Systematic Evaluation of Large Language Models for Robust Rating and Ranked List Prediction

Yiu-Kai Ng

Brigham Young University, Provo UT 84602, USA
ng@compsci.byu.edu

Abstract. The advent of Large Language Models (LLMs) has the potential to transform item ratings and ranked list generation by leveraging extensive training to provide nuanced and personalized suggestions based on comprehensive contextual information about items. LLMs offer advantages, such as easy updating with new preferences and flexibility, in maintaining current relevance. In this paper, we explore the capacities of four LLMs as movie rating predictors and ranked list generators. These four advanced LLMs, Llama 3.1 70B, Qwen 2.5 32B, Mistral 7B, and OpenLLaMA 2.0 13B, were chosen due to their capabilities and largely open-source nature. We also conducted an n-shot experimentation and fine tuning to assess how different amounts of data influenced the LLMs' rating and ranking prediction. It is theorized that with more data regarding a user's specific tastes the model will have more accurate results. Specifically, we evaluated three sample sizes—5, 10, and each user's full rating history—plus fine-tuning, to assess how increased sample information affects accuracy in rating and ranking prediction. Experimental results found that in terms of rating prediction, traditional User-Based Collaborative Filtering (UBCF) outperformed the LLMs, providing higher accuracy in predicting individual movie ratings. However, when generating ranked lists for making recommendations, LLMs showed promising results, particularly Llama 3.1 70B, which demonstrated better overall recommendation performance compared to the other three LLMs and UBCF. This study presents a systematic evaluation of LLMs for rating prediction and ranked-list generation, providing novel insights for the recommendation research community.

Keywords: Rating prediction, ranked list generation, LLMs

1 Introduction

In the Internet era, item recommender systems have become integral components of platforms such as Netflix and Amazon. These systems aim to enrich user search experience by suggesting items—be they movies or products—that align with their preferences. The effectiveness of these recommendations is crucial not only for user satisfaction but also for the economic viability of content providers.

Traditional item recommendation approaches frequently rely on collaborative-filtering (CF) related techniques, which involve analyzing user-item interaction

data to predict future preferences based on past behavior. [11]. While these methods have proven effective in many scenarios, they come with notable weaknesses [15]. One significant limitation is their inability to incorporate contextual information about items themselves. Instead, traditional models rely closely on relative ratings and interactions, often leading to recommendations that are less contextually relevant [5]. Secondly, this system is dependent on previous ratings of the target item by other users [9]. Consequently, new or sufficiently niche items may not receive recommendations due to the lack of existing ratings, the traditional cold start problem [1].

The advent of Large Language Models (LLMs) has ushered in a new era for item recommender systems. Trained on vast datasets and encompassing most of the Internet, LLMs possess an extensive knowledge base about virtually every topic [17]. This comprehensive understanding allows them to draw upon rich contextual information about items, thereby enhancing the quality and relevance of recommendations. For instance, an LLM can leverage its knowledge of a movie's plot, genre, director, cast, synopsis, and perhaps even the script to provide more nuanced and personalized suggestions than traditional models [23]. LLMs help mitigate the cold start problem in recommender systems by leveraging textual descriptions, user interactions, and pretrained knowledge to generate context-aware suggestions with minimal historical data [20].

Beyond their vast repository of information, LLMs also offer several advantages in the context of item recommendations. First, they do not require other user interaction data for generating recommendations, making them particularly useful in scenarios with sparse data or where other data is constantly changing [22]. Second, updating a model's understanding based on new user preferences is straightforward—simply incorporate the new information into the prompt [3]. This flexibility ensures that suggestions remain current and relevant.

Despite these promising capabilities, there remains a lack of structured analysis evaluating the effectiveness of LLMs in item recommender systems [7]. The use of LLMs represents an intriguing challenge and presents novel ground for research in this field. By exploring how LLMs can enhance traditional recommenders, we aim to provide insights into their potential impact on improving user experience across various platforms.

In this paper, we explore the novel use of LLMs in recommender systems, highlighting how n-shot learning and fine-tuning enhance their performance. While n-shot learning allows LLMs to generate relevant recommendations from minimal examples and reducing dependency on large datasets, fine-tuning customizes LLMs with domain-specific knowledge, improving their adaptability and precision compared to traditional collaborative filtering methods. Our study provides empirical insights into the effectiveness of LLMs for key recommender system tasks, highlighting their potential in advancing rating and ranking performance.

2 Related Work

The application of LLMs in rating prediction and ranked list generation has garnered significant attention due to their ability to understand and generate

human-like text. In this section, we review several recent works that explore the use of LLMs for these purposes, describing key findings and methodologies.

In the past, Kang et al. [6] investigated whether LLMs can effectively predict user preferences in a zero-shot setting. Their study reveals that LLMs perform reasonably well on rating prediction tasks with zero shot. Most importantly to this work, they showed that zero-shot prompting of LLMs for rating prediction performed better than the global average [14].

Zhang et al. [21] and Wang and Lim [14] evaluated the effectiveness of using BERT, GPT-2.0, and GPT-3.0 for recommendation tasks. They found that these LLMs do not outperform traditional approaches like GRU4Rec [14, 21]. This indicates a significant limitation in adapting general-purpose LLMs to make recommendations at that time. Yue et al. [19], who proposed a two-stage recommender system using LLMs for ranking, named LlamaRec, leverages smaller LLMs to generate initial lists and then employs larger models like Llama2-7B for ranking, aiming to improve efficiency while maintaining performance. They found their model to return better results than other transformer-based recommenders [19].

Liu et al. [8] conducted a preliminary study on whether ChatGPT can serve as an effective recommender system. They found that ChatGPT performs better than some baseline models for item rating predictions but struggles with sequential (ranked) recommendations. This research reveals the versatility of LLMs in certain aspects of recommendation while also pointing out areas where traditional methods still excel.

In summary, these studies collectively suggest that while LLMs offer promising capabilities for information retrieval and making recommendations, significant challenges remain and none of these existing works were able to beat the state of the art collaborative-filtering approaches. Model size, specialized prompting, and task-specific adaptations are critical in further advancing the application of LLMs in this domain.

3 Methodology

In this section, we explore the application of LLMs in movie ratings and recommendations that is applicable to other domains. To achieve this, a systematic methodology is adopted, encompassing LLM choice, evaluation metrics, traditional and novel approaches.

3.1 LLM Selection

To predict item ratings and make recommendations, we prioritized Qwen 2.5 32B [18], Mistral 7B [12], OpenLLaMA 2.0 13B [10], and Llama 3.1 70B [13] due to their advanced capabilities and largely open-source nature. All these models are trained on large and diverse datasets, but Llama 3.1 70B's larger parameter set allows it to handle a broader range of topics more effectively. OpenLLaMA's smaller size makes it more suitable for real-time applications where computational efficiency is crucial. These models are among the most sophisticated available, making them suitable for applications where companies prefer not to rely on proprietary platforms like OpenAI. Compared to other LLMs, all these models exhibit superior instruction alignment and response consistency.

3.2 LLM Approach

We leverage LLMs to predict user movie ratings and generate ranked lists, using the following methodology.

- **Movie Selection**. For each user-target-movie pair, we randomly select N movies from the user's previous ratings. If there were no previous ratings, we would not feed any previous examples and request a rating from the LLM. We experimented with N set to be 5, 10, and all movies that the user had previously rated.
- **Prompt Design and Rating Prediction**. Sampled movies and their corresponding ratings are used to prompt the LLM. The LLM is then asked to predict the user's rating for the target movie. To ensure consistent and usable outputs, we require the LLM to format its predictions in JSON.
- **Output Handling**: The predicted ratings from the LLM are saved for later analysis, allowing us to compare them independently with the true baseline ratings.
- **Ranked List Generation**. For each user, all predicted ratings are arranged in a ranked list. This enables further evaluation of the methodology as an item recommender. For instance, assume the LLM had predicted ratings in the following movie-ranking pairs: "Frozen": 4, "Ratatouille": 5, "Tarzan": 1. The resultant ranked list would be [Ratatouille, Frozen, Tarzan].

This structured approach ensures that we can systematically assess both the accuracy of individual rating predictions and the overall effectiveness of the LLM-based recommender.

3.3 *N*-shot Experimentation

We systematically varied the number of input samples fed into the prompt to assess how different amounts of data influence an LLM in predicting *true user ratings* and generating *ranked lists*. Specifically, we consider three distinct levels of sample sizes: 5, 10, and the full history of each user's movie-rating samples.

- **5 Samples**. By using a minimal set of 5 samples, we can evaluate how the LLM performs with very limited information. This scenario is particularly relevant in situations where data availability is constrained or when users have not rated many movies yet.
  - In **rating prediction**, 5-shot training allows an LLM to observe how a specific user has rated a small sample of movies. These examples help the model learn the user's rating behavior, such as a tendency to rate action movies highly and dramas less favorably. With this context, the LLM can more accurately infer how the same user might rate an unseen movie based on its description. This leads to *better personalization* and *reduced error* compared to zero-shot settings.
  - For **ranked list generation**, the same 5-shot examples inform the LLM about the user's implicit preferences. The model uses these cues to score and rank a list of candidate movies. By leveraging semantic and emotional

patterns within the movie descriptions, e.g., themes, tone, and genre, the LLM can prioritize movies that better match the user's established tastes. This results in more *relevant* and *coherent* recommendations.

Overall, 5-shot training provides lightweight but effective personalization, allowing the LLM to *align* better with *user preferences* in both rating and ranking tasks without requiring fine-tuning or large training datasets.

- **10 Samples**. Increasing the sample size to 10 allows our rating prediction model and recommender system to explore more data and observe its prediction accuracy. This intermediate step determines whether there is a noticeable gain in performance with a moderate amount of data.
  - **Movie rating prediction**. With 10 examples, an LLM gains a more comprehensive view of the user's rating behavior across diverse genres, themes, or emotional tones. This enables the model to *more accurately learn* and *generalize* how the user evaluates movies. The result is *higher prediction accuracy*, as the LLM can make *more fine-grained inferences* about how the user might rate a new movie based on its content features.
  - **Movie ranked list generation**. In ranked list generation, 10-shot training provides an LLM with a clearer profile of the user's preferences, enabling it to prioritize candidate movies that align closely with the user's demonstrated interests. The additional examples help the model distinguish *subtle preferences*, resulting in *better personalized ranking* with *improved relevance* and *diversity*.

  All in all, 10-shot training deepens an LLM's personalization capabilities, reducing ambiguity and improving both *rating accuracy* and *recommendation quality*. It strikes a balance between performance and efficiency, avoiding full fine-tuning while capturing more complex user-specific signals than 5-shot training.
- **Full History**. Utilizing the full history of movie ratings for each user provides a baseline to assess the maximum predictive power of the LLM. This configuration ensures that the model has access to all available information, which can be crucial for capturing nuanced preferences and trends.
  - **Movie rating prediction**. With access to a user's entire rating history, an LLM can learn the entire *comprehensive behavioral patterns*, including long-term preferences, genre-specific biases, rating consistency, and even changes in taste over time. This rich context allows the model to make highly accurate, fine-grained predictions for unseen movies by mapping new content to deeply understood user behavior. This leads to lowest *Root Mean Squared Error* (RMSE) and *Mean Absolute Error* (MAE), as the model benefits from a more complete signal.
  - **Movie ranked list generation**. For ranked recommendations, full-history input enables an LLM to construct a nuanced and context-aware user profile. The model can better assess the relative appeal of candidate movies by aligning them with established emotional, thematic, and stylistic preferences seen throughout the user's history. This results in *higher-quality, more diverse*, and *more relevant recommendation lists*, as evidenced by improved ranking metrics, such as nDCG, RBO, and Spearman.

Table 1: Findings on 5-shot, 10-shot, and full-history training

Training Setting	**Ranked List Quality**	**Strengths**	**Limitations**
5-shot	Basic personalization	Low computational cost; fast inference	Limited user preference signal; lower coverage
10-shot	Noticeably better personalization	Balances cost and performance; better taste modeling	May still miss subtle or evolving preferences
Full History	Most personalized and diverse	Richest user profile; strongest generation	High memory/computation cost; potential overfitting on noisy ratings

In summary, using a user's full rating history allows an LLM to closely emulate the behavior of traditional collaborative filtering methods while leveraging the LLM's strength in semantic and contextual understanding.

Table 1 summarizes the findings from our empirical studies on 5-shot, 10-shot, and full-history training of LLMs to improve performance in movie rating prediction and ranked list generation (see details in Section 5.3).

3.4 Fine Tuning LLMs

Qwen 2.5 32B is an open-source, 32-billion parameter model that offers strong performance in code generation, whereas Mistral 7B is a lightweight yet high-performing model, known for strong reasoning, coding, and multilingual abilities. In addition, OpenLLaMA 2.0 13B is an open-source LLM that provides accessible versions of Meta's LLaMA models. It offers models that researchers, developers, and organizations can freely use, modify, and fine-tune. Furthermore, Llama 3.1 70B, which is well-known for its strong performance on benchmarks used in various applications, have been chosen to analyze item ratings and recommendations. Fine tuning these LLMs allows for tailoring a pre-trained model to item ratings and making recommendations to improve accuracy and efficiency for these tasks. Using different subsets of the MovieLens 25m dataset, we have rendered user ratings, movie ID, and user ID, and this dataset has been split by 90% for training, 5% for testing, and 5% for validation. Both the training and validation sets were used to assess the performance of the four LLMs.

In **movie rating prediction**, fine-tuning enables the LLMs to learn consistent relationships between movie metadata, such as plot, genre, and cast, and user rating patterns. This leads to lower RMSE and MAE, as these models become better at estimating how users rate different types of movies. For **ranked list generation**, fine-tuned LLMs can internalize user preference signals, allowing them to produce more personalized and coherent movie rankings. This often improves metrics like nDCG and Spearman, reflecting better alignment with user taste in top-k recommendations. On the whole, fine-tuning LLMs on historical movie ratings allows the model to develop deeper *contextual awareness* of *user preferences*. It outperforms simple prompt-based methods in capturing long-term viewing habits (see Section 5 for the empirical study that verifies the per-

Table 2: Impact of fine-tuning LLMs

Aspect	Before Tuning	After Tuning	Impact
Rating Prediction	Generic, surface-level estimates based on pretraining knowledge	Ratings better aligned with user preferences and movie attributes	Lower RMSE/MAE; improved accuracy in numerical predictions
Ranked List Generation	Limited personalization; weak user modeling	Context-aware, user-specific ranked lists	Higher nDCG, RBO and Spearman; more relevant top-k recommendations

formance of various training methods), subtle genre preferences, and emotional tones—leading to more accurate and personalized movie recommendations. Table 2 highlights the impact of fine-tuning LLMs for movie rating prediction and ranked list generation based on observations from experimental results detailed in Section 5.3.

The fine-tuning process is done using cross-entropy loss, where the model learns to predict user ratings and generate movie recommendations based on user preferences. Parameter-efficient techniques like LoRA are employed to reduce computational load. Through instruction-based prompts, the models are trained to make both accurate rating predictions and personalized recommendations from minimal user input. Hereafter, the models are optimized for deployment using quantization and inference acceleration to support real-time, scalable recommendation systems.

3.5 User-Based Collaborative Filtering (UBCF)

We employ UBCF to predict user ratings based on

- **Neighborhood Selection**. Users who have rated items similarly to the target user are identified as 'neighbors'.
- **Rating Prediction**. The predicted rating for an item is computed by averaging neighbor ratings weighted by similarity scores, leveraging historical data to infer preferences without modeling individual features explicitly.

UBCF provides a baseline framework for comparing traditional collaborative-filtering with advanced LLM-based techniques in the context of movie recommender systems.

4 Performance Evaluation Metrics

Results of our empirical study were compared in two separate ways. First, the accuracy of the LLMs as rating predictors was analyzed via MAE, $RMSE$, and R^2 [2]. MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is calculated as the average of the absolute differences between predicted and actual values. MAE is straightforward to interpret and less sensitive to outliers, which can be particularly useful for datasets with a limited range of values where extreme values are less common. $RMSE$, on the other hand, measures the square root of the average squared differences between predicted and actual values and gives more weight to larger errors due to the squaring of the error terms, making it more sensitive

Table 3: An example of Ranked-Based Overlap (RBO)

Rank	**My List**	**Robin's List**	Common Items	Total Items	Over-lap	Average Overlap
1	K. Bekele	H. Gebrselassie	{}	{B, G}	0	0
2	H. Gebrselassie	K. Bekele	{B, G}	{B, G}	1	0.50
3	P. Tergat	E. Zatopek	{B, G}	{B, G, T, Z}	0.50	0.50
4	E. Zatopek	K. Jomet	{B, G, Z}	{B, G, T, Z, J}	0.60	0.53
5	K. Jomet	M. Farah	{B, G, Z, J}	{B, G, T, Z, J, F}	0.67	**0.55**

to outliers. Also known as the *coefficient of determination*, R^2 is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. A higher R^2 value indicates a better fit of the model to the data, suggesting that a larger portion of the variability in the dependent variable is accounted for by the model.

Second, the accuracy of the LLMs as recommenders was analyzed via the Spearman's Rank Correlation, Normalized Discounted Cumulative Gain (nDCG), and Rank-Based Overlap (RBO). *Spearman's Rank Correlation* is a statistical measure that assesses the *strength* and *direction* of a monotonic relationship between two variables by analyzing the *ranks* of their data points, whereas RBO, a similarity measure between top-weighted, incomplete and indefinite rankings, is straightforward to understand and compute, does not require both lists to have the same number of items, and assigns more importance to items that are ranked higher in the lists, as these values are calculated more frequently.

RBO is a system that measures the overlap between two lists at each step, calculated as the ratio of the intersection over the union of the sets of each list. Table 3 explains the system's calculation. In the graph, the overlap at any given point is shown, but the RBO is the Average Overlap that is given in the right-most column. This metric has several advantages over other methods:

- **Intuitive and Quick Calculation**. The Rank Based Overlap is straightforward to understand and compute.
- **Length Indeterminacy**. It does not require both lists to have the same number of items.
- **Weight to Higher Ranked Items**. It assigns more importance to items that are ranked higher in the lists, as these values are calculated more frequently.

nDCG (normalized Discounted Cumulative Gain) evaluates ranking quality by assigning higher scores to relevant items that appear earlier in the recommended list. It uses a logarithmic discount to reflect the reduced importance of lower-ranked items. In our setting, *relevance* is derived from the ground-truth order of user preferences. This metric suits LLM-based recommenders, aiming to generate high-quality ranked lists rather than predict exact ratings.

5 Experimental Results

In this section, we present the experimental results of an empirical study using various LLMs as rating predictors and ranked list generators. The source code and the results of the empirical study are made available through https://github.com/xyz7461/CBF-LLMs.git.

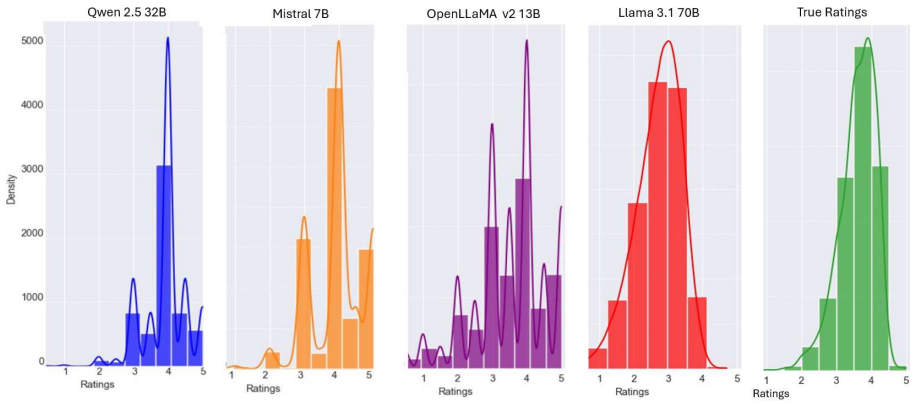


Fig. 1: Rating distributions of models using all the ratings of each user, where *True Ratings* denotes UBCF

5.1 Datasets

The primary dataset utilized for this study is the MovieLens 25m dataset [4]. This dataset includes ratings from approximately 162K users for about 62K movies. It is widely used in the recommender community due to its balanced mix of users and items, well-documented structure, and historical significance. For our evaluation purposes, we randomly selected 10 movies from each user's rating history to form a test set. Additionally, random subsets of potential sample movies were also included to simulate recommendation scenarios. This dataset is ideal as it provides a realistic and manageable scale for testing LLMs in a recommendation context.

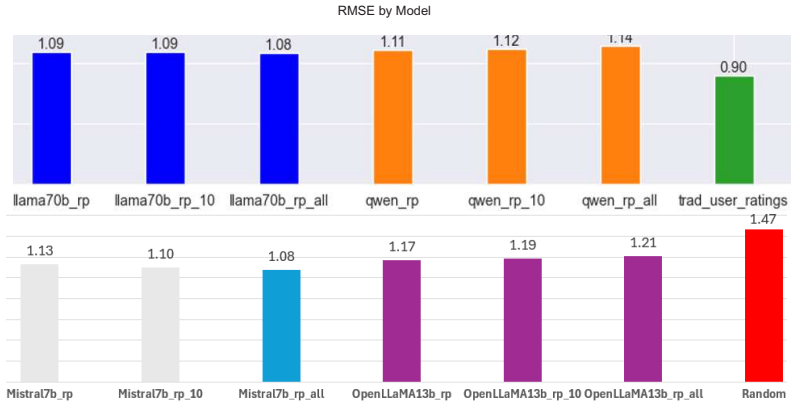
5.2 Item Rating Prediction

We first evaluate the *rating predictions* of various models–UBCF, Llama 3.1 70B, Qwen 2.5 32B, Mistral 7B, and OpenLLaMA 2.0 13B–based on the performance metrics presented in Section 4. We examine predicted rating distributions to identify model biases and analyze individual metrics for detailed assessment.

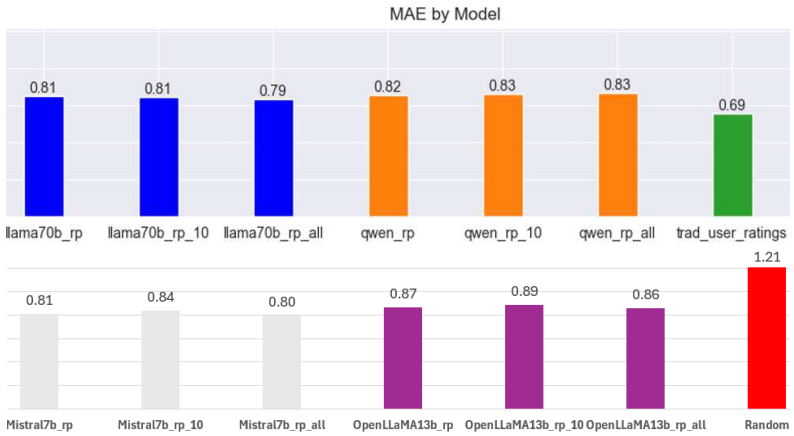
Distributions We examine the output *rating distributions* of each model using graphs. The graphs as shown in Figure 1 help us understand the native biases of each model and provide insight into the expected behavior of each model. However, these distributions do not indicate the accuracy of the models. To illustrate the accuracy, a random distribution was generated using the *mean* and *standard deviation* of the *true data distribution*, and its results are compared in the next section. Figure 1 depicts the full history of movie ratings[1], and the model names are listed above their respective distributions.

Metrics We examine the individual metrics of the rating predictions. The results of each model on these metrics are plotted in Figure 2, where *trad_user_rating* denotes *UBCF*.

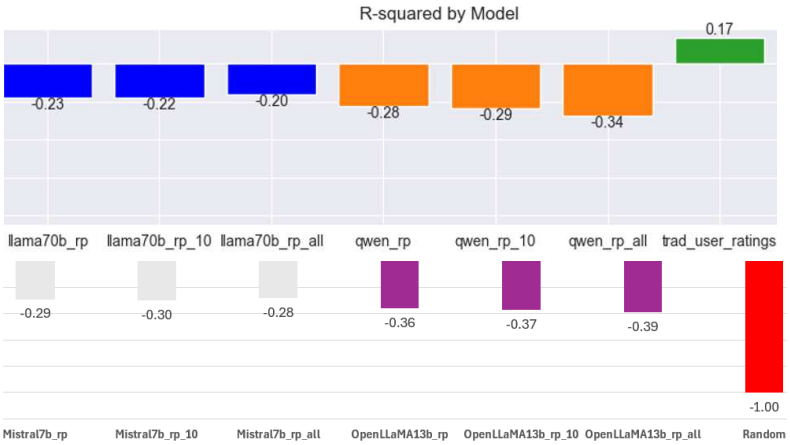
[1] The 5- and 10-samples follow the same pattern



(a) RMSE measures

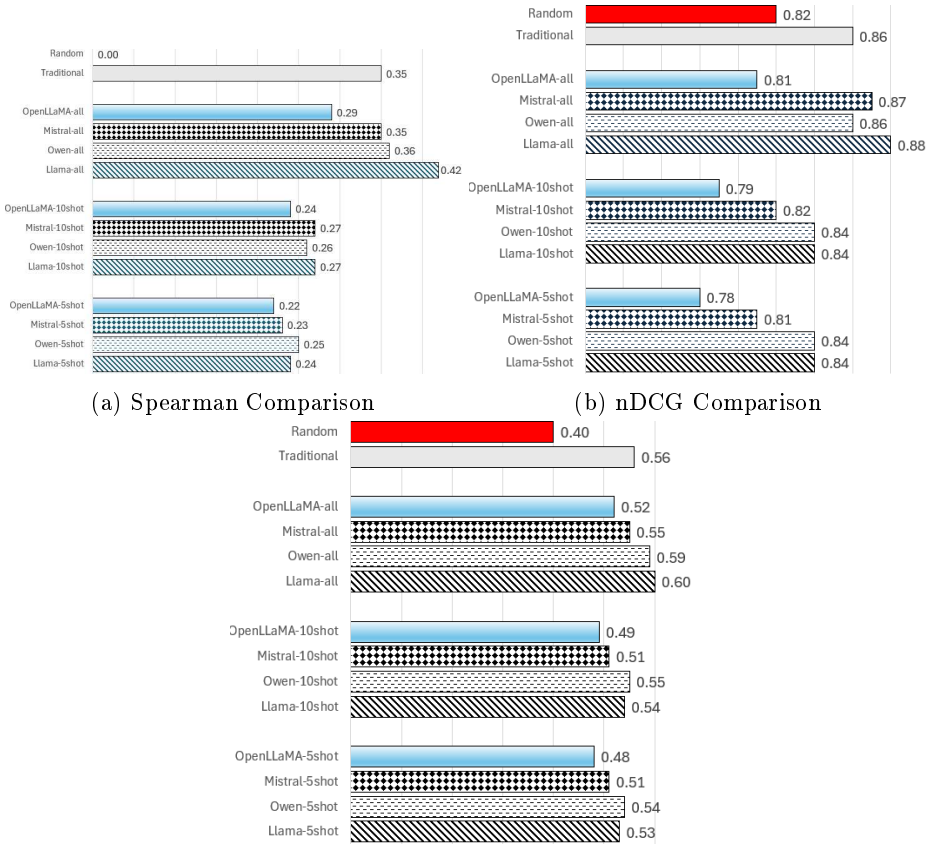


(b) MAE measures



(c) R2 measures

Fig. 2: Rating prediction metrics of various models



(a) Spearman Comparison (b) nDCG Comparison

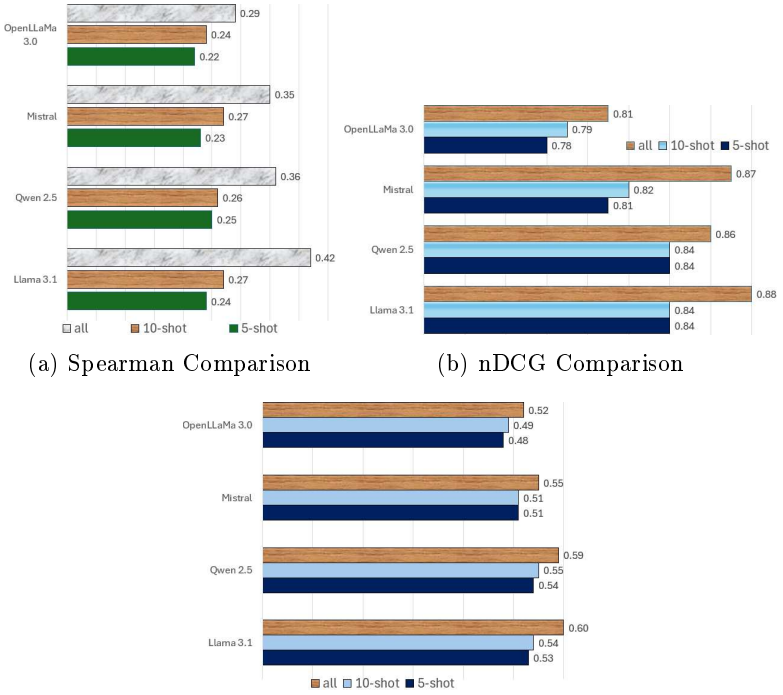
(c) RBO Comparison

Fig. 3: Ranked list metrics of various models

As depicted in Figure 2, neither Qwen 2.5, Mistral 7B, OpenLLaMA 2.0, nor Llama 3.1 achieved the accuracy of $UBCF$ in terms of RMSE and MAE. Consistently, Llama 3.1 exhibited slightly lower RMSE and MAE values than Qwen 2.5 32B. The coefficient of determination, i.e., R^2, indicated that only $UBCF$ yielded a positive R^2 value (see Figure 2). The results indicate that $UBCF$ outperformed the other LLMs because CF is specifically designed to model historical user-item rating patterns, leading to lower MAE and RMSE values. Its positive R^2 further suggests a better fit to the true rating distribution than the LLMs. This suggests that $UBCF$ had a better fit to the true data, capturing more variance in ratings compared to the four LLMs.

5.3 Ranked List Prediction

In this section, we compare the models' ability to generate ranked lists. Different metrics assess how well the models match the *relative ranking* of movies as provided by the user, regardless of individual accuracy. For example, if a user rated the movies *Ratatouille* a 5 and *Frozen* a 4, the model is considered accurate if it also ranks *Ratatouille* higher than *Frozen*, even if the specific ratings differ.



(a) Spearman Comparison

(b) nDCG Comparison

(c) RBO Comparison

Fig. 4: N-Shot performance comparison between Llama 3.1 and other LLMs

Ranked List Comparisons As shown in Figure 3's ranked list generation, Llama 3.1 outperformed other methods, where "Traditional" denotes UBCF, when *all* user ratings were used as inputs, and the results are statistically significant ($p < 0.05$) based on the Wilcoxon Signed-Ranks test [16]. Although the predicted exact values may not be 100% accurate, the relative ordering of items was more precise than the $UBCF$ model, validating the potential utility LLaMA 3.1 and Qwen 2.5 for *ranking prediction*.

N-Shot Performance There is a clear trend showing that increasing the number of shots ($N = 5$ and 10) generally improved performance across all models for *ranked list generation*. In Figure 4, it can be seen that for Llama 3.1 and Qwen 2.5, going from 5-shot to 10-shot was marked by a marginal increase, but when prompting based on all the data there is an increase, most notably seen in the Spearman metric. This suggests that providing more context to the models enhances their ability to generate accurate ranked lists.

Llama 3.1 and Qwen 2.5 performed remarkably similarly in most metrics as can be seen in Figure 4. The maximum difference between the models was 0.02 in nDCG and RBO (Rank-Biased Overlap). On the other hand, in the *Spearman rank correlation metric*, Llama 3.1 exhibited substantially better performance than Qwen when all samples were included. This is intuitive, as LLama3.1 70B is a larger model, and indicates that given full context, it is notably better to use Llama 3.1 than Qwen 2.5 and others if model-size is not an issue.

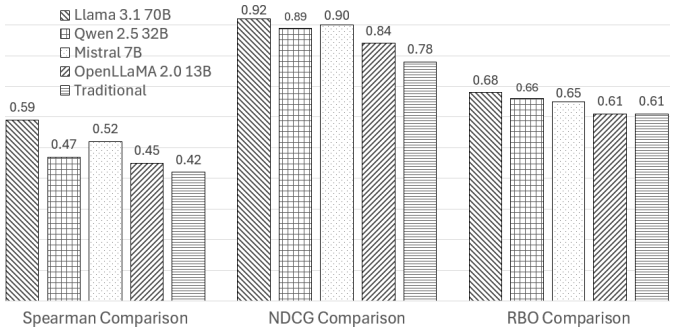


Fig. 5: Fine-tuning performance of the four LLMs

Note that while $UBCF$ demonstrated superior raw rating *prediction accuracy*, Llama 3.1 showed potential in generating more accurate *ranked* lists, particularly when given sufficient context through increased inputs. This highlights the complementary strengths of $UBCF$ and LLMs in information retrieval tasks. Moreover, the results were verified to be statistically significant ($p < 0.045$) based on the Wilcoxon Signed-Ranks test.

Fine-Tuning Performance Fine tuning adjusts its parameters to improve performance on a specialized task, which is rank-list prediction in our case. The fine-tuning approach not only boosts task-specific performance but also avoids the enormous computational cost of full model training, making it far more practical and accessible. In our case, the goal of fine tuning is to enhance performance on Llama 3.1 and Qwen 2.5 in terms of *rank prediction*.

As shown in Figure 5, LLaMA 3.1 demonstrates remarkable performance boost, since the fine-tuned model has revealed anywhere from significant to moderate improvements in various measures, whereas Qwen 2.5 also exhibits substantial gain with enhanced performance metrics.

Research Questions to be Addressed Based on the experimental results obtained from various empirical studies conducted in this paper, we address the following research questions and provide the corresponding answers:

– RQ_1. What factors contribute to the superior rating prediction accuracy of traditional UBCF compared to LLMs?

  The experimental results show that UBCF remains more accurate than LLMs when predicting user ratings. This suggests that, despite their scale and generalization power, LLMs may not fully capture the fine-grained, personalized user-item interactions that UBCF models are designed to exploit. UBCF directly leverages user similarity and historical co-rating patterns, which are highly effective for rating prediction tasks, especially in domains with dense rating matrices or strong user homogeneity. In contrast, LLMs often operate on textual or behavioral embeddings without explicit modeling of user-user relationships.

- RQ_2. How can LLMs be optimized or adapted to improve rating prediction tasks while retaining their strength in generating ranked recommendations?

  LLMs show strong performance in generating ranked lists but underperform in precise rating prediction. This raises opportunities to explore adaptations that enable LLMs to better handle both tasks simultaneously. One direction is to fine-tune LLMs specifically on explicit rating datasets, incorporating supervised regression objectives alongside ranking objectives during training. This could encourage the model to learn both the ordinal structure of preferences and the numeric calibration required for rating prediction.

- RQ_3. To what extent does training volume and model scale, e.g., Llama 3.1 70B versus smaller LLMs, influence the recommendation performance of LLM-based systems, and where are the diminishing returns in scaling for recommendations?

  The observed superior performance of Llama 3.1 70B compared to smaller models, such as Qwen 2.5 32B and OpenLLaMA 2.0 13B, suggests that larger models with more extensive training corpora can better generalize patterns for ranking tasks. However, the question remains how much additional performance gain is achieved with increased scale and whether those gains justify the substantial computational and resource costs. This question invites empirical studies that systematically compare model scale, training data volume, and downstream recommendation performance.

6 Conclusion

We evaluated the performance of Llama 3.1 70B and three well-known LLMs, Qwen 2.5 32B, Mistral 7B, and OpenLLaMA 2.0 13B, based on *rating prediction* and *ranked list generation* against the traditional user-based collaborative filtering (UBCF) model. The results provide valuable insights into the capabilities and limitations of these approaches. In *rating prediction*, $UBCF$ outperformed the four LLMs, achieving higher accuracy as measured by RMSE and MAE. The coefficient of determination (R^2) further confirmed that $UBCF$ had a better fit to the true data distribution compared to the four LLMs. The LLMs showed some bias towards predicting higher ratings, particularly around 4 and 3, indicating an area for improvement in their calibration, but they still significantly outperformed the random benchmark. In *ranking*, however, LLaMA 3.1 performed best, especially when all user ratings and fine-tuning were applied. This suggests that LLMs are adept at capturing relative item preferences. The performance improvement with increased n-shot inputs and fine-tuning underscores the importance of providing sufficient context to enhance model accuracy. The findings show the complementary strengths of common $UBCF$ and elite LLMs. Our work offers key insights into the distinct strengths of LLMs in generating high-quality ranked recommendations while highlighting areas where classical methods excel in precise rating prediction. This dual focus provides a comprehensive understanding that bridges rating accuracy and ranking effectiveness, marking a significant contribution to recommender system research.

References

1. Bai, H., et al.: Unified Representation Learning for Discrete Attribute Enhanced Completely Cold-Start Recommendation. IEEE TOBD **14**(8), 1–12 (August 2024)
2. Chicco, D., et al.: The Coefficient of Determination R-squared is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation. Peerj Computer Science **7**, e623 (2021)
3. Deldjoo, Y., et al.: A Review of Modern Recommender Systems Using Generative Models (Gen-Recsys). In: Proc. of ACM SIGKDD. pp. 6448–6458 (2024)
4. Fan, Y., Ji, Y., Zhang, J., Sun, A.: Our Model Achieves Excellent Performance on Movielens: What Does it Mean? ACM TOIS **42**(6), 1–25 (2024)
5. Google-Developers: Collaborative Filtering for Recommendations. developers. google.com/machine-learning/recommendation/collaborative/summary (2025)
6. Kang, W., Ni, J., Mehta, N., Sathiamoorthy, M., Hong, L., Chi, E., Cheng, D.: Do LLMs Understand User Preferences? Evaluating LLMs on User Rating Prediction. arXiv preprint arXiv:2305.06474 (2023)
7. Lin, J., et al.: How Can Recommender Systems Benefit from LLMs: A Survey. ACM TOIS **43**(2), 1–47 (2023)
8. Liu, J., et al.: Is ChatGPT a Good Recommender? A Preliminary Study. arXiv preprint arXiv:2304.1014 (2023)
9. Natarajan, S., et al.: Resolving Data Sparsity and Cold Start Problem in Collaborative Filtering Recommender System Using Linked Open Data. Expert Systems with Applications **149**, 113248 (2020)
10. Rangel, J., et al.: Sparql Generation: An Analysis on Fine Tuning OpenLLaMA for QA over a Life Science Knowledge GrapharXiv preprint arXiv:2402.04627 (2024)
11. Shao, P., et al.: Average User-Side Counterfactual Fairness for Collaborative FilteringACM TOIS **42**(5), 1–26 (2024)
12. Thakkar, H., Manimaran, A.: Comprehensive Examination of Instruction-Based Language Models: A Comparative Analysis of Mistral-7B and Llama-2-7B. In: Proceedings of ICERCS. pp. 1–6. IEEE (2023)
13. Vavekanand, R., Sam, K.: Llama 3.1: An In-Depth Analysis of the Next-Generation LLM (2024)
14. Wang, L., Lim, E.: Zero-Shot Next-Item Recommendation Using Large Pretrained Language Models. arXiv preprint arXiv:2304.03153 (2023)
15. Wang, Y., et al.: A Survey on the Fairness of Recommender Systems. ACM TOIS **41**(3), 1–43 (2023)
16. Woolson, R.: Wilcoxon Signed-Rank Test. Wiley Encyclopedia of Clinical Trials pp. 1–3 (2007)
17. Wu, L., et al.: A Survey on Large Language Models for Recommendation. World Wide Web **27**(5), 60 (2024)
18. Yang, A., et al.: Qwen2. 5 Tech. Report. arXiv preprint arXiv:2412.15115 (2024)
19. Yue, Z., et al.: LlamaRec: Two-Stage Recommendation Using Large Language Models for Ranking. arXiv preprint arXiv:2311.02089 (2023)
20. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Computing Surveys **52**(1), 1–38 (2019)
21. Zhang, Y., et al.: Language Models as Recommender Systems: Evaluations and Limitations. In: Workshop on I (Still) Can't Believe It's Not Better! (2021)
22. Zhao, Z., et al.: Recommender Systems in the Era of Large Language Models (LLMs). IEEE TKDE **36**(November), 6889–6907 (2024)
23. Zhu, Y., Liu, Y.: Innovative Prompting Strategies and Holistic Evaluation of LLM Movie Recommender. In: Proceedings of IEEE ICNGN. pp. 1–5 (2024)