys exist in the literature. In Table 1, we summarize the reproducibility level of each system, where each row corresponds to a specific system and each column represents a reproducibility factor listed in Section 3. Among these systems, we excluded those listed in rows 1 to 4 of Table 1 due to the lack of public code, which is a fundamental requirement for reproducibility. Of the remaining ones, we initially attempted to reproduce LLMRec [13,14]. However, despite using their publicly available code and preparing the data as outlined in their paper, we were unable to replicate the expected behavior of their model due to the high frequency of hallucinations, which made further experiments impractical. Therefore, we focused on LlamaRec [27], GenRec [8], and NIR [25] as our baselines.

This section is divided into subsections, each dedicated to a specific benchmark and offering a detailed description of that benchmark.

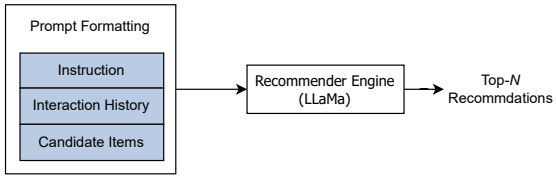
LlamaRec. LlamaRec [27] introduces a two-stage Llama-based sequential RecSys, depicted in Figure 2a. The first stage, or *retrieval* stage, uses an external, off-the-shelf RecSys–options include SASRec [9], BERT4Rec [24], and LRURec [28]—to initially retrieve a set of 20 candidate items. Following this, in the second stage, or *re-ranker* stage, an input prompt is created which comprises of three parts:

Table 1: Reproducibility Level of the Baselines

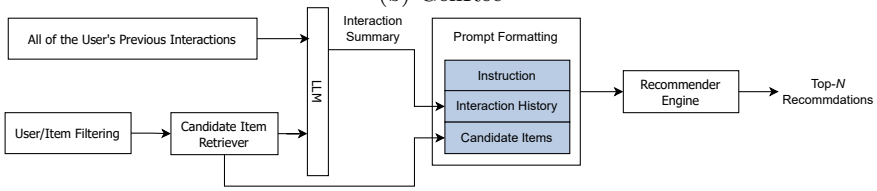
	R1: Code		R2: Data			R3: Methodological Details					R4: Evaluation Details
	R1.1	R1.2	R2.1	R2.2	R2.3	R3.1	R3.2	R3.3	R3.4	R3.5	
GPT4Rec [10]	✗	✗	✓	✓	✓	✓	✓	✓	✗	-	✓
PALR [2]	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓
RecMind [26]	✗	✗	✓	✓	✗	✓	-	✗	✗	✗	✓
InstructRec [29]	✗	✗	✓	✓	✓	✓	-	✓	✓	✓	✓
LLMRec [13,14]	✓	✓	✓	✗	✗	✓	✗	✓	✗	-	✓
LlamaRec [27]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
GenRec [8]	✓	✓	✓	✗	✓	✓	✓	✗	✗	-	✓
NIR [25]	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓



(a) LlamaRec



(b) GenRec



(c) NIR

Fig. 2: The Architectural Components of the Baselines

(1) an instruction, (2) a list of interaction history–as many items as possible up to the context window limit, and (3) the 20 candidates. The re-ranking process then begins, utilizing Llama2 to sort the candidate items. Unlike typical use, this stage does not generate tokens autoregressively. Instead, it employs a verbalizer that converts output scores from the LLM's head into ranking scores for the candidate items, effectively addressing the common hallucination issues. The output from this stage, a re-ranked list of items, constitutes the final recommendations provided by LlamaRec. Both the external recommender used in the retrieval stage and the Llama model employed in the re-ranker stage require fine-tuning. The effectiveness of LlamaRec's approach has been evaluated us-

ing three real-world datasets: Amazon Beauty [7,16], Amazon Games [7,16], and MovieLens [6], using LOO splitting strategy.

GenRec. GenRec [8] is an LLM-based sequential RecSys, as shown in Figure 2b. The system uses Llama2, which has been fine-tuned to enhance its understanding of user preferences and interaction patterns. Its main functionality is to predict the next item in a sequence based on a structured input prompt comprising (1) an instruction and (2) a list of the user's interaction history. However, the number of interaction items included in the prompt is not specified in the original paper. Unlike many systems, GenRec lacks a retriever or re-ranker module, relying solely on the fine-tuned capabilities of Llama2 to generate recommendations. Additionally, GenRec does not address the issue of hallucination, which could impact its overall performance. In the original paper, the effectiveness of GenRec was evaluated using Amazon Toys [7,16] and MovieLens [6].

NIR. NIR [25] introduces a zero-shot approach to RecSys using LLMs, enabling models like GPT-3 to generate recommendations without training on specific datasets. As illustrated in Figure 2c, NIR follows a three-step prompting strategy: (1) summarizing the user's preferences based on their interaction history, (2) selecting and ranking representative items from the user's history, and (3) recommending items from a pre-constructed candidate set, which is generated using user or item filtering techniques. User filtering selects items liked by similar users, while item filtering finds items similar to those in the user's history. The ground truth item (the correct next item) may or may not be present in this candidate list. These filtering techniques help reduce the recommendation space, enhancing the system's efficiency. The iterative process of preference summarization, item selection, and recommendation forms an effective dialogue between the RecSys and the LLM. The effectiveness of this approach is evaluated on MovieLens [6].

5 Reproducibility Issues and Results

This section is divided into subsections, each dedicated to a specific benchmark. For each, we first discuss its reproducibility issues, followed by the experiments conducted and their results, considering the identified reproducibility issues. The code for all the experiments is available on a public GitHub[4] repository.

5.1 LlamaRec

The architectural details of LlamaRec are presented in Section 4.3. Here, we first address the reproducibility issues and then outline the experiments carried out to further investigate these issues.

Reproducibility Issues. LlamaRec's reproducibility is affected by certain assumptions, notably the ground truth's position within the candidate items. This factor, which significantly affects the results, was not considered in their evaluations, violating **R3.5**. To assess the impact of this factor on the final performance, we conducted experiments injecting the ground truth at different positions (from 1st to 20th) in the candidate list. However, since running the experiments for each position of the ground truth can be computationally expensive,

[4] `https://github.com/ShirinTahmasebi/LLMReClarify`

we also conducted an additional experiment where the ground truth position is randomly selected. In this approach, the position is determined according to a uniform random distribution. Thus, in this second approach, instead of running the experiments 20 times for all users, we only run them once per user. While this method is less computationally intensive, it comes at the cost of potentially losing some accuracy.

Additionally, the original paper includes an evaluation scenario where only cases where the ground truth is among the top 20 candidate items are considered. While this assumption is clearly documented, it is not a standard approach across other systems, such as PALR, GPT4Rec, RecMind, BERT, and SASRec. This difference can lead to potential inconsistencies when comparing results.

Experiments and Results. To examine whether the ground truth's position among the top 20 candidates affects the reproducibility of results, we conducted a series of experiments. Given that this factor significantly influences performance, we sought to understand its impact on result consistency. For each experiment, we selected the top 19 candidate items (excluding the ground truth) and placed the ground truth at positions from 1st to 20th. To ensure reliability, each experiment was run five times, and average results were reported.

Figure 3 presents the results of these experiments in a line chart, with four lines representing NDCG@10, HR@10, NDCG@5, and HR@5 metrics. The X-axis shows the position of the ground truth within the candidate items, with each line composed of 20 points corresponding to the metric value when the ground truth is placed at each position. A horizontal dashed line indicates the metric reported in the original paper.

As shown in Figure 3, the position of the ground truth significantly impacts performance, confirming our claim that considering this factor is crucial for reproducibility. The reported performance in the paper is reproducible only under specific circumstances when the ground truth is in a particular position in the candidate list. However, by changing the position, the performance can change significantly, which highlights the effects of the ground truth position on performance. Considering this factor enables more thorough analysis, leading to better insights and more reliable comparisons across different systems.

As outlined at the beginning of this section, we proposed an alternative approach to evaluate the impact of ground truth position by randomly selecting it from a uniform distribution for each user, rather than testing all possible positions. This method requires only a single iteration over the user list, reducing computational costs and enhancing scalability for larger candidate sets, though with some accuracy trade-offs. Figure 4 illustrates these results: the *random* bars show the scenario with randomly determined ground truth positions, while the *average* bars reflect the averaged results from Figure 3. These findings suggest that random selection provides a reasonable approximation for HR@5 and NDCG@5 but is less accurate for HR@10 and NDCG@10.

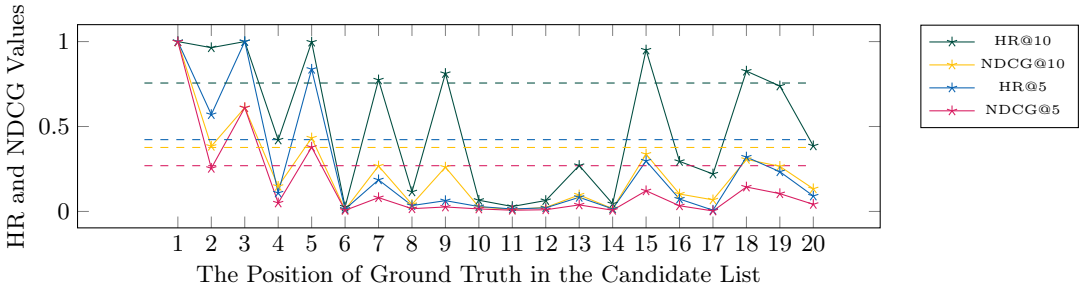


Fig. 3: LlamaRec: The Impact of Ground Truth Position on NDCG and HR

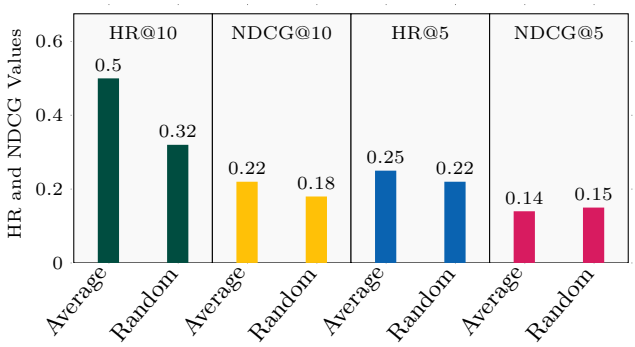


Fig. 4: LlamaRec: Randomly Selecting the Ground Truth Position from a Uniform Distribution

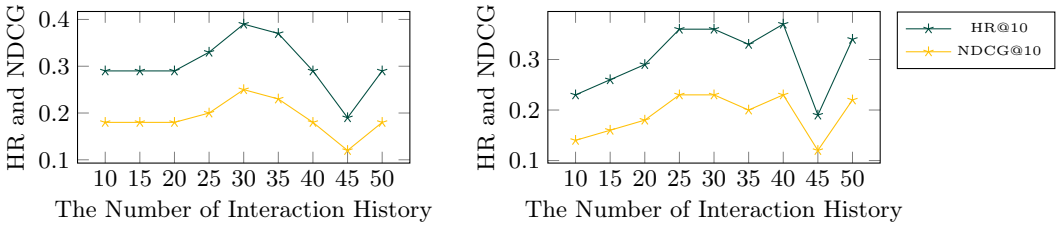
5.2 GenRec

Following the architectural overview of GenRec in Section 4.3, this section begins with exploring its reproducibility issues, followed by a thorough analysis of the experiments designed to evaluate these issues.

Reproducibility Issues. Several issues cast doubt on its reproducibility:

- **Dataset Splitting Strategy:** The dataset splitting strategy is neither mentioned in the paper nor included in the GitHub repository, violating **R2.3**.
- **Prompt Format:** The number of interaction history items included in the prompt is not clearly specified, neither in the paper nor in the GitHub code, violating **R3.3**. However, we hypothesize that this factor may significantly impact performance.
- **Hallucination Handling:** Although the model has been fine-tuned to reduce hallucination, this issue is not completely resolved, violating **R3.4**. The output remains susceptible to repetition and hallucination, with no specific strategy for handling it.

Experiments and Results. To evaluate GenRec's reproducibility, we designed a series of experiments to assess how changing the number of interaction history items affects performance. Due to missing details in the original paper, we assumed that the dataset splitting strategy is LOO. Additionally, we removed all hallucinated and repeated items from the results without attempting to fix them.



(a) Low Temperature (= 0.6) (b) High Temperature (= 1)

Fig. 5: GenRec: The Impact of the Number of Included Interaction History on NDCG@10, HR@10, NDCG@5, and HR@5 Metrics

We tested various configurations by adjusting the number of interaction items in the prompt, using 10, 15, 20, 25, 30, 35, 40, 45, and 50 items. Context window limitations prevented us from including more. Each experiment was repeated five times, and average performance was reported to ensure reliability.

The results of these experiments are depicted in Figures 5a and 5b. Each figure represents the results of experiments conducted using a specific temperature parameter[5]. Figure 5a shows results with the highest temperature of 1, indicating maximum creativity and randomness in the output, while Figure 5b shows results with a lower temperature of 0.6, producing more stable outputs. In both figures, the X-axis represents the maximum number of interaction items included in the history, while the Y-axis represents the performance values.

As shown in these figures, the number of items in the interaction history significantly impacts the model's performance. In both high and low temperatures, we observe that increasing the number of items in interaction history up to a specific point improves performance. However, beyond that point, further increasing the number of items has a negative and random effect on performance. It is worth mentioning that this point, where the randomness starts, differs between the two temperatures. With a higher temperature, the randomness starts earlier, indicating that at higher temperatures, the performance is more volatile.

5.3 NIR

Given the NIR's architecture in Section 4.3, this section first addresses its reproducibility issues, followed by experiments conducted to explore them further.

Reproducibility Issues. As shown in Table 1, NIR meets all the criteria in LLMReClarify. Specifically, the repository and datasets are publicly available, and all data-related factors are clearly reported, satisfying **R1** and **R2**. Also, methodologically, the paper describes all relevant aspects, including the LLM type and prompt formatting. To address hallucination, the authors provide a list of candidates to the LLM to select recommended items. They explained the construction of the list and evaluated the impact of varying the number of candidate items on recommendation performance. In summary, all factors (**R1** to

[5] Temperature parameter in LLMs controls output randomness: higher values increase creativity, while lower values yield more deterministic responses.

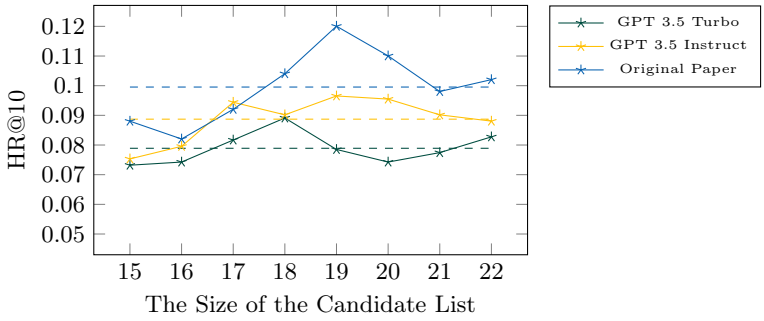


Fig. 6: NIR: The Impact of the Size of Candidate List on HR@10

R4) are thoroughly documented, making NIR a strong example of how following the guidelines in LLMReClarify can facilitate reproducibility.

Experiments and Results. To evaluate the reproducibility of NIR, we replicated the experiments from the original paper, where the size of the candidate list varied between 15 and 22, and HR@10 was calculated for each case. A key challenge was the deprecation of GPT-3.5 Davinci 3 [20], originally used in NIR, which we addressed by substituting GPT-3.5 Turbo and GPT-3.5 Instruct [19].

Figure 6 presents the results for the substituted models, and the original paper's results. We included horizontal dashed lines to represent the average for each model variant. The deviation between the average results in the original paper and those from our experiments ranges between 0.01 and 0.02. This deviation is minimal, especially considering that we had to use different GPT-3.5 variants due to the deprecation of the original model used in the paper. This suggests that, by reporting the details outlined in LLMReClarify, the system's results are reproducible.

6 Discussion

As mentioned in Section 4.3, we focused on LlamaRec, GenRec, and NIR. For each system, we first identified reproducibility issues based on the factors outlined in Section 3, then conducted extensive experiments to investigate further.

Our experiments with LlamaRec highlighted that the ground truth position within the candidate list significantly impacts performance. The metrics reported in the original paper are reproducible only under specific circumstances. By varying the ground truth position, we observed substantial fluctuations, emphasizing the need to consider this factor in evaluations for fair and thorough assessments.

For GenRec, we found that both the number of items in the interaction history and the output randomness (temperature) heavily affect performance. Up to a certain point, adding interaction items improved results, but beyond that, additional items led to random and negative effects. This threshold varies by temperature setting, with higher temperatures causing earlier volatility. This demonstrates the importance of documenting these factors to ensure reproducibility and enable accurate system comparisons.

In NIR, we observed minor variations from the original paper's results. However, this difference was negligible, especially given our use of alternative GPT-3.5 variants due to deprecation of the version used in the paper. This highlights that by documenting key details outlined in LLMReClarify, reproducibility can be maintained even with slight changes in the environment or model versions.

Takeaways. According to our experiments, our main takeaway is that different factors mentioned in Section 3 should be clearly documented to ensure reproducibility and fair comparisons. Specifically, providing detailed explanation the code factors (**R1**), data-related factors (**R2**), methodological details (**R3**), and evaluation assumptions (**R4**) are essential for enhancing the reproducibility of LLM-based RecSys for top-N recommendation task.

7 Future Work

This work primarily focuses on top-N recommendation tasks, which are crucial for many real-world applications. However, other recommendation tasks, such as rating prediction, explanation generation, and session-based recommendations, also present unique reproducibility challenges. For example, explanation generation involves subjective evaluations, while rating prediction demands finer-grained metric standardization. Future research could address these gaps by extending the analysis to other recommendation tasks and their specific reproducibility challenges.

Additionally, LLMReClarify examines a select set of LLM-based RecSys, which reflect the current state of the art. However, architectures relying on item embeddings, such as BERT4Rec [24], or systems using hybrid filtering techniques, remain unexplored. Expanding the scope to include such systems could enhance the generalizability of our findings and refine the checklist.

8 Conclusion

Our investigation into the reproducibility of LLM-based RecSys reveals critical factors affecting performance, such as evaluation assumptions, dataset splitting strategies, and prompt formatting details. The LLMReClarify framework offers a structured approach to enhance reproducibility through comprehensive reporting of these factors. Following these guidelines enables fair comparisons and reliable results in developing and evaluating LLM-based RecSys.

Acknowledgement. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

1. Bellogín, A., Said, A.: Improving accountability in recommender systems research through reproducibility. User Modeling and User-Adapted Interaction **31**(5), 941–977 (2021)
2. Chen, Z.: Palr: Personalization-aware llms for recommendation. arXiv preprint arXiv:2305.07622 pp. 1–5 (2023)

3. Ferrari Dacrema, M., Boglio, S., Cremonesi, P., Jannach, D.: A troubling analysis of reproducibility and progress in recommender systems research. ACM Transactions on Information Systems (TOIS) **39**(2), 1–49 (2021)
4. Ferrari Dacrema, M., Cremonesi, P., Jannach, D.: Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM conference on recommender systems. pp. 101–109 (2019)
5. Geng, S., Liu, S., Fu, Z., Ge, Y., Zhang, Y.: Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In: Proceedings of the 16th ACM Conference on Recommender Systems. pp. 299–315 (2022)
6. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) **5**(4), 1–19 (2015)
7. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: proceedings of the 25th international conference on world wide web. pp. 507–517 (2016)
8. Ji, J., Li, Z., Xu, S., Hua, W., Ge, Y., Tan, J., Zhang, Y.: Genrec: Large language model for generative recommendation. ECIR pp. 1–4 (2024)
9. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE international conference on data mining (ICDM). pp. 197–206. IEEE (2018)
10. Li, J., Zhang, W., Wang, T., Xiong, G., Lu, A., Medioni, G.: Gpt4rec: A generative framework for personalized recommendation and user interests interpretation. arXiv preprint arXiv:2304.03879 (2023)
11. Li, L., Zhang, Y., Liu, D., Chen, L.: Large language models for generative recommendation: A survey and visionary discussions. In: Proceedings of LREC-COLING. pp. 10146–10159. ELRA and ICCL, Torino, Italia (2024)
12. Lin, J., Dai, X., Xi, Y., Liu, W., Chen, B., Zhang, H., Liu, Y., Wu, C., Li, X., Zhu, C., Guo, H., Yu, Y., Tang, R., Zhang, W.: How can recommender systems benefit from large language models: A survey. ACM Transactions on Information Systems (2024)
13. Liu, J., Liu, C., Zhou, P., Lv, R., Zhou, K., Zhang, Y.: Is chatgpt a good recommender? a preliminary study. In: CIKM 2023 GenRec Workshop. pp. 1–10. ACM (2023)
14. Liu, J., Liu, C., Zhou, P., Ye, Q., Chong, D., Zhou, K., Xie, Y., Cao, Y., Wang, S., You, C., Yu, P.S.: Llmrec: Benchmarking large language models on recommendation task. arXiv preprint arXiv:2308.12241 pp. 1–13 (2023)
15. Lops, P., Silletti, A., Polignano, M., Musto, C., Semeraro, G.: Reproducibility of llm-based recommender systems: the case study of p5 paradigm. In: Proceedings of the 18th ACM Conference on Recommender Systems. pp. 116–125 (2024)
16. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. pp. 43–52 (2015)
17. NeurIPS: The machine learning reproducibility checklist. `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf` (2020), accessed: July 2024
18. NeurIPS: Neurips paper checklist guidelines. `https://neurips.cc/public/guides/PaperChecklist` (2024), accessed: July 2024
19. OpenAI: Gpt 3.5 turbo. `https://platform.openai.com/docs/models/gpt-3-5-turbo` (2021), accessed: October 2024

20. OpenAI: Gpt models' availability. https://openai.com/index/gpt-4-api-general-availability/ (2023), accessed: October 2024
21. Petrov, A., Macdonald, C.: A systematic review and replicability study of bert4rec for sequential recommendation. In: Proceedings of the 16th ACM Conference on Recommender Systems. pp. 436–447 (2022)
22. on Recommendatoin Systems, A.T.: Acm tors reporducablity. https://dl.acm.org/journal/tors/author-guidelines#reproducibility (2024), accessed: October 2024
23. Semmelrock, H., Ross-Hellauer, T., Kopeinik, S., Theiler, D., Haberl, A., Thalmann, S., Kowald, D.: Reproducibility in machine learning-based research: Overview, barriers and drivers. arXiv preprint arXiv:2406.14325 (2024)
24. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of 28th CIKM. pp. 1441–1450 (2019)
25. Wang, L., Lim, E.P.: Zero-shot next-item recommendation using large pretrained language models. arXiv preprint arXiv:2304.03153 (2023)
26. Wang, Y., Jiang, Z., Chen, Z., Yang, F., Zhou, Y., Cho, E., Fan, X., Huang, X., Lu, Y., Yang, Y.: Recmind: Large language model powered agent for recommendation. In: NAACL 2024. pp. 4351–4364 (2024)
27. Yue, Z., Rabhi, S., Moreira, G.d.S.P., Wang, D., Oldridge, E.: Llamarec: Two-stage recommendation using large language models for ranking pp. 1–6 (2023)
28. Yue, Z., Wang, Y., He, Z., Zeng, H., McAuley, J., Wang, D.: Linear recurrent units for sequential recommendation. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. pp. 930–938 (2024)
29. Zhang, J., Xie, R., Hou, Y., Zhao, W.X., Lin, L., Wen, J.R.: Recommendation as instruction following: A large language model empowered recommendation approach. arXiv preprint arXiv:2305.07001 (2023)