https://doi.org/10.36315/2025v2end001
Education and New Developments 2025

EXPLORING THE USE OF ARTIFICIAL INTELLIGENCE IN RUBRIC PRODUCTION AND DETECTION OF REVIEWER'S BIAS

Juuso Rytilahti, Erkki Kaila, & Erno Lokkila
Department of Computing, University of Turku (Finland)

Abstract

This study explores the use of artificial intelligence in education from a unique angle, providing insight into whether a large language model (LLM) can be used to create a rubric or detect a scoring bias in teachers' evaluation. For this, we examine a specific, publicly available model by OpenAI, called GPT-4o. The study was conducted using student submissions for assignments in an introductory programming course. As the first approach, we provided the selected LLM with a sub-set of submissions with feedback from differently graded answers to produce a rubric. The second approach was to provide the LLM with a reference answer to produce the justification for grading from "nothing". Finally, we tried combining these two approaches. We also tested producing the rubric with the exercise description. After the AI model output grading justifications, we compared the justification produced by LLM to ones created by teaching personnel. We also tested if a bias in reviews could be detected. Our results indicate, that there is a lot of variances in the AI-generated rubrics, and in general they are stricter than the human-generated ones. Moreover, we found that utilizing the exercise description often produced better results. Regarding bias detection, the model seemed to be mostly able to detect bias, if only one answer was examined at the time. If multiple answers were introduced at once, this led to a considerable inability to detect incorrect reviews. The insights gained from this study can guide tool development and make teachers more aware of the possibilities and limitations of utilizing LLMs in assessment.

Keywords: *Large language models, automatic assessment, AI in education.*

1. Introduction

There is always a human factor involved when student submissions are reviewed by teaching personnel. All people have their own biases, and that will affect the reviewer's output, thus potentially affecting a grade given to a student. In this study, we discuss the possibility of reducing bias, reducing incorrectness, and increasing uniformity in the grading done by different assessors. Although the reviewer bias has a bigger role in more subjective areas, such as creative writing, it applies to more content-focused student submissions, such as programming exercises, as well.

While it is typical to automatically assess programming tasks at the introductory level (see Paiva et al. 2022), students of varying skill levels can also benefit from exercises that do not require that the code be compiled. Such examples include for example designing and commenting programs as well as evaluating the quality of the code. This creates a need for human reviewing instead (or in addition to) automatically assessed tasks.

In this paper, we explore the potential of LLM generating evaluation rubrics automatically, and to use that to automatically detect biases in the human reviewers' gradings. We focus on the following research questions:

1. Can a large language model produce good evaluation criteria for novice-level programming exercises?
2. Is it viable to use a large language model for detecting reviewer bias in programming exercises?

2. Background and related work

LLM performance is affected by different factors, such as the model itself, parameters, e.g., temperature, used vocabulary, language, and even the order of presented options seem to affect the performance of models. However, the effects are not always straightforward e.g., Renze et al (2024) studied the effect of temperature on the problem-solving performance of LLMs and found that changing the temperature did little to the problem-solving capabilities of LLMs. Deng and Ling (2022) also point out in

their overview of an earlier ChatGPT model (GPT3-.5), that an LLM output could include biases present in the model's training data.

LLMs have several limitations, for example, the output of an LLM can include the so-called hallucinations. Xu, Z., Jain, S., & Kankanhalli, M. (2024) defined hallucination as a situation where the LLM output includes content that sounds plausible but is factually incorrect or nonsensical information. LLMs might also resort to utilizing shortcuts, a phenomenon where LLMs resort to utilizing spurious correlations within the input, see for example, Tang et al. (2023) for a more detailed analysis of the factors affecting the LLM tendency to utilize shortcuts. For a more concrete example, Han et al. (2024) noticed how spurious correlations in the input decreased the LLM performance in simple reasoning MCQs significantly. For example, when in the examples presented in input there was one option always correct (e.g. option A), the model was inclined to select the same option even in trivial questions, even though the option was incorrect. Performance decrease can also be caused e.g., position bias, Alzahrani et al. (2024) noted a performance decrease in their study of LLM performance in multiple-choice-question (MCQ) benchmarks when e.g. the symbol or order of presented options was changed. A similar effect was noticed by Jiang et al (2024) when assessing the mathematical reasoning capabilities of the models. They note, that LLMs are not able to formally reason. Although performance in MCQs is not directly applicable to other tasks, these examples highlight the challenges current state-of-the-art LLMs have.

Wu et al. (2024) examined how the rubric produced by humans vs. LLM differs and found that notable alignment gaps between the two exist. They utilized Mixtral-8x7B-instruct in their study (Jiang et al. (2024)). They also note that adding human responses to the model's input when producing analytical rubrics seems to reduce the similarity of the generated rubrics to the human-created ones. They also note that the model seems to tend to use shortcuts on rubric generation, focusing on specific keywords instead of deeper analytical meaning, and they warn that this tendency could be in turn transferred to students as well, leading students to focus on including specific keywords instead of the actual content of their written works.

3. Research setup

The course utilized in this study is called Fundamentals of Programming, a programming course where students are taught the basics of programming. Such courses are often referred to as CS1 courses in the literature (see e.g., Nagappan 2003). As the name suggests, the course contains the basics of programming, focusing on initial concepts, e.g., variables, loops, and programmatic thinking. The programming course used in the course is Python. The course is 8 weeks long. On the course, there are automatically assessed tutorial exercises and human-graded demonstration exercises. There are in total eight different demonstration assignments, divided into sets of two. The demonstration assignments start from week 4 and a new set is introduced weekly.

The examined exercises were quite diverse, with themes depending on the concepts students were introduced to that week. In the first demonstration, the assignments were about creating a simple program from scratch or completing a simple program with missing implementations. In the second demonstration, the exercises revolved around fixing syntax and logical errors. In the third demonstration, students were introduced to planning programs. The final demonstration introduced students to an external programming library, Pygame, and the main theme was to familiarize students with utilizing external libraries.

The human-made rubrics for assignments were very content-focused, as all of the exercises utilized in this study were programming exercises. The rubrics contained a rather detailed explanation of the grading, clearly indicating which faults in the student submission caused a reduction in the given points. The rubrics had been created by an experienced programming teacher.

For the rubric production, the following prompt was given to the LLM: *"Based on the reference answer and student submissions below, create content-focused evaluation criteria. The maximum amount of points given from the exercise is 10 points. The students already know <topic keyword list> <reference answer>*". The beginning of the prompt was the same also when the student submission was excluded. We also decided to try rubric generation with exercise descriptions, as in most of the exercises they produced rubrics more aligned with the human-made rubrics. Rubric generation was tested with all of the demonstration assignments.

For detection of the reviewer's bias, the following prompt was used: "*Are these evaluations accurate to the rubric? rubric:<rubric> answers: <answers>*". The rubric used was always human-made and not AI-generated. Bias detection was tested with exercises used in demonstrations from weeks 2, 3, and 4. In bias detection, we mainly utilized one assignment from each week but increased the minor modifications made by adding varying orders, and amount of examples, and increased amount of different chat threads to verify the result.

Data was collected from students who gave explicit, opt-in consent to utilize their data as part of research, and the data was anonymized before it was used. This research is conducted utilizing the temporary chat feature of the browser version of the ChatGPT. The study was conducted through a paid subscription account in February of 2025. OpenAI states that "*Temporary Chats won't appear in your history, and ChatGPT won't remember anything you talk about. For safety purposes we may still keep a copy for up to 30 days.*" (accessed: February 2025[*]). They also state that temporary chats won't be used for improving their models. Additionally, we note that, unlike the normal user UI, temporary chats do not transfer any information between different chat threads.

4. Results

The LLM-produced rubrics were quite detailed. For example, for the first assignment, the rubric contained four sections: basic functionalities (3 points), arithmetic operations (3 points), error handling and input validation (2 points), and user experience and clarity (2 points). The point amounts were also automatically assigned by LLM. For each category, the model listed 1-point rules for grading (such as "The program correctly exits when an empty input is given. (1 point)".

On a general level, there was a lot of variance between rubrics, even on the same question between different chat threads. The assigned scores differed between rubrics quite a lot. However, especially in rubrics produced by using the reference answers, the common structure was almost always similar. The scores had some variance between different threads. This can be mostly credited to the higher temperature setting of the UI version.

The human-made rubrics were often more forgiving, especially on the later assignments. For example, on one of the tasks, the rubric instructed awarding points for producing an answer that worked only with the exact example input values mentioned in the assignment. The model-produced answer did not include this rule.

On the second set of assignments, where exercises were focused on fixing different syntactical and logical errors, the produced rubric contained the main themes of correcting syntax errors, logical errors, and input handling corrections. However, the rubric still contained rules for code structure and readability. The broken program given as input for the assignment contained many mistakes, but its structure did not require fixing.

In the third set, the main themes of the rubrics were similar to human-made rubrics. If a reference answer was used containing a detailed description of the code, the outputted rubric was aimed at implementing the requirements, instead of focusing on planning and design of the program. Implementing a new feature to the existing structure aligned better with the original human-made rubric

On the fourth set of assignments, the main focus on the human-made rubrics was on implementing the requested functionality, and no emphasis was put on the code structure, as this was a challenging task already. However, while the AI-generated rubric from the assignment description contained these features, on an assignment revolving around creating and moving around snowmen, the rubric mentioned rewarding additional features and creativity. This was similar in both generated rubrics.

Regarding the use to detect reviewer bias, when utilizing examples with minor mistakes, the model hallucinated on some reasoning, but the final verdict was correct. On most occasions, the model did go through each answer and checked each point mentioned in the rubric separately. The overall performance seemed good. However, this was a wrong assessment. If an incorrectly evaluated answer was the only one in the input, it was usually correctly identified and received correct point reductions. However, when the incorrect input was part of a large input set with multiple other answers, the other answers affected the evaluation, resulting in missed corrections. This seemed to be a consistent behavior. Even when the wrong justification for a (too high) score of an answer was correctly identified, the number of points reduced was inconsistent between different threads with identical input.

5. Limitations

The experiments conducted were qualitative. Additionally, the used data consisted of programming exercise submissions. To produce more generalizable results, a more quantitative approach should maybe be used. LLMs' ability to detect reviewer bias is likely highly dependent on the model used, the prompt, exercise type, etc. This study noticed the effects of LLMs' tendency to shortcut when trying to detect reviewer bias, however, these results are highly preliminary due to the small sample size on a specific domain. To say anything definitive about the subject, a larger-scale experiment should be conducted.

[*] https://help.openai.com/en/articles/8914046-temporary-chat-faq

6. Discussion

One possible problem with utilizing LLM is over-reliance on the tool. Reviewers have different biases, and of course, as noted before, the review is often affected by human factors. The LLM provides an additional perspective, easily produced. It is important to note that hallucinations can and will still occur within the produced output, regardless of the input and model used.

Another problem is what is the appropriate amount of context. It is worth noting that even on courses focusing on the fundamentals of programming (CS1) there are often differences between universities in the order the different topics are presented to students. If AI-generated rubrics are to be used as is (e.g. to produce evaluation criteria for adaptive learning environments) the input to the model should include the topics the student is familiar with. It should be noted that in our experimentation mentioning the skill level at a very shallow level (e.g., "functions", "variables", "loops") did not seem to change the produced rubrics that much. A more fruitful route could be also to include the topics that the student has not yet mastered, as this could align the model better with the goal.

On rubric generation, depending on the exercise the results can differ quite much. The exact number scoring should not be treated as ground truth, as the variance was quite high between the different chat threads. When asking a model to produce a rubric from a reference answer, the structure of the generated rubric was often similar between different assignments. However, quite often the rubric revolves around the function names used in the answer. Sometimes this is preferred, but often this can also seen as a surface-level approach. If a student is asked to produce a program, the focus should be on using suitable features instead of forcing a student to follow a certain structure or naming convention in the program.

In the fourth week, the model's inclusion of the "creativity" criteria could be seen as quite problematic. It should be noted that the exercise description contained a mention "*... of course, you can decorate the figure however you lik*e". This means that the decoration was optional, and should not be included in the rubric itself. The utilized reference answer also contained a similar mention as a code comment, as well as an example of such a feature. These, as well as the theme, could be the reasons for such a misalignment, highlighting the importance of inspecting the input to only include text relevant to the aimed goal. It should be noted, that providing the LLM with additional instructions on creating the rubric could help with this.

All in all, the results display a clear misalignment between human-made and LLM-produced rubrics. The LLM seems to easily be misaligned. However, the rubrics seemed to produce within rather consistent structure, and themes. If a human oversees the rubric production, and critically evaluates the result before utilizing the result, the LLM-produced rubrics can produce a good starting point. If used fully, or even semi-autonomously, careful consideration and testing on the specific type of exercise is required to reveal if the outputted rubric is suitable for purpose. Additionally, one could argue that the creation of a rubric is a task requiring advanced reasoning, so the newer models with improved reasoning capabilities may produce better results.

Regarding the detection of reviewer bias, we noticed a similar effect as Han et al. (2024). When multiple examples were introduced, the output seemed to be highly affected by the previous answers that appeared in the input. However, as mentioned in the results, the model seemed to be able to detect incorrectly reviewed answers much more often when they were introduced one by one instead of evaluating a group of answers at once. We also would like to note, that to confirm these results, the experiment should be duplicated with a larger amount of data. Of course, in this case, the performance can also be highly dependent on the rubric, and as we noted in our previous studies (Rytilahti, Kaila & Ingman, 2025), LLM will make mistakes when automatically assessing these types of demonstration assignments, indicating that the current models are not (at least yet) ready to assess these assignments autonomously. However, our results seem to indicate that an LLM can be used as an additional tool to help detect incorrect evaluations of a single review.

7. Conclusion and future work

Regarding the question, of whether AI can produce good evaluation criteria for novice-level programming exercises, the answer is: "Sometimes". Utilizing an exercise description produced better results than the reference answer, as the model could better align with the aimed learning goals. Additionally, the produced rubrics tend to overscope when compared to the human-made ones. Utilizing only reference answers creates shallow rubrics more often than providing LLM with an exercise description.

Regarding the detection of reviewer bias, it seems to be somewhat possible, as long as the rubric is utilized and only one answer at a time is evaluated. However, this is of course highly dependent on the answers being evaluated, and probably also the type of mistake made and the exercise's type. If multiple

answers were evaluated in one prompt, the model's ability to detect incorrect answers was severely impaired by the effect of the other answers present in the input. Our preliminary results indicate that an LLM can be a helpful tool to detect bias, but it should be treated as an additional assistant to human reviewer instead of relying upon it to complete the whole process.

The future work should include looking into more subjective areas, as well as different kinds of answers including essays and reports. Additionally, a look into how tuning the settings of the LLM (for example the temperature) affects the output should be examined. A more quantitative approach to rubric evaluation could be fruitful, including measuring the semantic similarity (Wu et al. 2024). Furthermore, more research is needed to include the recent advancements in the field of LLMs, including recent models such as DeepSeek r1 or OpenAi o1. Finally, more diverse prompting strategies, as well as the effect of different combinations in the input should be examined.

Acknowledgements

This work has been supported by FAST, the Finnish Software Engineering Doctoral Research Network, funded by the Ministry of Education and Culture, Finland.

References

Alzahrani, N., Alyahya, H. A., Alnumay, Y., Alrashed, S., Alsubaie, S., Almushaykeh, Y., ... & Khan, H. (2024). *When benchmarks are targets: Revealing the sensitivity of large language model leaderboards*. arXiv preprint arXiv:2402.01781.

Deng, J., & Lin, Y. (2022). The benefits and challenges of ChatGPT: An overview. *Frontiers in Computing and Intelligent Systems*, *2*(2), 81-83.

Han, P., Song, P., Yu, H., & You, J. (2024). *In-context learning may not elicit trustworthy reasoning: A-not-b errors in pretrained language models*. arXiv preprint arXiv:2409.15454.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., ... & Sayed, W. E. (2024). *Mixtral of experts*. arXiv preprint arXiv:2401.04088.

Jiang, B., Xie, Y., Hao, Z., Wang, X., Mallick, T., Su, W. J., ... & Roth, D. (2024). *A Peek into Token Bias: Large Language Models Are Not Yet Genuine Reasoners*. arXiv preprint arXiv:2406.11050.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM Sigcse Bulletin*, *35*(1), 359-362.

Paiva, J. C., Leal, J. P., & Figueira, Á. (2022). Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education (TOCE)*, *22*(3), 1-40.

Renze, M., & Guven, E. (2024). *The effect of sampling temperature on problem solving in large language models*. arXiv preprint arXiv:2402.05201.

Rytilahti, J., Kaila, E., & Ingman, V. (2025). *Large language model performance in automatic assessment on an introductory programming course*. Paper submitted to ITiCSE 2025, Nijmegen, Netherlands.

Tang, R., Kong, D., Huang, L., & Xue, H. (2023). *Large language models can be lazy learners: Analyze shortcuts in in-context learning*. arXiv preprint arXiv:2305.17256.

Wu, X., Saraf, P. P., Lee, G. G., Latif, E., Liu, N., & Zhai, X. (2024). *Unveiling scoring processes: Dissecting the differences between llms and human graders in automatic scoring*. arXiv preprint arXiv:2407.18328.

Xu, Z., Jain, S., & Kankanhalli, M. (2024). *Hallucination is inevitable: An innate limitation of large language models*. arXiv preprint arXiv:2401.11817.