Comparing Human and LLM Ratings of Music-Recommendation Quality with User Context

Sherol Chen[1] **Sally Goldman**[1]
Amnah Ahmad[4*] **Ndaba Ndebele**[3]
Andrew Lampinen[2] **Michael C. Mozer**[2] **Jie Ren**[2]
[1]Google Research [2]Google DeepMind [3]Google Cloud [4]Independent
{sherol,sgoldman,ndebele,lampinen,mcmozer,jjren}@google.com
amnahahmad@gmail.com

Abstract

When developing new recommendation algorithms, it is common to run user studies in order to obtain initial evaluation metrics. We explore using Large Language Models (LLMs) as a proxy for human ratings (an AutoRater) for this setting. In particular, we explore how effectively an LLM with basic Chain-of-Thought (CoT) prompting can predict human ratings, and also leverage knowledge of a user's music likes and dislikes. We ran a study in which paid users provided queries and then rated the resulting playlist. These users also provide their music likes and dislikes. We compare the ratings between the AutoRater system to a human rater baseline—a group of participants who were asked to perform the same task of rating the playlist recommendations, either with or without user context.

We found the AutoRater to be as effective if not more so than human raters, and that providing user context leads to higher correlations for both. Also the correlations reliably increase with the size of the rater pool. These results were statistically reliable. Although numerically higher correlations are obtained with 8 songs than with 4 songs, the difference is not statistically reliable. Interestingly, interaction between the rater type crossed with the number songs is statistically reliable reflecting that human raters perform as well or better than the AutoRaters for 4 song playlists, whereas the pattern is flipped for 8 song playlists where AutoRaters can use broad knowledge of music to make more nuanced ratings.

1 Introduction

When developing new recommendation algorithms, it is common to run user studies in order to obtain initial evaluation metrics. In this paper, we explore using Large Language Models (LLMs) as a proxy for human ratings (an AutoRater) for such settings. How effectively can LLMs with basic Chain-of-Thought (CoT) prompting predict human ratings, and can they leverage knowledge of a user's music likes and dislikes, to improve their performance. We explore these questions in the domain of evaluating custom music playlist generation.

When analyzing the results of user studies, we typically aggregate the ratings of all users in order to measure the quality of various algorithms. However, there is significant overhead in setting up and collecting human ratings from these user studies. Recently, LLMs have been used as a proxy model for human ratings (an AutoRater), e.g. human preferences, evaluations, or feedback [4, 10, 2, 3, 7]. Many automatic rating methods have been built through fine-tuning on human rating data, or even zero-shot/few-shot prompting LLMs [9]. For example, GPT-4-Judge, which is built

*This work was done when the author was at Google.

NeurIPS 2024 Workshop on Behavioral Machine Learning.

based on GPT-4 models [1] has been used to evaluate models' responses on open-ended questions [15, 5, 6], predicting human preference on a pair of responses [9], and grading students' homework and generating feedback [8]. In particular, it has been shown that GPT-4-Judge can achieve over 80% agreement with human preferences on model evaluation on using open-ended questions [15].

However, not all questions have an objective answer. In this work, we perform a case study on developing an AutoRater for a music playlist recommendation system where ratings are idiosyncratic. A very good playlist for one person, might not be appreciated by a different person. Thus, in order to accurately predict user ratings, an AutoRater needs to use information about a users personal likes and dislikes to predict how well a playlist would satisfy a user's query with respect to their personal preferences. While prior work has considered the role of contextual personas (or role-playing) in LLM responses [12], most of this work has not considered the specific goal of developing AutoRaters that can capture peronsal music likes and dislikes. To measure the quality of AutoRater predictions, we compare them to a human baseline, in which a human rater who is different than the original user predicts the original user's ratings, using the same information about the user's music likes and dislikes.

In our study, for half of the queries the playlist returned had 4 songs, and in the other half the playlist returned had 8 songs. We studied the extent to which an AutoRater could utilize the provided contextual information about a user's musical taste in predicting that user's rating for a playlist as compared to the human raters. We compared the by-user average rating of n independent runs of the AutoRaters with the average rating of n human raters as we vary n. Since there are many tied ratings, we use Kendall-Tau-b as our correlation metric to measure the agreement between raters' prediction and the ground truth users' satisfaction level. See Section 3 for more details. For our experiments, we used a human user rating interface to collect human rating data. Gemini 1.5 Flash model [11] was used to build an AutoRater to predict user's satisfaction level on the recommended content with an added prompt to capture the user's music likes and dislikes.

We found that the AutoRater is as effective if not more so than the human raters, and that providing user context leads to higher correlations than when the context is not provided. Also the correlations reliably increase with the size of the rater pool. These results were statistically reliable. Although numerically higher correlations are obtained with 8 songs than with 4 songs, the difference is not statistically reliable. Interestingly, the rater type $\times$ number songs interaction is statistically reliable reflecting the fact that human raters perform as well or better than the AutoRaters for 4 song playlists, whereas the pattern is flipped for 8 song playlists where the AutoRater can use its broad knowledge of music to make more nuanced ratings.

2 Methodology

Our study involves two groups of human participants: (1) The User Group which generates the queries and evaluates the quality of the playlists. The ratings are used as the ground truth ratings. (2) a separate Human Rater Group which predicts the ratings given the users' queries and the generated playlists. See Figure 1. All human participants, including both human users and human raters, were based in the United States and chosen based off a yes answer to the question, "Have you listened to music apps at least once a week for the last month?" These participants are paid contractors who receive a standard contracted wage, which is above the living wage in their country of employment.

User Group. We first recruited 20 human users to generate queries to the music playlist generation system. Our experiments were conducted through an internal human labeling platform that allowed the users to interact with the playlist recommendation service. In each user session, they were asked to generate about 8 custom playlists. First each user performed a phase where they provide a query to a commercial music playlist creation system which responded with a playlist title, playlist description, and a playlist of either 4 or 8 songs for which the user can listen to the songs.

After the user completed all their queries, they switched into a "rater mode" in which they revisited each query and resulting playlist and we asked to answer the question "Overall, how satisfied are you with the music that was recommended?" Options provided were: "Very dissatisfied", "Somewhat dissatisfied", "Neither satisfied nor dissatisfied", "Somewhat satisfied", "Very satisfied", which we coded respectively as 1, 2, 3, 4, and 5. These ratings are used as the ground truth ratings. After all of these ratings were provided, the user answered the following two questions, which are used as the context for the second phase of the study.

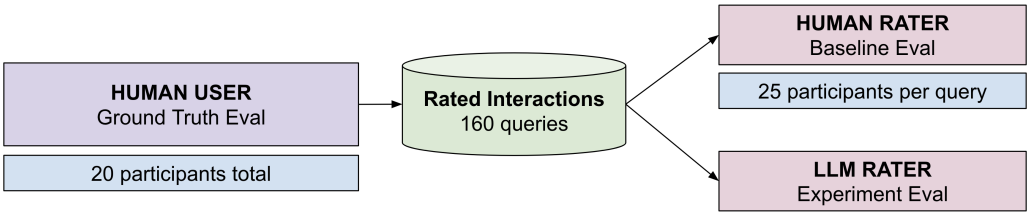

Figure 1: A group of 20 human users first generates the 160 rated interactions. Human Raters and LLM Raters are measured with respect to their ability predict these 160 ratings.

- "Please write a couple sentences about your taste in music. What genres do you like? What are your favorite artists? Describe the type of music you like."
- "Please write a couple sentences about the types of music you dislike."

An example response: "I like a LOT of music types. Grunge, Pop, Metal, Indie and 80s/90s..." and "I do not enjoy nor listen to much country or oldies..."

In total, we collected 160 tuples of (`user`, `query`, `playlist`, `ground truth rating`). Both the human raters and AutoRaters are given the task to predict the ratings, given users queries and the generated playlists so that we can compare their performance. To reduce the variance of the ratings, each of the 160 rating tasks (`user`, `query`, `playlist`), is rated by multiple human raters and multiple runs of the AutoRater (with variations via the temperature). The independent variables that we control are, (1) whether the playlist is of length 4 or 8, (2) whether context about the musical likes/dislikes of the user are provided, and (3) whether a human rater or AutoRater is used. We now describe a bit more details about the two rater groups.

Human Rater Group We recruited a separate group of human raters who were given the task to predict the ratings of the playlist result for each of the original user queries, with or without the information collected about the user's music likes and dislikes.

LLM AutoRaters For Autorater, we use a LLM AutoRating platform [7] that ensures the consistency between tasks issued to human rater and the AutoRater. More specifically, the platform issues and manages similar configuration files that setup the LLM evaluations as well as the evaluations presented to the Human Users and Raters using the same instructions. Besides the basic prompting to the LLM using (`query`, `playlist`), we also add users contextual sentiments of overall music like/dislike information. Gemini 1.5 Flash [11] is used as the LLM. The LLM is provided the same instructions as the user along with a basic prompt include a CoT question about the results before being asked the rating question. The exact prompts are included in the Appendix.

3 Results

To analyze the correspondence between user (ground truth) and rater scores on the question, "Overall, how satisfied are you with the music that was recommended?," we computed a user-rater correlation coefficient based on Kendall tau-b correlation coefficient [13, 14]. We chose Kendall tau-b as our dependent measure because it is based on ranking—i.e., whether the raters and users matched in their preference of one query response to another—and because it addresses ties where the two scores are identical, as was common in our data set. We sampled the correlation 100 times to get a low-variance estimate of the distribution mean. Each correlation required the bootstrap sampling from our 25 (either human or LLM) raters.

Recall that all users first provide their queries and see the results, and then switch to a phase where they are asked the question about the result quality. Thus we expect that the ground truth metrics for each user are well calibrated in that ratings of 5 ("Very satisfied") were better results than ratings of 4 ("Somewhat satisfied"), and so on. This same calibration is unlike to hold across users. Thus we

compute the correlation for each user and average those. More specifically, let U be the set of users, and let Q_u be the set of queries and playlists for user $u \in U$ for with 4 (or 8) songs depending on which setting we are considering. The ground truth rating vector for user u is $T_u = (t_1, \ldots, t_i, \ldots, t_{|Q_u|})$ where t_i is the satisfactory result rating given for question $q_i \in Q_u$. We vary the number of raters n selected from 1 to 20 via sampling with replacement (bootstrap sampling) from the 25 raters (human or LLM runs) that we have available. We then construct $Y_u = (y_1, \ldots, y_i, \ldots, y_{|Q_u|})$ where y_i is the average for q_i of the predictions over the n raters. We define $\mathrm{corr}_u = KT_b(T_u, Y_u)$ for user u. Our final correlation measure is the average of corr_u over all users, i.e. $\mathrm{corr} = \frac{1}{|U|} \sum_{u \in U} \mathrm{corr}_u$.

Figure 2 presents the Kendall's tau coefficient as defined above for both human and AutoRaters (red and green, respectively), 8- and 4-song conditions (top and bottom row, respectively), and with or without user context (left and right columns, respectively), as a function of the number of raters pooled to obtain the rating (the horizontal axis). The points are the average of corr_u over all users. For each user u, the correlation is the agreement between the ground truth rating and the average prediction rating. The average prediction rating is computed by randomly selecting the specified number of raters from the pool and averaging their ratings. The solid lines are a model fit, to be described shortly.

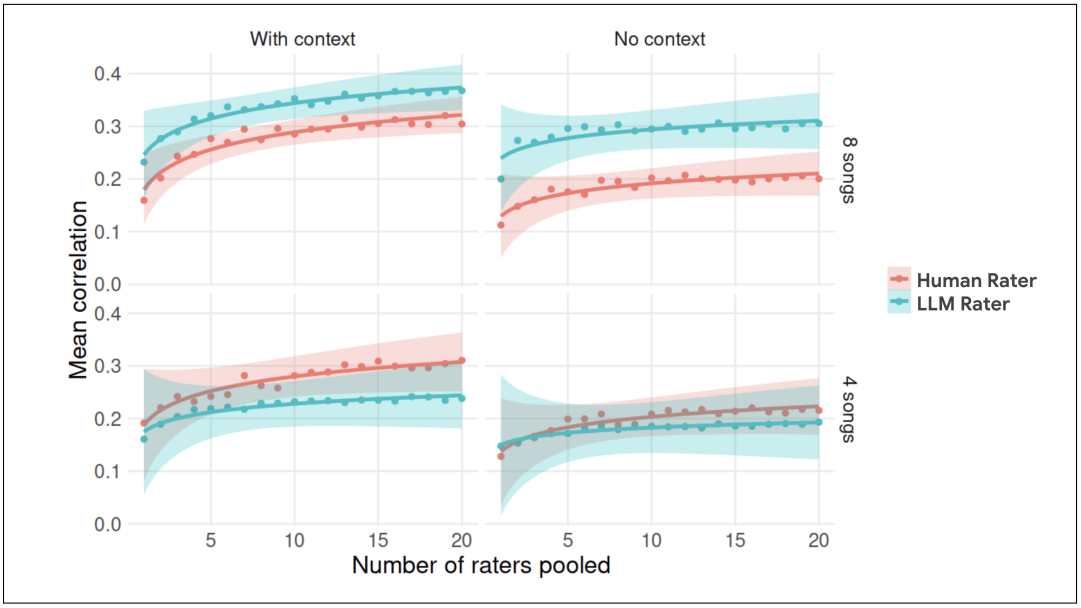

Figure 2: Comparison of human raters and LLM AutoRaters, with and without user context, on their agreement with ground truth human ratings, as measured by averaged per-user rank correlation. Points are average values, curves and error regions are smoothed (independently per condition) via linear regression from the logarithm of the number of raters.

To determine the effect of different factors on user-rater correlation, we fit the correlation to a linear mixed-effects regression model with three categorical factors (human vs. AutoRater, 4 vs. 8 songs, and context or no context) and one interval-scale factor (number of raters pooled). We perform the statistical testing for each factor to determine if its effect is statistically reliable. The degrees of freedom—and resulting p values—for the Student's t test are based on the Welch-Satterthwaite equation. We log transformed the number of raters based on visual inspection of the original correlations, which permitted a linear fit. Using a Bayesian information criterion (BIC) to perform model selection, we included only the rater-type by number-of-songs interaction term. (Note that the curve fits in Figure 2 do not exactly correspond to the model we have described because the curves are fit to each series separately.)

The regression model allows us to determine the statistical reliability of individual effects, which mostly match one's qualitative intuitions when inspecting the figure. Specifically, we find that

1. The LLM rater is as effective if not more so than the human raters ($t(1181)$=3.34, p<.001)

2. Providing context leads to higher correlations than excluding the context ($t(1181)$=9.27, p<.001)
3. Correlations reliably increase with the size of the rater pool ($t(1181)$=7.00, p<.001)
4. Although numerically higher correlations are obtained with 8 songs than with 4 songs, the difference is not reliable ($t(13)$=0.52, p>.05)
5. Interestingly, the rater type $\times$ number songs interaction is reliable ($t(1181)$=8.14, p<.001), reflecting the fact that human raters perform as well or better than the LLM raters for 4 songs, whereas the pattern is flipped for 8 songs where the LLM raters perform better.

The interaction between the human raters and LLM raters with whether there are 4 or 8 songs, might be caused by the fact that with 8 questions the recommendations need to go past the most "obvious" choice which is likely to require more knowledge of songs to assess the quality of the playlist. In this case, the LLM rater can leverage its knowledge of the specific songs and their relationship to the query better than people.

4 Conclusion

This study explored using LLMs as a proxy for human ratings for the setting of custom playlist creating. In particular, we studied how effectively an LLM with basic CoT prompting can predict human ratings, and also leverage knowledge of a user's music likes and dislikes. We compared the performance of LLM AutoRaters, with and without access to user musical preferences, to a set of human raters in predicting user satisfaction with recommended playlists when the playlist length is either 4 or 8 songs.

We found the AutoRater to be as effective if not more so than human raters, and that providing user context leads to higher correlations for both. Also the correlations reliably increase with the size of the rater pool. These results were statistically reliable. Although numerically higher correlations are obtained with 8 songs than with 4 songs, the difference is not statistically reliable. Interestingly, interaction between the rater type crossed with the number songs is statistically reliable reflecting that human raters perform as well or better than the AutoRaters for 4 song playlists, whereas the pattern is flipped for 8 songs playlists where AutoRaters can use broad knowledge of music to make more nuanced ratings.

Future research could explore more sophisticated prompting strategies, fine-tuning on larger and more diverse datasets, or incorporating additional modalities like audio features to further enhance the performance of LLM AutoRaters.

5 Acknowledgements

We thank Yuri Vasilevski for all of his help with the LLM AutoRating platform. In alphabetical order, by first name, we'd like to additionally thank the following for feedback and correspondence on the study, results, as well as paper editing and comments: Ajit Apte, Carrie Cai, Chris Tar, Daphne Domansi, Dima Kuzmin, Fei Sha, Hardik Patel, Hubert Pham, Jeremiah Liu, John Anderson, Josh Lee, Lora Aroyo, Marco Zagha, Muqthar Mohammad, Oliver Siy, Sean Li.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine

Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

[4] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[5] Y Dubois, X Li, R Taori, T Zhang, I Gulrajani, J Ba, C Guestrin, PS Liang, and TB Hashimoto. Alpacafarm: a simulation framework for methods that learn from human feedback. arxiv. *Preprint posted online on May*, 22:2023, 2023.

[6] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

[7] Sally Goldman and Yuri Vasilevski. Roborater: Automating ratings of task-oriented conversations using llms.

[8] Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, and Paul Denny. Open source language models can provide feedback: Evaluating llms' ability to help students using gpt-4-as-a-judge. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, pages 52–58. 2024.

[9] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

[10] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[11] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[12] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Yu-Ching Hsu, Jia-Yin Foo, Chao-Wei Huang, and Yun-Nung Chen. Two tales of persona in llms: A survey of role-playing and personalization. *arXiv preprint arXiv:2406.01171*, 2024.

[13] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

[14] Xiaoyu Zhang, Yishan Li, Jiayin Wang, Bowen Sun, Weizhi Ma, Peijie Sun, and Min Zhang. Large language models as evaluators for recommendation explanations, 2024.

[15] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena." arxiv. *arXiv preprint cs.CL/2306.05685*, 2023.

A Appendix

The LLM prompt used when there user context is provided is shown below, where

- personal_music_likes,
- personal_music_dislikes,
- query, and
- playlist

are replaced by the value for the query being evaluated.

```
  You are an expert music analyst with deep knowledge of diverse genres,
    artists, and music history.
  Your task is to rigorously assess the accuracy and quality of a music
    playlist recommendation for this user.
  Imagine you are a user who is looking for a playlist. These are situations
    in which you might find yourself listening to music for 10+ minutes.
  These are the expressed musical tastes of the user:

  --- BEGIN USER MUSIC LIKES ---
  personal_music_likes
  --- END USER MUSIC LIKES ---

  --- BEGIN USER MUSIC DISLIKES ---
  personal_music_dislikes
  --- END USER MUSIC DISLIKES ---

The User and the Music Service have the following conversation:

--- BEGIN CONVERSATION ---
User: query
Music Service: response
--- END CONVERSATION ---

  Rate the Playlist Result. Your task is to provide a detailed and nuanced
    assessment of different aspects of the Music Service playlist's quality,
    considering how well it aligns with the user query and their musical
    preferences.


  Question:
    Overall, the user likes the returned result for this query.
     - Strongly Disagree
     - Agree
     - Disagree
     - Strongly Agree
     - Neither Agree or Disagree

  Please think aloud about how would you answer this question and provide
    your reasoning prior to answering.

    Answer:
```

In the situation where no context is provided the following portion of the above is not included:

```
  These are the expressed musical tastes of the user:

  --- BEGIN USER MUSIC LIKES ---
  personal_music_likes
```

```
--- END USER MUSIC LIKES ---

--- BEGIN USER MUSIC DISLIKES ---
personal_music_dislikes
--- END USER MUSIC DISLIKES ---
```

and

```
Rate the Playlist Result. Your task is to provide a detailed and nuanced
  assessment of different aspects of the Music Service playlist's quality,
  considering how well it aligns with the user query and their musical
  preferences.
```

is replaced by

```
Rate the Playlist Result. Your task is to provide a detailed and nuanced
  assessment of different aspects of the Music Service playlist's quality,
  considering how well it aligns with the user query.
```