Evaluating Language Models for Generating and Judging Programming Feedback

Charles Koutcheme
Aalto University
Espoo, Finland
charles.koutcheme@aalto.fi

Nicola Dainese
Aalto University
Espoo, Finland
nicola.dainese@aalto.fi

Sami Sarsa
University of Jyväskylä
Jyväskylä, Finland
sami.j.sarsa@jyu.fi

Arto Hellas
Aalto University
Espoo, Finland
arto.hellas@aalto.fi

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Syed Ashraf
Aalto University
Espoo, Finland
syed.ashraf@aalto.fi

Paul Denny
The University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Abstract

The emergence of large language models (LLMs) has transformed research and practice across a wide range of domains. Within the computing education research (CER) domain, LLMs have garnered significant attention, particularly in the context of learning programming. Much of the work on LLMs in CER, however, has focused on applying and evaluating proprietary models. In this article, we evaluate the efficiency of open-source LLMs in generating high-quality feedback for programming assignments and judging the quality of programming feedback, contrasting the results with proprietary models. Our evaluations on a dataset of students' submissions to introductory Python programming exercises suggest that state-of-the-art open-source LLMs are nearly on par with proprietary models in both generating and assessing programming feedback. Additionally, we demonstrate the efficiency of smaller LLMs in these tasks and highlight the wide range of LLMs accessible, even for free, to educators and practitioners.

CCS Concepts

• **Social and professional topics** → **Computing education**.

Keywords

open source, large language models, generative AI, automatic feedback, automatic evaluation, programming feedback, LLM-as-a-judge

ACM Reference Format:
Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, Syed Ashraf, and Paul Denny. 2025. Evaluating Language Models for Generating and Judging Programming Feedback. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641554.3701791

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA

ACM ISBN 979-8-4007-0531-1/25/02
https://doi.org/10.1145/3641554.3701791

1 Introduction

High-quality and timely feedback is essential for students in programming courses. Some types of feedback, such as whether a program runs or passes a provided test suite, are readily available via simple automated means [13, 28]. However, feedback on the causes of subtle programming errors and suggestions for resolving them can be difficult to produce [14]. Especially in large classes, providing accurate and personalised explanations of bugs as feedback to students can be a manual and time-consuming task for educators, and yet essential for reducing frustration and aiding learning.

The automated generation of human-like feedback has recently been made possible thanks to the accessibility of state-of-the-art generative AI tools, such as ChatGPT. In particular, API access to powerful large language models (LLMs) has sparked the development of many programming feedback tools that are now being deployed in classrooms [33]. These include tools for generating improved error messages [22], aiding real-time debugging [12], explaining code [21, 23] and tailoring next-step hints [35]. Such systems have been successful not only in generating feedback, but also in assessing feedback quality, offering the potential for generating high-quality feedback through iterative improvement.

Despite the promise of LLM-based feedback generation and evaluation approaches, the vast majority of research and usage in computing education contexts has relied on proprietary models such as GPT-4. This reliance on closed-source LLMs is concerning for several reasons. It involves sending potentially sensitive data to third parties with no guarantees on how the data will be used, provides little insight into how the models are trained or what deliberate or inadvertent biases they may contain, and may come with unpredictable licensing expenses [20]. Open-source LLMs, on the other hand, are freely accessible and open for modification and distribution, and have started to become viable alternatives. Nevertheless, very few studies have explored their capabilities for providing or assessing programming feedback.

In this work, we investigate the potential of open-source models to produce high-quality feedback, and assess the quality of feedback generated by other LLMs. We focus on feedback that provides explanations of bugs or issues in student-written programs and outlines the steps to address these issues. While prior research suggests that open-source language models are competitive alternatives to proprietary models for generating feedback, the extent to which they can serve as judges (validators) of such feedback remains unclear. Using a publicly available benchmark dataset of student-written programs, we address the following two research questions:

RQ1 How do open- and closed-source models compare with respect to the quality of their generated bug explanations and suggested fixes?

RQ2 To what extent can open- and closed-source models assess the quality of programming feedback generated by other models relative to expert human judgment?

To answer our first research question, we generate explanations of bugs and their corresponding fixes using five state-of-the-art open-source and three popular proprietary language models. We manually evaluate this feedback using a custom rubric that includes the completeness and comprehensibility of the explanations and the accuracy of the suggested fixes. To answer our second research question, we use these expert human-generated ground truth labels to evaluate the performance of the models on the task of judging the programming feedback.

Our findings suggest that open-source language models are competitive with proprietary models for both generating and assessing programming feedback. Given the potential benefits of open-source models in terms of transparency, trust, and cost, we argue that they should be increasingly adopted in computing education contexts.

2 Related Work

2.1 Using Language Models For Feedback

Automating assessment of programming exercises and providing feedback on the exercises have been studied for decades within the computing education research domain [14, 27, 28]. Classically, much of the existing work on automating feedback has focused on informing students about mistakes in their code, while providing formative feedback has been less common [14]. Providing suggestions on the location of the issue or hints on how to fix the issue can improve students' performance over just pointing out that a test failed [7], but manually creating quality feedback can be very time-consuming.

The recent emergence of powerful language models has led to researchers exploring their capabilities for programming feedback [3, 5, 9, 15, 22, 23, 29, 30] and, in general, the observations on the quality or utility of feedback has evolved with the introduction of better language models [9]. As an example, GPT-3 had high variability in the quality of feedback, at times generating incorrect and inconsistent feedback [3], while GPT-3.5 would often provide meaningful feedback and find issues in code, but also often hallucinate issues that were not present in the code [9]. Language models are also better at detecting some types of errors than others [9, 15], being useful, especially for providing feedback on syntax or compilation errors [15, 22, 30]. Despite the advances, even the state-of-the-art models like GPT-4 are still not on par with humans when generating feedback for programming exercises [31]. At the same time, there are increasing amounts of evidence that language model-powered feedback systems and chatbots in programming [8, 10, 23, 38] can aid students, at least when the programming languages or used frameworks are not brand new [8].

Most existing work on language models for programming feedback in the computing education research context has focused on utilizing proprietary models (mainly from OpenAI). In contrast, the use of open-source models has received only little attention. Calls for increasing the use of open-source models have been voiced [39], already due to potential privacy issues related to sharing student data with language model providers. Work on utilizing open-source models for the task is also starting to emerge [10, 16, 25], where one of the research aspects has been contrasting the performance of open-source models to the proprietary ones; researchers have already observed that open-source models are on par with models such as GPT-3.5-Turbo for programming feedback [18].

In our work, our first research question re-investigates how various language models, including open-source ones, perform in explaining issues in student programs and providing fixes, complementing prior studies.

2.2 Using Language Models as Judges

The idea of using an LLM to judge the output of other LLMs – LLMs-as-judges – was first studied in the work of Zheng et al. [40], showing good promise, but also limitations, e.g., in grading math and reasoning tasks. Since then, GPT-4 has been used in multiple studies as a judge of the quality of other LLMs' generations [26], also in educational contexts [10, 18]. Moreover, the reliance on GPT-4, a proprietary model, has sparked interest in leveraging other open-source language models to act as judges [4]. Yet, recent work has highlighted the limitations of relying on a single language model for evaluating the quality of other language models' outputs, and suggested employing a diverse ensemble of smaller models from different LLM families as a jury for cheaper and less biased evaluations [37]. When answering our second research question, we test this hypothesis by comparing the usage of single judges (both open-source and proprietary) and that of a jury of smaller open-source language models.

3 Methodology

In this section, we describe our methodology for answering our two research questions. We first introduce the dataset used in our evaluations, then our methods used for answering RQ1 and RQ2.

3.1 Dataset

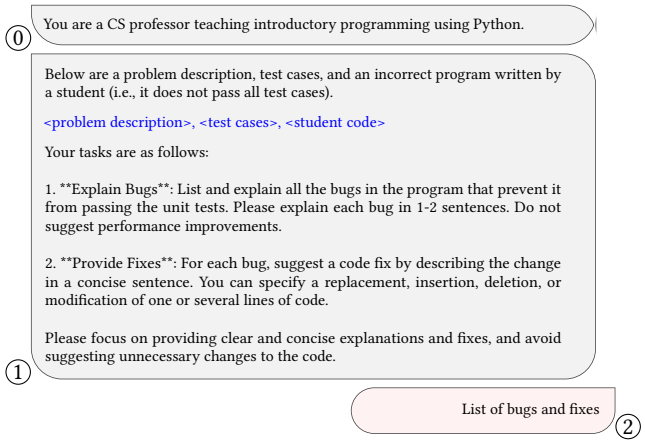
We use data from the Socratic guidance benchmark [2], which consists of 57 introductory-level programming assignments requiring students to write functions. Each of the assignments is accompanied by the associated test cases, a unique incorrect student solution, the ground truth description of a single bug in the program, a list of potential bug fixes, and several conversation threads between a fake student and a teaching assistant. The ultimate goal for the benchmark is evaluating LLMs' ability to help students using the socratic method, i.e., guiding students in finding a solution on their own, by asking a series of relevant questions that help their reasoning.

However, for this study, we focus solely on identifying the issues in the code and any required fixes, as it is a fundamental step for effective use of LLMs for socratic guidance – models incapable of identifying issues in students' programs would be likely to provide them erroneous guidance. We leave the problem of evaluating LLMs for socratic guidance for future work.

3.2 Generating High-Quality Feedback

Given a student's incorrect program, our goal regarding RQ1 is to evaluate LLMs' ability to provide two particular types of feedback: **an explanation of the bugs** in the student's program and **suggested fixes** for the found bugs.

Feedback Generation. We prompt the models to provide feedback according to the following example:



To elaborate, we provide a language model: ⓪ a system prompt and ① a description of the task (with all the necessary contextual information), which results in output ②.

Feedback Language Models. We consider the following open-source models: Gemma-2B [6], Phi-3-mini [1] (3.8B parameters), Mistral-7B [11], Llama-3.1.1-8B [4], Llama-3.1.1-70B [4]. We chose these models because of their extensive documentation, community adoption, strong performance on code and language reasoning benchmarks (e.g., MMLU), their parameter count, and their ability to follow instructions. This selection covers the recent state-of-the-art models from various companies across the most used model sizes for LLMs. Furthermore, we also evaluate three of OpenAI's proprietary flagship models, GPT-3.5-turbo, GPT-4o-mini, and GPT-4o, which represent the current industry standards.

We query proprietary models using the OpenAI Python library, and open-source ones with HuggingFace Transformers Python library to simplify querying them through the HuggingFace Inference API. All models are evaluated using greedy decoding [18]. Next, we explain the annotation process before detailing the grading rubric.

Annotation. We use the eight models presented above, and the 57 programs of the benchmark, which results in 8 × 57 = 456 model outputs. To answer our first research question, two annotators (two paper authors who are expert Python programmers) annotated all 456 model outputs. First, we selected 11 problems out of the 57 available problems using the manual annotation subset presented in [2]. Then, the two annotators independently annotated 8 × 11 = 88 model outputs with an initial description of each grading criterion on this subset. We then computed an inter-annotator agreement score using Cohen's Kappa coefficient. The resulting annotation process yielded a moderate inter-rater agreement of 0.54. After comparing annotations and resolving conflicts, the remaining feedback examples were split equally between the two annotators. The final annotated dataset formed the basis for evaluating the quality of the feedback generated by the language models.

Grading Criteria. During the final annotation phase, each expert used the following grading criteria for evaluating the quality of a single generated bug explanation (E), and the quality of the generated fixes (F):

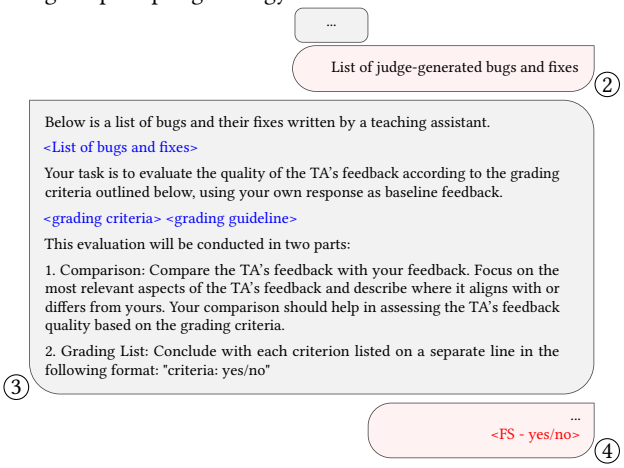
- EA - Explanation Accurate: the explanation identifies and correctly explains the bug in the student program.
- ES - Explanation Selective: the explanation does not mention non-existent (or non-relevant) bugs or issues.
- EC - Explanation Clear: the explanation is easy to understand for a novice programmer, presented in a readable format and contains the right amount of information. Note: this criterion is independent of the correctness of the explanations.
- FA - Fixes Accurate: the required bug fixes are laid out and explained.
- FS - Fixes Selective: no unnecessary or irrelevant changes are outlined;
- FC - Fixes Clear: the proposed fixes are succinct and mention the unique changes to perform in the code.

These criteria extend prior work [9, 18, 32]. Criteria EA, ES, FA, and FS (resp. EC, and FC) represent how correct (resp. how understandable) the explanations and criteria are. The annotators used the following guidelines: for each ground truth bug (provided in the dataset), match the bug with one (or several) model-generated explanations. If any generated model bug descriptions did not match the ground truth, we set the criteria ES to false, unless the model explicitly noted the irrelevance of the unmatched bug. Then, independently of the correctness, we looked at whether or not a novice programmer unaware of the real issues could understand the meaning of the provided bug description. We follow the same strategy for the fixes. Moreover, for the clarity criterion, we ensure that the fixes provide clear descriptions of changes with snippets or at least highlight changes in a repaired program (if present).

3.3 Automatic Feedback Evaluation

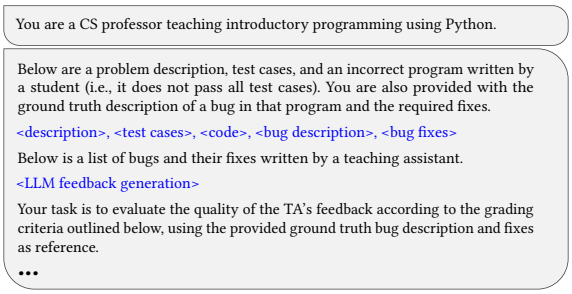
In this subsection, we present the methods we used to automatically evaluate the quality of LLM-generated feedback using other language models (answering RQ2). We explored two approaches: a single LLM as a judge, and an ensemble of LLMs as a jury on two scenarios, depending on whether a reference answer is available or not. We first describe how we generate the responses to the grading criteria using a single LLM as a judge for the two scenarios. We then outline our ensemble of LLMs and how we obtain the jury annotations.

No Reference Answer Available. Given the feedback generated by a language model, we prompt another language model (the judge) to grade this feedback (according to the criteria outlined in Section 3.2) using the prompting strategy shown and described below:



Before asking the judge language model to grade the quality of the feedback ③ (using the grading criteria list as part of the prompt), we first ask the judge to generate its own descriptions of the bugs and fixes in the student program, as described in Section 3.2 (⓪①②). This strategy is a form of zero-shot-chain-of-thought Single Answer Grading (SAG) [40], i.e., the judge uses its solution to the "problem" as a reasoning step to grade the solution of another language model. This first scenario without a reference answer is applicable for educators and practitioners interested in evaluating language models' feedback abilities on their private datasets without the need for ground truth annotations.

Reference Answer Available. Educators may also be interested in evaluating their language models on existing benchmarks containing ground truth annotations of issues. Although such benchmarks are not abundant, we expect more to come as educational AI advances and becomes more widespread. In consideration of this scenario, we experiment with providing the judge with the ground truth descriptions of issues, instead of the more error-prone approach of generating them using the judge itself. The example below illustrates our prompting strategy:



This prompting strategy is a form of reference grading [40]. We refer to this strategy as GAG (Ground truth Annotated Grading). We note that for both strategies (SAG, GAG), we extract the response to each grading criterion from the final judge model output.

Ensemble of Judges. Prior work suggests that using a single LLM as a judge has limitations. For example, GPT-4 favours outputs from OpenAI's GPT line of models [34]. As mentioned in Section 2.2, instead of using a single LLM, Verga et al. [37] showed that using multiple language models from different families can address many of these issues. We aim to test their hypothesis in our educational context. We prompt three popular open-source language models to provide their judgement, and then, we combine the model decisions using majority voting separately for each criterion. For example, given three LLMs outputting 'yes', 'yes', and 'no' respectively for a given criterion, the final ensemble result will be 'yes'. We consider this "jury" both when reference answers are available (GAG) and not available (SAG).

Judge and Jury Language Models. We evaluate both proprietary and open-source models as judges. We use GPT-3.5-turbo and extend prior work [18] by also including GPT-4o, and GPT-4o-mini. We compare these proprietary models against state-of-the-art open-source language models Phi-3-mini, Llama-3.1-8B, Mistral-7B, and Llama-3.1-70B. For the jury, we use the three language models: Llama-3.1-8B, Mistral-7B, and Phi-3-mini. We selected these three strong LLMs due to them being at the top of the leaderboards for LLMs of their size, and being from different families of models. As with the feedback generation, we query these language models using the OpenAI and Huggingface Transformers Python libraries and obtain the outputs using greedy decoding.

Evaluation. To answer RQ2, we compare the evaluation of each judge/jury model for the 456 generated feedback outputs against the manually written ground truth annotations, with respect to our grading criteria (see Section 3.2). We then report the performance of each judge/jury across all of their outputs and for our two scenarios. We report the weighted average version of the $f_{0.5}$ score as in [18] (although they used plain $f_{0.5}$) for each judge/jury, as this metric accounts for false positives and potential class imbalances. We also report the kappa score to complement our observations.

4 Results

4.1 Generating Feedback

Table 1 shows the performance of each language model on various grading criteria, including both individual and grouped criteria, based on human evaluations. We make the following observations.

Table 1: Feedback results for various language models based on human evaluations. Legend (see also grading criteria in 3.2): E_{all} (resp. F_{all}): all explanations (resp. all fixes) related criteria are correct, ALL: all criteria are correct.

model	EA	ES	EC	FA	FS	FC	E_{all}	F_{all}	ALL
Gemma-2b	0.44	0.02	0.65	0.42	0.02	0.65	0.00	0.02	0.00
Phi-3-mini	0.72	0.19	0.93	0.74	0.21	0.89	0.18	0.19	0.18
Mistral-7b	0.70	0.23	**0.96**	0.70	0.25	0.93	0.21	0.21	0.19
Llama3.1-8b	0.70	0.04	0.74	0.68	0.05	0.72	0.04	0.04	0.04
Llama3.1-70b	0.89	0.28	0.89	0.88	0.40	0.89	0.26	0.37	0.26
GPT-3.5-turbo	0.84	0.35	0.93	0.81	0.37	0.84	0.32	0.28	0.28
GPT-4o-mini	0.96	0.35	0.93	0.93	0.37	0.96	0.30	0.35	0.30
GPT-4o	**0.98**	**0.44**	0.93	**0.93**	**0.42**	**0.98**	**0.39**	**0.42**	**0.39**

Open-Source vs. Proprietary Models. The GPT-4o and GPT-4o-mini models show very strong performance across nearly all individual and grouped criteria, beating all other models. Among the open-source models, there is significant variance in performance across different criteria. For instance, Llama-3.1-70B performs on par with or better than GPT-3.5-turbo on the accuracy and clarity of explanations (EA, EC), and the clarity of fixes (FC). Llama-3.1-70B remains also competitive with both GPT-4o-mini and GPT-4o for generating perfect explanations (E_{all}), and perfect fixes (F_{all}). In contrast, smaller models such as Gemma-2b perform poorly across the board. However, size alone does not determine performance, as other small open-source language models, like Phi-3-mini (with 3.8 billion parameters – slightly more parameters than Gemma-2b), despite their smaller size, perform decently well on several criteria, notably for generating accurate explanations and fixes (EA, FA).

Strengths and Weaknesses. Each model has its strengths and weaknesses. However, we notice that most models struggle with selectivity (i.e., they identify irrelevant issues or fixes), while they generally produce clear outputs (i.e., well-formatted and understandable responses). When looking at the feedback generations, the stronger models (e.g. Llama-3.1-70B, and the GPTs) often added performance suggestions (e.g. replace a for loop with a built-in function), while the other models often added incorrect outputs. This indicates a broader challenge in developing models that can effectively identify and focus on relevant issues without including redundant or irrelevant information. Improvements in this area could lead to substantial overall performance gains.

Explanation, Fixes, and Repairs. We can observe a direct relationship between models' abilities to explain issues and their ability to generate fixes: overall, language models that produce more accurate (resp., more selective) explanations tend to generate more accurate (resp., more selective) fixes. During our experiments, we also noticed that the generated feedback often included repaired programs after the fixes, even though we did not prompt the models for this. Although program repairs are not our primary focus, they represent another valuable form of feedback for students and can later be leveraged for hint generation [16]. Building on this, we hypothesize that a language model's performance in generating accurate and selective fixes provides insights into its ability to generate high-quality program repairs—repairs that are functionally correct and preserve the intent of the student's original code [19]. If our hypothesis holds, our findings complement prior work [17] suggesting that repair abilities may serve as a proxy for explanation quality in language models. This connection could enable educators to more easily identify language models suited for generating effective programming feedback.

4.2 Evaluating Feedback

Table 2 shows the results of the judgment task, detailing the $f_{0.5}$ scores and kappa scores for each language model under the two scenarios (SAG and GAG). We can make several observations from these results:

Open-Source vs. Proprietary Models. Without ground truth bug descriptions and fixes, LLama-3.1-70B shows superior judging ability compared to GPT-3.5-turbo and smaller open-source models. The model performs comparably to GPT-4o-mini and GPT-4o across many individual criteria. It even surpasses OpenAI's flagship models in average $f_{0.5}$-scores overall criteria (i.e., average scores across self-generated and other-generated feedback). While larger models generally perform better, smaller models such as Phi-3-mini and Mistral also demonstrate strong judging performance, rivalling GPT-3.5-turbo, a previous state-of-the-art model. These small models also rival top language models in terms of explanation accuracy (EA), explanation clarity (EC), and fixes clarity (FS). However, they still lag in terms of judging selectivity. Providing models with ground truth descriptions of bugs and required fixes (GAG setting) leads to significant performance gains (nearly 10%) over relying solely on their explanations (SAG setting). This improvement is most pronounced for models like Mistral, which struggle with selectivity in the SAG setting. In the GAG setting, the performance gap between top models (LLama-3.1-70B, GPT-4o-mini, and GPT-4o) becomes negligible across all criteria, indicating these models are viable alternatives to one another. However, in the GAG setting, Phi-3-mini's performance does not improve. When looking at the model reasoning process in that setting, the model appears to overly rely on exact matches with the ground truth, showing less flexibility in its evaluation. This behaviour might stem from the model's smaller size, which could limit its capacity for nuanced reasoning.

Kappa Scores. The low kappa scores in the SAG setting indicate that most model results could be due to random chance. When investigating the reasons for the scores, we see that most models tend to be overly positive, predicting 'yes' the majority of the time. This phenomenon aligns with observations in [18], highlighting a tendency of models to overestimate the quality of feedback. When provided with ground truth annotations, the results improve to reach a moderate level of agreement.

Self-Evaluation vs. Evaluation of Others. Models perform better when evaluating other models' outputs than their own, with Llama-3.1-70b performing the best. This might be due to a bias toward positive evaluations, especially within models of the same family, as noted in previous research [40]. Our result thus suggests that language models from different families should be used for generating and validating feedback [32].

Ensemble Performance. Combining multiple models into an ensemble does not improve judgment quality; instead, it biases the results. The ensemble approach, which combined the outputs of the models Phi-3-mini, Mistral-7B, and LLama-3.1-8B, did not yield better performance. This contrasts with previous work by Verga et al. [37], possibly due to the absence of few-shot examples, which provide the model with additional context and training data that could enhance the performance of the individual models. In the previous study, models competitive with GPT-3.5-turbo were used, leading to better ensemble performance. Moreover, our method of taking the mode of the generations from three LLMs means that if two weaker models consistently disagree with the stronger model, the output can be negatively biased. This highlights a key limitation in ensemble approaches: the quality of the ensemble is highly dependent on the individual models' performance and their ability to complement each other.

Table 2: Judging results.

(a) Detailed f0.5 scores. We show the SAG score and in parenthesis the GAG score. Legend: AVGO: (resp. AVGS) average f0.5 over all criteria when judging other models' (resp. the judge's own) feedback.

judge	EA	ES	EC	FA	FS	FC	AVGO	AVGS
Phi-3-mini	0.70 (+0.00)	0.41 (-0.03)	0.81 (-0.02)	0.65 (+0.00)	0.37 (-0.04)	0.78 (+0.00)	0.58 (-0.02)	0.41 (+0.09)
Mistral	0.67 (+0.07)	0.14 (+0.50)	0.81 (-0.02)	0.63 (+0.05)	0.19 (+0.42)	**0.81** (-0.04)	0.49 (+0.18)	0.44 (+0.13)
Llama-3.1-8b	0.63 (+0.12)	0.55 (+0.15)	0.75 (+0.04)	0.63 (+0.12)	0.56 (+0.09)	0.75 (+0.04)	0.60 (+0.11)	0.51 (+0.22)
Ensemble	0.68 (+0.06)	0.28 (+0.33)	0.81 (-0.01)	0.64 (+0.06)	0.33 (+0.24)	0.79 (+0.01)	0.53 (+0.14)	/
Llama-3.1-70b	0.69 (+0.17)	0.72 (+0.10)	**0.81 (+0.01)**	0.71 (+0.10)	0.71 (+0.10)	0.80 **(+0.03)**	**0.69 (+0.14)**	**0.69** (+0.14)
GPT-3.5-turbo	0.65 (+0.00)	0.09 (+0.19)	0.79 (-0.01)	0.65 (+0.01)	0.20 (+0.26)	0.78 (-0.01)	0.46 (+0.09)	0.51 (+0.08)
GPT-4o-mini	**0.74** (+0.08)	0.66 (+0.16)	0.75 (+0.01)	**0.74** (+0.05)	0.70 (+0.12)	0.76 (+0.02)	0.64 (+0.13)	0.61 (+0.24)
GPT-4o	0.72 **(+0.16)**	**0.78 (+0.09)**	0.74 (+0.04)	0.72 **(+0.12)**	**0.76 (+0.11)**	0.77 (+0.02)	0.68 (+0.12)	0.66 **(+0.26)**

(b) Kappa scores. SAG (+- GAG diff)

judge	kappa
Phi-3-mini	0.09 (+0.00)
Mistral	0.03 (+0.23)
Llama-3.1-8b	0.12 (+0.27)
Ensemble	0.06 (+0.20)
Llama-3.1-70b	**0.36 (+0.29)**
GPT-3.5-turbo	0.02 (+0.08)
GPT-4o-mini	0.25 (+0.27)
GPT-4o	0.34 (+0.26)

5 Discussion

Teaching and Learning Implications. While the educational community has been focused on leveraging proprietary models, our study aims to show that alternative solutions are accessible to educators and practitioners. In particular, Llama-3.1-70B performs on par with GPT-3.5-turbo for feedback generation and competes with GPT-4o for judging the quality of other language models generated feedback. Educators could leverage such a model in building their feedback tools or for benchmarking purposes [17, 31]. Notably, the size of a language model no longer correlates directly with performance. For example, a smaller language model like the Phi-3-mini competes with the 7B Mistral and 8B Llama models, offering promising feedback generation and judging performance. While this model (like multiple of the others) struggles with hallucination, we believe that fine-tuning techniques might alleviate this issue, and strengthen the results [16].

Practical deployment. Importantly, these open-source models are also easy to access thanks to APIs offered by companies such as HuggingFace. For instance, for conducting our experiments, using the Transformers library, all models were freely accessible, except for Llama-3.1-70B (which required paying 9 dollars for a month of rate-limited access). We acknowledge that using an external API to query open-source language models might defeat the purpose of data privacy. However, several institutions leverage ChatGPT APIs in one way or another [24], and HuggingFace platforms, which are dedicated to open-source, offer the same data privacy guarantees. Moreover, prior works show evidence that smaller models (such as Phi-3-mini) could be deployed on consumer devices [25], or even in browsers [16], alleviating the need to rely on external API.

Limitations. Our work has limitations. The prompts we used likely influenced the results, and more specific prompts or alternative prompting strategies (which we did not explicitly compare) could impact model performance. Also, we only considered introductory programming assignments written in Python and not other programming languages. Moreover, we only considered two types of feedback, but other types such as hints, (with potentially more pedagogical benefits) exist. Our selection of language models, although considered the recent state-of-the-art at the time of writing, does not exhaust all possible alternatives to popular models. Our labelling process is also not perfect, as we only used two raters, which resulted in a moderate inter-rater reliability score (0.54), and our grading criteria did not use actual students (the intended audience) to rate the clarity of the outputs. Additionally, for judging feedback, we could have used judge language models which are specifically designed for evaluation, but we used generic language models instead. Finally, our report lacks a discussion (and examples) of the specific type of issues encountered when generating and judging feedback.

Future Work. In the future, we will conduct a larger-scale evaluation of open-source language models' ability to generate (and judge) other types of feedback and support. In particular, we are extending the Socratic benchmark to include ground truth next-step hints [35], and we are running an evaluation of language models' ability to generate such hints [16], as well as their ability for being Socratic guides. Beyond tracking models' performances, we aim to improve the ability of small language models (e.g. Phi-3-mini) to be teaching assistants by using Reinforcement Learning techniques to tackle the selectivity problem [36]. The varied strengths of different models also suggest that a combined approach (ensemble methods) might yield even better results for feedback generation. As our LLM jury results contrasted those by Verga et al. [37], we also intend to conduct a study on how the variability, individual performance, and the number of judges in the LLM jury affect judging performance.

6 Conclusions

In this paper, we evaluated (1) how language models perform in providing explanations of issues in programs and generating bug fixes (RQ1), and (2) how well different language models, including open-source ones, perform in evaluating the quality of feedback generated by other LMs (RQ2). Our paper highlights that top open-source language models are valid competitors to proprietary language models for both generating and assessing the quality of programming feedback. Open-source language models could provide benefits for powering free tools, which is particularly important for institutions with limited funding. As an additional contribution, we release the code used to conduct our experiments, including the models' outputs and the annotators' responses[1].

Acknowledgments

This research was partially supported by the Research Council of Finland (Academy Research Fellow grant number 356114).

[1] https://github.com/KoutchemeCharles/feed_genju

References

[1] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, et al. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. arXiv:2404.14219

[2] Erfan Al-Hossami, Razvan Bunescu, Justin Smith, and Ryan Teehan. 2024. Can Language Models Employ the Socratic Method? Experiments with Code Debugging. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 53–59. https://doi.org/10.1145/3626252.3630799

[3] Rishabh Balse, Bharath Valaboju, Shreya Singhal, Jayakrishnan Madathil Warriem, and Prajish Prasad. 2023. Investigating the Potential of GPT-3 in Providing Feedback for Programming Assessments. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 292–298.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[5] John Edwards, Arto Hellas, and Juho Leinonen. 2024. On the Opportunities of Large Language Models for Programming Process Data. *arXiv preprint arXiv:2411.00414* (2024).

[6] Google. 2024. Gemma: Our open-source models for machine learning fairness. https://blog.google/technology/developers/gemma-open-models/

[7] Qiang Hao, David H Smith IV, Lu Ding, Amy Ko, Camille Ottaway, Jack Wilson, Kai H Arakawa, et al. 2022. Towards understanding the effective design of automated formative feedback for programming assignments. *Computer Science Education* 32, 1 (2022), 105–127.

[8] Arto Hellas, Juho Leinonen, and Leo Leppänen. 2024. Experiences from Integrating Large Language Model Chatbots into the Classroom. *arXiv preprint arXiv:2406.04817* (2024).

[9] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*. ACM, 93–105.

[10] Yann Hicke, Anmol Agarwal, Qianou Ma, and Paul Denny. 2023. AI-TA: Towards an Intelligent Question-Answer Teaching Assistant using Open-Source LLMs. arXiv:2311.02775 [cs.LG]

[11] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

[12] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proc. of the 2023 CHI Conf. on Human Factors in Computing Systems*. ACM, New York, NY, USA, Article 455, 23 pages.

[13] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2016. Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 41–46.

[14] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Trans. Comput. Educ.* 19, 1, Article 3 (2018), 43 pages.

[15] Natalie Kiesler, Dominic Lohr, and Hieke Keuning. 2023. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. *arXiv preprint arXiv:2309.00029* (2023).

[16] Nachiket Kotalwar, Alkis Gotovos, and Adish Singla. 2024. Hints-In-Browser: Benchmarking Language Models for Programming Feedback Generation. arXiv:2406.05053 [cs.LG] https://arxiv.org/abs/2406.05053

[17] Charles Koutcheme, Nicola Dainese, and Arto Hellas. 2024. Using Program Repair as a Proxy for Language Models' Feedback Ability in Programming Education. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, Ekaterina Kochmar, Marie Bexte, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Anaïs Tack, Victoria Yaneva, and Zheng Yuan (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 165–181.

[18] Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, and Paul Denny. 2024. Open Source Language Models Can Provide Feedback: Evaluating LLMs' Ability to Help Students Using GPT-4-As-A-Judge. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education, Volume 1* (Milan, Italy) *(ITICSE '24)*. https://doi.org/10.1145/3649217.3653612

[19] Charles Koutcheme, Nicola Dainese, Sami Sarsa, Juho Leinonen, Arto Hellas, and Paul Denny. 2024. Benchmarking Educational Program Repair. arXiv:2405.05347 [cs.SE] https://arxiv.org/abs/2405.05347

[20] Sanjay Kukreja, Tarun Kumar, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. 2024. A Literature Survey on Open Source Large Language Models. In *Proceedings of the 2024 7th International Conference on Computers in Management and Business* (Singapore, Singapore) *(ICCMB '24)*. Association for Computing Machinery, New York, NY, USA, 133–143. https://doi.org/10.1145/3647782.3647803

[21] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, et al. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 124–130. https://doi.org/10.1145/3587102.3588785

[22] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, et al. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, New York, NY, USA, 563–569.

[23] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proc. of the 23rd Koli Calling Int. Conf. on Computing Education Research*. ACM, New York, NY, USA, Article 8, 11 pages.

[24] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 1927. https://doi.org/10.1145/3626253.3635427

[25] Suqing Liu, Zezhu Yu, Feiran Huang, Yousef Bulbulia, Andreas Bergen, and Michael Liut. 2024. Can Small Language Models With Retrieval-Augmented Generation Replace Large Language Models When Learning Computer Science?. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 388–393. https://doi.org/10.1145/3649217.3653554

[26] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 2511–2522. https://doi.org/10.18653/v1/2023.emnlp-main.153

[27] Marcus Messer, Neil CC Brown, Michael Kölling, and Miaojing Shi. 2024. Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Computing Education* 24, 1 (2024), 1–43.

[28] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Trans. Comput. Educ.* 22, 3, Article 34 (2022), 40 pages.

[29] Maciej Pankiewicz and Ryan S. Baker. 2023. Large Language Models (GPT) for automating feedback on programming assignments. arXiv:2307.00150 [cs.HC]

[30] Tung Phung, José Cambronero, Sumit Gulwani, Tobias Kohn, Rupak Majumdar, et al. 2023. Generating High-Precision Feedback for Programming Syntax Errors using language Models. arXiv:2302.04662 [cs.PL]

[31] Tung Phung, Victor-Alexandru Pădurean, José Cambronero, Sumit Gulwani, Tobias Kohn, et al. 2023. Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors. *Int. J. of Management* 21, 2 (2023), 100790.

[32] Tung Phung, Victor-Alexandru Pădurean, Anjali Singh, Christopher Brooks, José Cambronero, et al. 2023. Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation. arXiv:2310.03780 [cs.AI]

[33] James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Virginia Pettit, Leo Porter, et al. 2024. How Instructors Incorporate Generative AI into Teaching Computing. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2*. 771–772.

[34] Nazneen Rajani, Nathan Lambert, Sheon Han, Jean Wang, Osvald Nitski, et al. 2023. Can foundation models label data like humans? https://huggingface.co/blog/llm-v-human-data.

[35] Lianne Roest, Hieke Keuning, and Johan Jeuring. 2024. Next-Step Hint Generation for Introductory Programming Using Large Language Models. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) *(ACE '24)*. Association for Computing Machinery, New York, NY, USA, 144–153. https://doi.org/10.1145/3636243.3636259

[36] Alexander Scarlatos, Digory Smith, Simon Woodhead, and Andrew Lan. 2024. Improving the Validity of Automatically Generated Feedback via Reinforcement Learning. In *Artificial Intelligence in Education*, Andrew M. Olney, Irene-Angelica Chounta, Zitao Liu, Olga C. Santos, and Ig Ibert Bittencourt (Eds.). Springer Nature Switzerland, Cham, 280–294.

[37] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models. arXiv:2404.18796 [cs.CL] https://arxiv.org/abs/2404.18796

[38] Sierra Wang, John Mitchell, and Chris Piech. 2024. A large scale RCT on effective error messages in CS1. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 1395–1401.

[39] Lixiang Yan, Lele Sha, Linxuan Zhao, Yuheng Li, Roberto Martinez-Maldonado, et al. 2023. Practical and Ethical Challenges of Large Language Models in Education: A Systematic Scoping Review. *British Journal of Educational Technology* (2023).

[40] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, et al. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL]