Large Language Models for Student Code Evaluation: Insights and Accuracy

Alfonso Piscitelli[a], Mattia De Rosa[b], Vittorio Fuccella[c] and Gennaro Costagliola[d]
Department of Informatics, University of Salerno, Fisciano (SA), Italy
{*apiscitelli, matderosa, vfuccella, gencos*}*@unisa.it*

Keywords: Programming Education, Large Language Models, Automatic Code Evaluation.

Abstract: The improved capabilities of Large Language Models (LLMs) enable their use in various fields, including education. Teachers and students already use LLMs to support teaching and learning.
In this study, we measure the accuracy of LLMs *gpt-3.5, gpt-4o, claude-sonnet-20241022, and llama3* in correcting and evaluating students' programming assignments. Seven assessments carried out by 50 students were assessed using three different prompting strategies for each of the LLMs presented. Then we compared the generated grades with the grades assigned by the teacher, who corrected them manually throughout the year.
The results showed that models such as *llama3* and *gpt-4o* obtained low percentages of generated evaluations, while *gpt-3.5* and *claude-sonnet-20241022* obtained interesting results if they received at least one example of evaluation.

1 INTRODUCTION

In recent years, Large Language Models (LLMs) have demonstrated significant versatility in addressing a broad range of tasks, extending their utility well beyond generating natural language text. Indeed, LLMs have been employed to address a variety of tasks, including classification (Alahmadi et al., 2024), program analysis, and code generation across multiple programming languages (Jiang et al., 2024; Dong et al., 2024). As their capabilities increase, and as these tools become easier to use, they can also be used by educational staff to support teaching or analysing assignments.

In this study, we analysed the capabilities of LLMs in evaluating programming assignments (written in the C language) carried out by 50 students, to understand the extent to which these tools can be used to support teachers in correcting assignments, or at least as support tools in student self-assessment. We automatically invoked the various LLMs used in this study: *gpt-3.5 (OpenAI, 2022), gpt-4o (OpenAI, 2024), claude-sonnet-20241022 (Anthropic, 2024), llama3 (Meta, 2024)*. The prompts required for the execution of the experiment are defined in a JSON file, structured in such a way as to be able to test these three different strategies: *zero-shot* (i.e. without providing any examples to the model), *one-shot* (i.e. with an example provided to the LLM), *few-shot* (i.e. providing a few examples, five, to the model). One example consists of the pair $(assignment, vote)$.

During the experiment, a dataset composed of 350 assignments carried out during a Programming and Data Structures course was used. The assignments covered a range of programming concepts, including basic data types, control structures, and simple algorithms.

The results showed that the LLMs did not perform well in the assessment tasks, with a significant gap compared to the human evaluation provided during the lecture. The difference between the evaluation generated by the LLMs and that of the correcting teacher is approximately 1.5 points out of 10; however, these models generally perform better with the one-shot prompting strategy, in particular *claude-sonnet-20241022*, which achieves an average difference of 1.17 (S.D. 1.25), and 60% of the marks with a difference of 1 or less.

The remainder of the paper is structured as follows. Section 2 presents the related work, while Section 3 describes the methodology, the research ques-

[a] https://orcid.org/0000-0002-3567-1504
[b] https://orcid.org/0000-0002-6922-5529
[c] https://orcid.org/0000-0003-3244-3650
[d] https://orcid.org/0000-0003-3816-7765

Piscitelli, A., De Rosa, M., Fuccella, V. and Costagliola, G.
Large Language Models for Student Code Evaluation: Insights and Accuracy.
DOI: 10.5220/0013287500003932
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU 2025) - Volume 2*, pages 534-544
ISBN: 978-989-758-746-7; ISSN: 2184-5026
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

tions and the software developed to carry out this experiment. Section 4 presents the results and answers to the research questions, while Section 5 presents the implications of the results and describes some ethical concerns. Finally, Section 6 presents the conclusions and next steps.

2 BACKGROUND

Large Language Models are models trained on a very large amount of data and can understand and generate text in different contexts. Communication with the LLM takes place via *instructions* (which may be simple or complex) that are commonly referred to as *prompts*. Querying LLMs with appropriate prompts is crucial to the quality of the expected response, as is the number of parameters in the model itself (Brown et al., 2020). An appropriate prompt generally consists of some contextual information and a set of expected questions and answers (examples) that form the knowledge base of the LLM. The following definitions formalize key concepts related to the interaction with large language models, including examples and prompts, which are essential for effective communication and response generation.
Definition 1: Example. Let Q be the question to be asked of the LLM and A the answer. Let us define the pair (Q,A) as example E. In the case where the answer is obtained from the language model, this will be referred to as $A_{obtained}$; conversely, in the case where this answer is generated as training data, it will be referred to as $A_{expected}$.
Definition 2: Prompt. A prompt P is defined as a pair $(C,E*)$, where C represents the *context* instructions (also called *system*) while $E*$ represents a list of examples (0 or more examples).

The size of the example set, denoted as $|E*|$, significantly influences LLM performance (Brown et al., 2020). Using the number of examples, we can define three distinct prompting strategies:

- **zero-shot**, which involves sending prompts to the LLM without providing any examples;
- **one-shot**, which involves sending prompts to the LLM with one example;
- **few-shot**, which involves sending prompts to the LLM with a few examples (from less than 10 to even a hundred)

There are several strategies for designing the example set. One approach is to define a list of expected results. Alternatively, counterexamples can be included, as demonstrated in the chain-of-thought method (Wei et al., 2024), which helps LLMs better understand the task at hand. Another effective strategy involves providing verification messages for the generated response, particularly useful when the response must adhere to a specific syntax that can be validated by a parser or validator (Tabari et al., 2025).

In addition to presenting examples as a list of (Q,A) pairs, (Wang et al., 2023) explored the use of meta-languages, such as context-free grammars (CFGs), to define examples. This approach leverages the recursive nature of CFGs, where a finite set of rules can generate an infinite number of valid examples by repeatedly applying these rules. For instance, a simple grammar rule like:

$$S \rightarrow aSb \mid \varepsilon$$

can produce strings such as 'ab,' 'aabb,' 'aaabbb,' and so on, without any predefined upper limit. This allows for the representation of diverse and flexible example structures in a compact manner, enabling the systematic generation of complex examples that would otherwise be impractical to enumerate exhaustively. This type of prompting, called *grammar-prompting*, reached interesting results in working with domain-specific languages. LLMs have also proven to be particularly useful in the understanding and translation of programming languages (Eniser et al., 2024).

LLMs have also found a place in education and teaching. In (Piscitelli et al., 2024), an empirical study analysed students' interaction with a *gpt-4o-based* bot when solving programming assignments, finding that students generally tend to rely on the LLM in requesting solutions (which is something to be discouraged) but also that several students tend to use it as a kind of online help (which is something to be encouraged). Similar conclusions were reached in (Liffiton et al., 2024), particularly on the over-reliance of students (and junior programmers) on LLM. Over-reliance on LLMs is also an issue analysed in (Prather et al., 2023), highlighting how in certain cases the code generated by LLMs makes it difficult to analyse and fix errors at later stages. In any case, these types of tools, with appropriate modifications, can be useful in student learning, as highlighted in (Qi et al., 2023), where they limited LLMs to providing only code examples and never the solution to the problem, through prompting (Dong et al., 2024).

The use of LLMs for student assessment represents a further use of these models, which on the one hand can support the teacher in his work and on the other hand can provide the student with a self-assessment tool available during his studies.

In (Henkel et al., 2024), the use of *gpt-4o* to evaluate students' answers to quizzes was analysed, and the results showed comparable performance to human evaluations. In the study of (Chiang et al., 2024), on the other hand, *gpt-4o* was again used to automatically assess the assignments of around 1,000 students, and the results were mixed. While the evaluations obtained were acceptable, the model did not always comply with the instructions for evaluating the assignments. Not only that, in some cases, students were able to manipulate the LLM-based assistant to still obtain a high score, but without doing the assignment.

Studies specifically looking at the use of LLMs (of various LLMs in particular) for the assessment of programming tasks have not been carried out. However, some studies have investigated the use of LLMs to carry out program analysis. In (Mahbub et al., 2024) gpt-4o was used to detect inconsistencies of comments in the code, while in (Chapman et al., 2024) prompting techniques were used to perform static program analysis and detect anomalies.

3 METHODOLOGY

In this section, we will describe the research methodology used for this study, the software architecture, the data used for the experiment, and the metrics that were collected during the execution of the software.

3.1 Research Questions

This project aims to analyse the ability of the most common LLMs to analyse student assignments and assess them, generating a grade and a comment, based on an evaluation grid provided to the LLM. This study, therefore, will attempt to answer the following research questions:

- **RQ1** - What is the accuracy of LLMs in generating grades and comments for programming assignments?
- **RQ2** - Which LLM achieves the best performance in terms of evaluation time for programming assignments?
- **RQ3** - What is the difference between LLM-generated grades and those assigned by educators for programming assignments?

3.2 The ClueLLM4Eval Platform

To achieve the objectives presented in the RQ section, we implemented a software, ClueLLM4Eval, using the Java language, that would allow the different LLMs to be queried with the different prompt strategies available in the system. The various classes for querying the following LLMs were implemented by adopting a Strategy Pattern. In addition to the strategy pattern, a workflow was also implemented that reads from a JSON file the sequence of instructions to be sent to the LLM. The JSON file takes up the structure of the 'conversation' with the model, having the following structure:

```
[{
  "role": "system",
  "content": "You are a teacher who has
             to evaluate with a mark
             from 0 to 10 the
             following assignment."
}, {
  "role": "user",
  "content": "C code of the assignment"
},{
  "role": "assistant",
  "content": "10; The work done is correct
               and all points have been
               completed."
}]
```

This structure, appropriately configured according to the prompting strategy adopted, is then loaded at runtime and sent to the LLM together with the actual task to be done by the system.

3.3 Prompting Strategies

During this experiment, three different prompting strategies were used, described below:

- **zero-shot:** No example is given to the LLM, only the system prompt.
- **one-shot:** an example of a well-done assignment for the LLM (graded 10), and the system prompt.
- **few-shot:** two examples: one assignment well done (graded 10) and another assignment incomplete and rated 0 at the LLM, and the system prompt.

The initial system prompt included both the description of the task (in our case, evaluating student assignments) and the evaluation grid that the system had to consider when grading the various assignments. Table 1 reports the text of the system prompt used for the experiment.

The grid was designed to provide the evaluation criteria for assigning a score to each assignment, with scores ranging from a maximum of 10 to a minimum of 0. Table 3 describes the grid used for the evaluation: we defined with grade *ten* an assignment that was correctly done and well documented through inline and block comments, while the grade *zero* defined an assignment that was not done, or done incorrectly or incompletely.

Table 1: System prompt used in this study.

You are a teacher who has to evaluate with a mark from 0 to 10 the following assignment. Describe the evaluation of the assignments using this format: **GRADE;COMMENT**. 0 totally wrong assignment. 10 totally correct assignment. Grade must respect the following grid, and ALL criteria for a grade must be met to assign that grade: To evaluate, use the following grid: $\{\{EVALUATION\_GRID\}\}$ Before assigning a grade, evaluate each required function separately, noting its presence and correctness. Then, based on these individual evaluations, assign an overall grade that matches ALL criteria for that grade level. Provide a brief justification (less than 100 words) for the assigned grade, referencing the specific criteria met or missed. This is the assignment: $\{\{ASSIGNMENT\}\}$

Table 2: Aggregate results of LLMs in evaluation tasks, combining performance across all three prompting strategies.

LLM	Valid Response	Format Non-Compliant Grade	Invalid Response
gpt-3.5	62.76%	28.57%	8.67%
gpt-4o	5.43%	29.71%	64,86%
claude-sonnet-20241022	47.36%	35.08%	17.56%
llama3	0%	0%	100%

Table 3: Evaluation grid used for the correction of the assignments and described in the prompt.

Grade	Meaning
0	The assignment was not carried out
1	The assignment is incorrect and incomplete
2	The assignment is incorrect and incomplete, with some sketched parts
3	The assignment is incorrect and incomplete, with several sketched parts
4	The assignment is mainly incorrect and incomplete, with some parts almost completed
5	The assignment is partially incorrect and incomplete, with many parts almost completed
6	The assignment is partially correct and well designed, with several syntactic and semantic errors
7	The assignment is correct and well designed, with few semantic errors
8	The assignment is correct and well designed, but with not a good structure and poorly commented
9	The assignment is correct and well designed, but with several parts not commented
10	The assignment is entirely correct, well designed, and well commented

3.4 Datasets

For this experiment, we created a dataset consisting of 7 assignments completed by 50 students each, resulting in a total of 350 assignments. This dataset consists of assignments carried out by real students attending the first year of the Bachelor of Science in Computer Science in the Programming and Data Structures course, who voluntarily granted the use of the assignments for this experiment. The assignments covered the implementation of abstract data types, array sorting algorithms, and data structures such as lists, queues, stacks, and trees. The evaluation and errors of these assignments are also known, as they were manually checked and assessed by the course assistant lecturer. The manual evaluations are used during the results analysis phase as a comparison to the evaluations provided by the LLM.

3.5 Response Classification

To assess the outputs produced by LLMs, a well-defined taxonomic framework is essential to differentiate between responses based on their conformity to the required format and level of completeness. This classification system partitions responses into valid, partially valid, and unusable categories, facilitating a structured evaluation of the models' performance and their suitability for educational applications.

1. **Valid Responses**
   - **Definition:** A *Valid Response* is an LLM response which adhere to the required format, a numerical grade followed by a text comment.
   - **Example.** 8;well-structured code with minor issues in commenting

2. **Format Non-Compliant Responses**
   - **Definition.** A *Format Non-Compliant Response* is an LLM response that contain both a grade and a comment but fail to comply with the exact required format. These responses are still interpretable, but require further parsing or manual correction to be converted into a compliant format.
   - **Example.** Grade: 8. Comments: Well-structured code with minor issues.
3. **Incomplete or Empty Responses**
   - **Definition.** An *Incomplete Response* is an LLM response that do not contain both required elements (grade and comment) or are entirely off-task.
   - **Examples.** *The code looks good but could be improved.* (missing grade) or *I cannot process this assignment.* (off-task response).

3.6 LLMs Used

In this experiment, the following LLMs were used:

- *gpt-4o* (OpenAI, 2024)
- *gpt-3.5* (OpenAI, 2022)
- *llama3* (Meta, 2024)
- *claude-sonnet-20241022* (Anthropic, 2024)

For all these LLMs we used the same prompt and we set the *temperature* : 0.5. Temperature is a parameter used to control the randomness and creativity of responses generated by a language model, generally between 0 and 2. It determines how the model assigns probabilities to different possible outputs. A higher temperature (e.g., 1.0 or above) results in more diverse and unpredictable outputs, as the model gives less priority to the most probable next word and explores a wider range of possibilities. Conversely, a lower temperature (e.g., close to 0) makes the model more focused and deterministic, favoring highly probable outputs and reducing variability.

3.7 Metrics

Each session of this experiment involved three prompting strategies, four different LLMs, and the dataset of assignments provided by students. The metrics that will be measured are as follows:

- **Correction Time:** the time required by the LLM to evaluate the assessment given in input;
- **Assessment Difference:** difference between the grade generated by LLMs and the grade given by the course assistant lecturer.
- **Valid Responses:** number of *valid responses* generated by LLMs;
- **Format Non-Compliant Grades:** number of *Format Non-Compliant responses*;
- **Invalid Responses:** number of *incomplete or empty responses*;

3.8 Procedure

Before conducting the experiment, the dataset was prepared by organizing the assignments. Subsequently, we generated the JSON file containing the prompts, which incorporated the evaluation grid and the assignment track, as presented in Table 1.

Throughout the experiment, the platform executed the prompt against all the LLMs described in Section 3.6. To respect the token limitations of each platform, the platform implemented a ten-second timeout between each execution.

At the end of the experiment, the collected data were analysed and categorised into three groups: valid responses, format non-compliant grades, and invalid responses. After enumerating the results and incorporating them into Table 2, a data cleansing process was performed to refine the incorrectly formatted grades, thereby generating valid responses from them.

4 RESULTS

This section describes the results obtained for the three research questions guiding this study. The entire experiment took approximately one day to run, the majority of which was required by the *llama3* model, as it was run locally on the researchers' machines. The other models analysed in this study took approximately one hour each to run for the analysis of the dataset described in the previous section.

Based on the results obtained, it was determined that the *llama3* model did not generate any accurate responses, as indicated in Table 2. As a result, the discussion and comparison of the research questions in the subsequent sections will exclude any references to this model.

4.1 RQ1: Accuracy of LLMs

In this section of the study, we will attempt to answer the question RQ1: *What is the accuracy of LLMs in generating grades and comments for programming assignments?*. Table 2 shows the results obtained from individual LLMs, aggregated for all

Table 4: Overall results of prompting strategies in evaluation tasks.

LLM	Strategy	Valid Response	Format Non-Compliant Grade	Invalid Responses
gpt-3.5				
	zero-shot	0.57%	80.57%	18.86%
	one-shot	93.14%	3.43%	3.43%
	few-shot	94.57%	1.71%	3.72%
gpt-4o				
	zero-shot	1.14%	41.43%	57.43%
	one-shot	2.57%	20.29%	77.14%
	few-shot	12.57%	27.43%	60.00%
claude-sonnet-20241022				
	zero-shot	0.00%	61.53%	38.47%
	one-shot	78.95%	10.53%	10.52%
	few-shot	64.86%	32.43%	2.71%

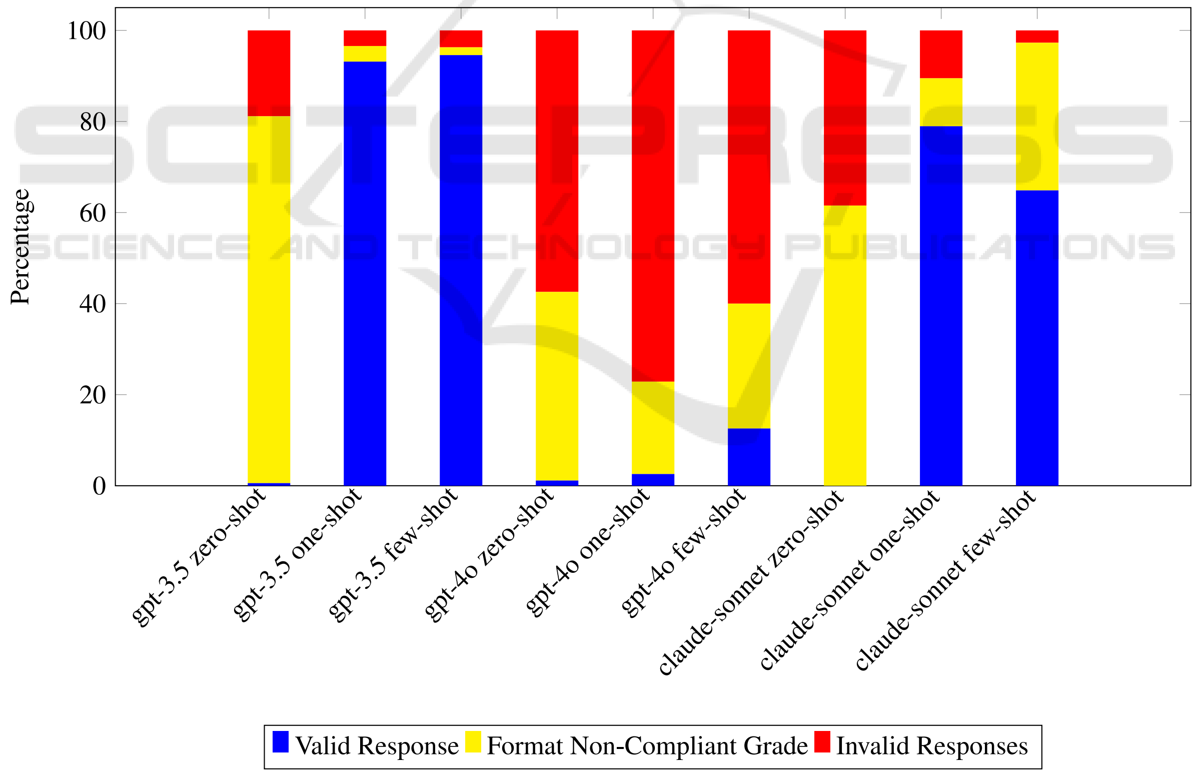

Figure 1: Overall results of prompting strategies in evaluation tasks. The blue bars depict the valid responses generated, the yellow bars represent the grades produced in an incorrect format, and the red bars illustrate the invalid responses generated.

prompting strategies (which will be discussed in Section 4.3). The prompt required to assess the assignment and provide the result of the grade in the format *GRADE*;*COMMENT* where the first value was the grade (numerical) and the second was the comment.

The *valid response* column shows the percentage of answers that complied with the required format, while the *Format Non-Compliant Grade* column shows the percentage of answers that contained both the grade and the commentary but did not strictly comply with the required format (many answers were of the type GRADE:8, or ##Grade:8).

Table 2 shows that *gpt-4o* was able to provide a valid response (that corresponded to the format described in the prompt's request) in 5 percent of the cases. When including responses that contained both a grade and a comment but were non-compliant with the required format, the total percentage of responses increased to 35 percent. The *llama3* model did not provide correct answers: in none of the analysed prompt strategies did the model provide a numerical evaluation, always limiting itself to a descriptive evaluation of the assigned exercise.

The other two models analysed, however, *gpt-3.5* and *claude-sonnet-20241022*, gave better results. *gpt-3.5* was able to provide answers that respected the required format in 62.76% of the evaluations, while 28.57% of its evaluations included errors in formatting but still contained both a score and a comment. *Claude-sonnet-20241022*, on the other hand, provided formally correct answers in 47.36% of cases, while about 35% of the time it produced answers with a grade preceded by additional textual commentary, which were non-compliant with the expected format. This analysis pertains to formal correctness, while the substantive correctness of the scores and comments is examined separately in Section X (or elsewhere, if applicable).

Table 4 instead shows the results obtained from the various models, highlighting the prompting strategy used. The first result that emerges from these results is that the prompting strategy *zero-shot*, without examples, rarely succeeds in providing a correctly formatted response, while still performing the assigned task and evaluating the work performed (80% for *gpt-3.5* and 61% for *claude-sonnet-20241022*). Already providing a single evaluation example increases the model's ability to provide an answer in the required format and to reduce the number of null values, i.e. failed evaluations. The graph in Figure 1 makes it clear that as the number of examples increases, the number of responses that comply with the format increases, and errors generally decrease.

4.2 RQ2: Time Analysis

This section examines the second research question, *Which LLM achieves the best performance in terms of evaluation time for programming assignments?*. Table 5 exhibit varying performance across the different prompting strategies. Specifically, the assignment evaluation on *gpt-3.5* took approximately 30 minutes, *gpt-4* required over an hour, and *claude-sonnet-20241022* required around two hours. Furthermore, the average time required to evaluate each assignment was 1.81 seconds for *gpt-3.5*, 3.73 seconds for *gpt-4*, and 6.66 seconds for *claude-sonnet-20241022*. These results suggest that the *gpt-3.5* model provided the fastest execution times among the three LLMs tested in this study.

The time required for evaluating the assignment does not appear to be significantly impacted by the prompting strategies employed, except for the *gpt-3.5* model. In the case of the *gpt-3.5* model, the execution time was reduced from 1.99 seconds for zero-shot prompting to 1.71 seconds when utilizing one and few-shot prompting strategies. This suggests that the prompting strategies can affect the performance of the *gpt-3.5* model, leading to a noticeable decrease in the execution time for the assignment evaluation.

The analysis of the prompting strategies indicates that sending samples to the large language models may also be beneficial for execution time, but a more in-depth analysis is required. Specifically, for the models *gpt-3.5* and *claude-sonnet-20241022*, the one-shot strategy demonstrated the fastest execution, while no significant differences were observed for the *gpt-4o* model. The low variability exhibited in the Standard Deviation column suggests a high level of consistency across the executions. Further investigation is needed to fully understand the impact of the prompting strategies on the execution time of these language models, as the current findings suggest potential performance advantages, but more comprehensive testing is required to draw definitive conclusions (see Section 5).

4.3 RQ3: Comparison with Human Assessment

This section investigates the third research question, *What is the difference between LLM-generated grades and those assigned by educators for programming assignments?*. As reported in Table 6, the rating discrepancies across the various large language models and prompting strategies vary, with some models and strategies showing substantial differences, while others are more closely aligned with human assessments. The smallest average difference of 1.17 (S.D. 1.25) points was achieved by the *claude-sonnet-20241022* model using the one-shot prompting approach, followed by the zero-shot strategy with a 1.38 (S.D. 1.28) point difference out of 10, and the one-shot strategy of *gpt-4o* with a 1.58 (S.D. 1.21) point difference. However, other strategies and models exhibit larger disparities, in some cases approaching 2 points,

Table 5: Time analysis on evaluation tasks performed by LLMs across the different prompting strategies.

LLM	Strategy	Total Time (s)	Mean Time (s)	S.D. (s)
gpt-3.5		1901,40	1,81	1,81
	zero-shot	695,86	1,99	6,89
	one-shot	606,89	1,73	4,91
	few-shot	598,66	1,71	4,01
gpt-4o		3916,23	3,73	1,63
	zero-shot	1302,45	3,72	4,62
	one-shot	1303,60	3,72	3,75
	few-shot	1310,18	3,74	6,04
claude-sonnet-20241022		6989,49	6,66	2,57
	zero-shot	2305,21	6,59	1,26
	one-shot	1777,85	5,08	3,02
	few-shot	2922,67	8,35	2,04

which can be considered a significant deviation for an evaluation.

The *claude-sonnet-20241022* model demonstrated the best performance in terms of the number of evaluations that matched those of human raters, and overall, this model had the highest rate of evaluations that were quite close to the human grades. The one-shot strategy attained 31.58% of evaluations that were identical to the human ones for program assignments, and 28.95% of evaluations that differed from the human evaluation by between 0.5 and 1 point, for a total of 60.53% of grades that differed from the human evaluation by less than 1 point. In general, the *claude-sonnet-20241022* model achieved good results across each strategy, followed by the *gpt-3.5* model, which obtained the best results using the zero-shot strategy, with 38.86% of grades differing from the human evaluation by less than 1 point.

The LLMs generally exhibit moderate discrepancies in their evaluations compared to human assessments, although certain models and prompting strategies demonstrate larger deviations. However, *claude-sonnet-20241022* demonstrates a stronger capacity to emulate human-like evaluations, particularly in the one-shot strategy. Conversely, *gpt-3.5* and *gpt-4o* exhibit more substantial deviations, implying that their calibration to mimic human evaluations could be enhanced, especially in the zero-shot and few-shot approaches.

5 DISCUSSIONS

The results presented so far show that using Large Language Models to generate automatic corrections of programming assignments does not offer consistent results compared to the assessment given by the teacher, deviating rather markedly from the intended assessment. This evidence encourages the pursuit of more prompting strategies to enhance the quality of assessments and align them as closely as possible with the desired outcomes.

Although the activity of correcting 350 assignments took quite some time, reaching around two hours *claude-sonnet-20241022* to correct the assignment, it must be said that this time is far less than the time a teacher would spend correcting 350 assignments: time reduction has the advantage of providing students with feedback quickly, with the undeniable advantages this has on learning. However, assessments must also be consistent and as close to the grading grid as possible; otherwise, there is a risk of creating barriers to student learning. The reliability highlighted shows that LLMs alone are not usable in a self-assessment context, as the assessments may deviate widely from the grid with which they would be assessed by the teacher himself.

The models analysed in many cases did not comply with the required format: the 2 table highlighted in the *Format Non-Compliant Grade* column the grades obtained in a different format: this phenomenon occurs in the majority with the *zero-shot* strategy, which suggests that providing examples allows LLMs to better consider the required response

Table 6: Analysis of the rating differences between the ratings generated by the language models and those assigned by humans, based on different prompting strategies. This table shows the mean difference (Average Diff) and standard deviation (S.D.) between the ratings, the percentage of identical ratings (Equal Grade) and the percentage of differences of less than 1 point ($0 < \mathit{Difference} \leq 1$) calculated over the total number of assignments, including those for which the LLM was unable to give an evaluation.

LLM	**Strategy**	**Average Diff**	**S.D.**	**Equal Grade**	**0 < Difference ≤ 1**
gpt-3.5		1.86	1,40	8,86%	24,19%
	zero-shot	1.72	1,41	10,29%	28,57%
	one-shot	1.86	1,22	7,71%	21,14%
	few-shot	1.98	1,52	8,57%	22,86%
gpt-4o		1.82	1,48	2,57%	11,90%
	zero-shot	1.72	1,31	2,86%	14,57%
	one-shot	1.58	1,21	2,29%	8,00%
	few-shot	2.06	1,74	2,57%	13,14%
claude-sonnet-20241022		1,41	1,28	18,42%	28,07%
	zero-shot	1,38	1,28	12,82%	28,21%
	one-shot	1,17	1,25	31,58%	28,95%
	few-shot	1,67	1,28	10,81%	27,03%

format. In this direction, the addition of further examples including invalid formats, or even an agent verifying the syntax of the response, possibly alerting the model to errors to be corrected, can be analysed.

Moreover, the analysis of the comments generated in natural language by the various LLMs indicated that high variability was associated with difficulties in comprehending the code under evaluation. When the assessment diverged by more than 1.5 points from the human judgment, the LLMs tended to penalize the structure of the comments or excessively emphasize secondary elements. Future research will involve a more detailed examination of the models' responses during the evaluation process.

In general, the *claude-sonnet-20241022* model was seen to obtain the best results, despite being slower than the *gpt-3.5 gpt-4o and llama3* models. *llama3* in particular did not obtain analysable results for this study: the same prompts used for the other models, with *llama3* did not generate acceptable answers in any case, but only comments on the code. Again, as mentioned above, the system could benefit from the use of a validator that requests consistent responses from the LLM concerning the format and task required.

The findings of this study suggest that prompting alone is inadequate for managing the complexity of assignment evaluation and generating consistent results. To address this limitation, future work should explore new prompting strategies and develop a model specifically tailored for this purpose. Additionally, future research should consider incorporating other large language models beyond those included in this study to further enhance the evaluation and generation of consistent results.

Furthermore, the evaluation of an assignment is a multifaceted task that involves more than simply assigning a grade. In this study, we asked the large language models to provide the rationale behind the assigned grades. However, a preliminary analysis suggests that the LLMs' responses either merely reiterate the comments from the evaluation rubric or offer an extended analysis of the provided code, which may not be particularly useful in an educational context. Instead, students require constructive feedback that enables them to comprehend their errors and enhance their solutions, a consideration that will be further explored in future work. (Hundhausen et al., 2013; Huang et al., 2018)

5.1 Ethical Concerns

In this study, assignments carried out by students in an academic year 2023 class were used: none of these generated assessments were provided to students. However, the dissemination of increasingly accurate LLM models also requires ethical evaluations of how such a tool should be used by a teacher or a student, in the case of self-assessment. The teacher establishes a relationship with the learner and in the correction of assignments may detect progressions or regressions that a model might not highlight, or worse, provide, unless he or she is adequately trained for this. LLM alone could compromise this relationship, impacting the effectiveness of didactic personalisation.

However, these tools have undeniable potential as valuable allies in personalising teaching and supporting student self-assessment. Leveraging techniques such as Retrieval-Augmented Generation (RAG), which combines the generative power of language models with the precision of retrieving relevant and up-to-date information from external sources (such as teacher-provided learning materials), they can provide students with quick, accurate and personalised feedback on their level of preparation, improving learning outcomes and adaptability. (Lewis et al., 2020).

6 CONCLUSIONS

In this study, we presented an experiment on the use of Large Language Models for correcting and evaluating student assignments from the Programming and Data Structures course. In the course of the experiment, 7 assignments carried out by 50 students, totaling 350 submissions, were evaluated using four different LLMs: *gpt-4o, gpt-3.5, llama3, claude-sonnet-20241022*. The experiment aimed to measure the ability of the LLMs to generate answers in the required format (*GRADE*;*COMMENT*) and to measure the distance between the evaluation generated by the LLMs and the evaluation already in the possession of the researchers provided by the teacher who carried out the manual corrections. The results showed that:

- **RQ1: Evaluating LLM Accuracy on Task-Based Assessments.** For this type of task, the LLMs showed difficulties in complying with the format, particularly in the zero-shot strategy. *llama3* did not provide any evaluation but only comments. In contrast, the other LLMs (in particular *gpt-3.5* and *claude-sonnet-20241022* provided valid evaluations in 94.57% and 78.95%, for the few-shot and one-shot strategies, respectively).
- **RQ2: Timing Results of LLMs: Performance Insights.** For this type of task, the *gpt-3.5* model was generally faster (taking on average 1.81s per evaluation, S.D. 1.81s), followed by *gpt-4o* (mean 3.73s, S.D. 1.63s) and *claude-sonnet-20241022* (mean 6.66s., S.D. 2.57). No significant difference in timing was observed between the various prompting strategies.
- **RQ3: The Gap Between LLMs and Human Evaluation.** For this type of assignment, the average distance between the LLMs' assessment and that previously provided by the teacher is around 1.7 points out of 10, with *claude-sonnet-20241022* having the lowest distance (1.41, S.D. 1.28) and the greatest number of assessments no more than one point out of 10 apart with the *one-shot* strategy (60.53%). Analysing the comments on the evaluations, it emerged that a high-grade difference is often related to the evaluators having a different interpretation of the assessment grid or to them perceiving errors that are not present in the work being evaluated. This variability suggests, however, that these instruments need further improvement before they can be used for these tasks.

Future work will attempt to analyse the limitations already presented in the previous section. We will first try to analyse the reasons why the proposed prompting strategies did not work with *llama3*. We will also analyse the impact of other prompting strategies, increasing the number of evaluation examples. Moreover, Retrieval-Augmented Generation (RAG) has been shown to be suitable in several fields and also in education (Liu et al., 2024). We plan to analyse the impact of RAG in the evaluation of assignments in order to automatically obtain the best evaluation grid for different types of assignments. In addition, other scenarios will be analysed. Finally, we will integrate additional LLMs, and assess the possibility of training an ad hoc model for this type of task.

ACKNOWLEDGEMENTS

This work was partially supported by grants from the University of Salerno (grant number: 300392FRB22COSTA).

REFERENCES

Alahmadi, M. D., Alshangiti, M., and Alsubhi, J. (2024). Scc-gpt: Source code classification based on generative pre-trained transformers. *Mathematics*, 12(13).

Anthropic (2024). https://www.anthropic.com/news/claude-3-5-sonnet - last access 28th october 2024.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Chapman, P. J., Rubio-González, C., and Thakur, A. V. (2024). Interleaving static analysis and llm prompting. In *Proceedings of the 13th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis*, SOAP 2024, page 9–17, New York, NY, USA. Association for Computing Machinery.

Chiang, C.-H., Chen, W.-C., Kuan, C.-Y., Yang, C., and yi Lee, H. (2024). Large language model as an assignment evaluator: Insights, feedback, and challenges in a 1000+ student course.

Dong, G., Yuan, H., Lu, K., Li, C., Xue, M., Liu, D., Wang, W., Yuan, Z., Zhou, C., and Zhou, J. (2024). How abilities in large language models are affected by supervised fine-tuning data composition. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, Bangkok, Thailand. Association for Computational Linguistics.

Eniser, H. F., Zhang, H., David, C., Wang, M., Christakis, M., Paulsen, B., Dodds, J., and Kroening, D. (2024). Towards translating real-world code with llms: A study of translating to rust.

Henkel, O., Hills, L., Boxer, A., Roberts, B., and Levonian, Z. (2024). Can large language models make the grade? an empirical study evaluating llms ability to mark short answer questions in k-12 education. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, L@S '24, page 300–304, New York, NY, USA. Association for Computing Machinery.

Huang, L., Zhao, H., Yang, K., Liu, Y., and Xiao, Z. (2018). Learning outcomes-oriented feedback-response pedagogy in computer system course. In *2018 13th International Conference on Computer Science & Education (ICCSE)*, pages 1–4.

Hundhausen, C. D., Agrawal, A., and Agarwal, P. (2013). Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Trans. Comput. Educ.*, 13(3).

Jiang, X., Dong, Y., Wang, L., Fang, Z., Shang, Q., Li, G., Jin, Z., and Jiao, W. (2024). Self-planning code generation with large language models. *ACM Trans. Softw. Eng. Methodol.*, 33(7).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Liffiton, M., Sheese, B. E., Savelka, J., and Denny, P. (2024). Codehelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, Koli Calling '23, New York, NY, USA. Association for Computing Machinery.

Liu, C., Hoang, L., Stolman, A., and Wu, B. (2024). Hita: A rag-based educational platform that centers educators in the instructional loop. In Olney, A. M., Chounta, I.-A., Liu, Z., Santos, O. C., and Bittencourt, I. I., editors, *Artificial Intelligence in Education*, pages 405–412, Cham. Springer Nature Switzerland.

Mahbub, T., Dghaym, D., Shankarnarayanan, A., Syed, T., Shapsough, S., and Zualkernan, I. (2024). Can gpt-4 aid in detecting ambiguities, inconsistencies, and incompleteness in requirements analysis? a comprehensive case study. *IEEE Access*, pages 1–1.

Meta (2024). https://ai.meta.com/blog/meta-llama-3/ - last access 28th october 2024.

OpenAI (2022). https://platform.openai.com/docs/models/gpt-3-5-turbo - last access 28th october 2024.

OpenAI (2024). https://platform.openai.com/docs/models/gpt-4o - last access 28th october 2024.

Piscitelli, A., Costagliola, G., De Rosa, M., and Fuccella, V. (2024). Influence of large language models on programming assignments – a user study. In *ICETC 2024*. ACM.

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., and Santos, E. A. (2023). "it's weird that it knows what i want": Usability and interactions with copilot for novice programmers. 31(1). Place: New York, NY, USA Publisher: Association for Computing Machinery.

Qi, J. Z.-P. L., Hartmann, B., and Norouzi, J. D. N. (2023). Conversational programming with llm-powered interactive support in an introductory computer science course.

Tabari, P., Piscitelli, A., Costagliola, G., and Rosa, M. D. (2025). Assessing the potential of an llm-powered system for enhancing fhir resource validation. In *Studies in Health Technology and Informatics*.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. (2024). Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.