How well does GPT-3.5 perform on course assignments from the TU Delft Computer science and engineering Bachelor?

Finding themes in course assignments GPT-3.5 performs well on and does not perform well on

Mike Segers

Supervisor(s): Fenia Aivaloglou, Xiaoling Zhang

[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

Since large language models (LLM) have been emerged, they took a present role in today's society. From society, they also found their way into the field of education that is why in this research paper, we looked into assignments and exams from the TU Delft Computer science and engineering bachelor and assessed which problems Generative pre-trained transformer (GPT) version 3.5, the current version used by ChatGPT, performs well on (i.e. at least above a pass rate) and on which problems it performs less good (i.e. below pass rate). For our research, we collected assignments by asking professors for consent, to make sure our research was ethically correct. Upon receiving consent, professors had the option to send material, which allowed a deeper analysis, or they could also allow a Brightspace (site where TU Delft courses are hosted) course page scrapping. Once all the questions were gathered, we processed them by prompting them into ChatGPT. We gathered the results and categorized them as wrong or right. We did this all with as few modifications to the questions as possible. The only modifications we did were corrections of copy errors from a PDF, for example: € becoming e after copying. From the results, we found that ChatGPT has its limitations, particularly in large code understanding and complex mathematical reasoning. However, the model performed well in defining concepts and connecting different ideas. We suggest that GPT lacks a comprehensive understanding of coding principles, which hinders its ability to comprehend code. Future work could include exploring other LLMs like GPT-4 and comparing their performance. Further work could also look at assignments from other universities, possibly in different educational fields. Additionally, investigating different prompting techniques to enhance the model's accuracy and reliability could be done as well.

1 Introduction

The integration of Artificial Intelligence (AI) technologies in education has the potential to revolutionize the learning experience, offering new opportunities for student engagement and knowledge acquisition. As AI models like GPT continue to advance, it becomes essential to understand their capabilities and limitations within the context of education. By categorizing these problems into themes, the goal is to incorporate problems that challenge students to collaborate with AI, requiring them to understand and apply concepts rather than either ignoring the existence of AI tools or relying solely on AI tools like ChatGPT. This approach aims to develop an educational environment that embraces AI while ensuring that students meet learning objectives. This is why we will be investigating:

How well does GPT-3.5 perform on course assignments from the TU Delft Computer science and engineering Bachelor?

While the performance and potential applications of GPT models are already been studied [1–10], the answer to the research question of how well GPT-3.5 performs on problems from the TU Delft Computer science and engineering Bachelor is still missing. Evaluating this data set will allow professors to adapt their courses in a specific way so, TU Delft can learn to incorporate AI in their courses. Overall, no study goes in depth about themes of questions GPT can perform well on and not well on and only goes into how well an AI performs or which AI tool performs best. Finding these themes will allow a more general public to find good questions for learning collaboratively with AI.

This research paper focuses on analysing assignments and exams from the TU Delft Computer Science and Engineering bachelor program to assess the performance of Generative pretrained transformer version 3.5 (GPT-3.5), more specifically the ChatGPT model. The aim of the study is to identify the types of problems where the model performs well (i.e. above pass rate) and those where it performs less effectively (i.e. below pass rate).

Relevant material for answering the research question, such as assignments, exams, rubrics, and other course-related information from the TU Delft Computer Science and Engineering (CSE) Bachelor program were obtained through two options presented to professors. Option 1 involved professors contributing their past course material for analysis. Option 2 allowed professors to permit scraping of publicly available course material from Brightspace pages. The chosen language model for the study was ChatGPT from the company OpenAI, specifically GPT version 3.5, as GPT-4 was not publicly available at the time. The prompt design aimed to maintain the originality and integrity of questions, making minor edits only for clarity or context. If an answer seemed incorrect, then an additional chance was given to correct itself. Results were categorized for multiple-choice and open-ended questions, with additional measures taken for cross-referencing and verification to make sure that the categorization happened correctly.

2 Related work

The use of ChatGPT, a language model trained on the GPT-3.5 architecture, has been explored in various educational settings. D. Nunes et al. [1] evaluated the performance of GPT-3.5 and GPT-4 models with several prompting techniques in solving multiple-choice tests, specifically the Brazilian university admission exam, and found that GPT-4 with Chain-of-Thought prompts outperformed GPT-3.5 and different prompting techniques that were tested in the research. This study demonstrates the applicability of language models like GPT-3.5 in multidisciplinary tasks and indicates their potential for improving performance on similar problem-solving assessments in our research. In another study, D.M. Katz et al. [2] evaluated the zeroshot performance of GPT-4 on the entire Uniform Bar Examination and found that it outperformed humans and prior models in all components. This study highlights the capability of GPT models in achieving superior performance in standardized exams, which is very interesting to show that AI is capable to pass harder tests and thus, possibly, the TU Delft CSE bachelor. J. Savelka et al. [3] investigated the capability of GPT models of passing assessments in higher education programming courses, finding that GPT models exhibit remarkable capabilities, including correcting solutions based on an auto-grader's feedback. However, the study also found that GPT models have limitations in handling exercises requiring complex chains of reasoning steps. This study is particularly relevant to our research as it highlights the potential of GPT-3.5 in solving programming problems and emphasizes the importance of evaluating its performance on problems that involve complex reasoning. In yet another study, J. Finnie-Ansley et al. [4] examined the educational implications of AI-generated code for undergraduate computing education using OpenAI Codex. They found that Codex performs well on more advanced data structures and algorithmic problems used in CS2 exams. Overall, these studies demonstrate the potential of ChatGPT and other language models to support education and other complex tasks. However, limitations and areas for improvement also exist, highlighting the need for continued research and development in this field. Furthermore, papers by B. A. Becker [9] and J. Robinson [10] delve into the application of LLMs like GPT-3 in MCQ (multiple choice question) answering tasks. They propose a more natural prompting approach where the LLM is presented with the question and answer options jointly, allowing explicit comparison and reducing computational costs. The papers emphasize the importance of the LLM's multiple choice symbol binding ability to associate answer options with corresponding symbols. Through empirical analysis, they demonstrate that models with high MCSB ability perform significantly better with the natural approach. These findings challenge previous underestimations of LLMs' MCQ answering capabilities. Overall, these papers contribute valuable insights to the literature on utilizing ChatGPT and LLMs in education, offering potential areas for improving teaching and learning practices.

The potential of ChatGPT in education is significant, according to several papers on the topic. Since this research will be conducted in an educational setting, these papers are all relevant to our research. In a survey, Liu et al. [5] found that the models have significant potential in various fields. ChatGPT's adaptability and performance can be enhanced through the integration of Reinforcement Learning from Human Feedback (RLHF), allowing it to offer personalized responses to students. This research shows that RLHF greatly improves the results, which is great with the research goal in mind, since now we have proof that collaborative education between AI and humans is possible and will yield the best results. In a paper by Kasneci et al. [6], ChatGPT is described as highly flexible and capable of generating various things. Its ability to learn from previous conversations and give personalized responses makes it a valuable tool in collaborative education. However, the paper notes that ChatGPT is not perfect and can make mistakes, particularly on simple maths problems, and thus needs to be checked by humans with an understanding of the topic. Roose [7] also emphasizes the potential of large language models (LLM) in education, particularly in improving reading and writing skills, language learning, problem-solving abilities, critical thinking, and collaborative learning. However, Roose also notes ethical considerations and limitations related to interpretability, bias, and risks of misuse. D. Baidoo-Anu [8] conducted a review article that synthesized extant literature, highlighting ChatGPT's remarkable capacity to perform complex tasks in education. The paper emphasizes the potential benefits of ChatGPT, such as promoting personalized and interactive learning, generating prompts for formative assessment activities, and providing ongoing feedback for teaching and learning. However, limitations were also acknowledged, including the generation of wrong information, biases in data training, and privacy concerns. The study provides recommendations for leveraging ChatGPT to maximize teaching and learning, calling for collaboration among policymakers, researchers, educators, and technology experts to ensure safe and constructive use of evolving generative AI tools in education. Overall, ChatGPT has significant potential in education, particularly in providing personalized and effective learning experiences for students.

3 Methodology

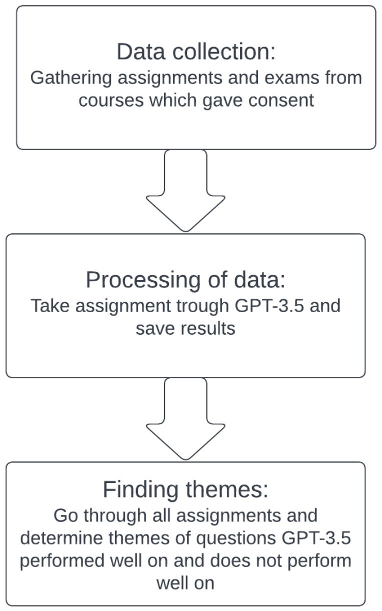
Data collection:
Gathering assignments and exams from courses which gave consent
Processing of data:
Take assignment trough GPT-3.5 and save results
Finding themes:
Go through all assignments and determine themes of questions GPT-3.5 performed well on and does not perform well on

Figure 1: Flowchart to describe the methodology section

3.1 Dataset

To conduct this research, data was collected through a combination of methods. The objective was to obtain relevant material such as assignments, exams, rubrics, and other course-related information from bachelor CSE courses. The dataset gathering process involved seeking the consent of professors, who were approached via email with two options.

Option 1: Material Contribution
Professors were requested to provide material from past course editions, including exams, assignments, rubrics, and any other available resources. Professors who were willing to participate in this manner could submit the relevant material for analysis. By submitting more material, we could go more in depth in evaluating that specific course.

Option 2: Brightspace Page Scraping
Alternatively, professors were given the option to allow the scraping of the TU Delft Brightspace course pages from previous years. Brightspace pages contain publicly (i.e. for all TU Delft students and staff) available course material and resources. By utilizing this method, we could gather openly accessible material to evaluate the course in a less depth intensive matter.

In return for their participation and cooperation, professors were provided with a comprehensive report based on the data collected. The report aimed to offer insights and recommendations for improvement in their courses, tailored specifically to their own data set. This personalized approach ensured that professors received practical feedback that could be directly incorporated into their teaching practices. The ultimate goal of this research was to assist professors in incorporating AI into their own courses, without requiring them to review an entire research paper that may contain general data unrelated to their specific context.

Eventually, six courses gave consent to process their assignments. The courses that went with material contribution were: computer organisation, software engineering and methods and operating systems. The courses which allowed Brightspace scraping were: object oriented programming, concepts of programming languages and software quality and testing. Scanning all material, we've obtained 349 MCQs and 215 open questions for research.

3.2 GPT-model

GPT-3.5 became available for users in March 2022 and has been since then the most used AI tool worldwide. The architecture of GPT-3.5 has 175 billion parameters which were pre-trained on an undisclosed data set which contains millions of websites and billions of words.

In this study, we chose to use the LLM ChatGPT from OpenAI[1]. Its publicly available version is currently (June 2023) based on GPT version 3.5. We thus do not yet use GPT-4 as this version is not openly available to the public yet and can only be accessed with a paid subscription, which most users do not have.

[1]https://openai.com/blog/chatGPT

3.3 Prompt design

During the course of our research, we endeavoured to preserve the originality and integrity of the questions provided as much as possible, since prompting is not the topic under research in this paper.

We made a conscious effort to avoid edits or alterations to the original questions. Our aim was to maintain the clarity and intent of the questions, while seeking answers and insights from the language model. Only minor changes were made if a question appeared unclear or required additional context to produce a meaningful response. For example, copying a € from a PDF would often result in a regular e. We changed it back to a euro sign. The same holds for other non-ASCII tokens. Also, information is often given in the first question of an array of sub-questions. We made sure that the necessary information to solve a sub question was always present within the prompt itself, so that no previous questions were necessary to solve this question. An example of this can be found in Appendix A.

When ChatGPT failed to answer a question, we chose to give it a second retry by simply mentioning ”This answer is wrong, can you please retry”. In this manner, we stay as objectively as we could in the field of prompt engineering [11] but we can see how well the scores become after a retry. We did not delve into a more detailed version of feedback, like for example, the feedback an auto grader would give. This is because not all questions in our dataset had an auto grader. Allowing detailed feedback only to a subset of questions would lead to meaningless results, so we went for the same prompt for all questions.

3.4 Evaluation of results

In the process of analysing the results obtained from the research, categorizing the responses proved to be a significant aspect of the evaluation. Categorization played a crucial role in organizing the data and extracting meaningful insights from the collected answers. However, the process of categorizing differed between multiple-choice questions (MCQs) and open-ended questions.

For MCQs, categorization was relatively straightforward since there was typically a clear and definitive answer provided. The responses could be easily classified into correct or incorrect categories based on the model solution given out by the course. This allowed for a more objective assessment. Since we received too much assignments to process during the time span of the research, we prioritized this kind of data due to the easy processing and objective results.

On the other hand, categorizing the responses to open-ended questions presented a greater challenge. These questions often yielded a range of diverse answers, and it was rare to impossible to find a response that was the exact same as the reference solution. The subjective nature of these questions made it difficult to determine a single correct answer. In such cases, the categorization process involved evaluating the relevance and accuracy of the responses based on the knowledge acquired during the course, combined with cross-referencing with the internet and peers.

Furthermore, it is important to acknowledge that open-ended questions often have the potential for partial correctness. In such instances, where a response contained both correct and incorrect elements, it was treated as a partially correct answer. This approach recognized the effort and understanding while acknowledging any inaccuracies or areas that needed improvement.

By following this methodology, we are doing meaningful research into the field of AI performance. To ensure reproducibility and enable peer reviewing, we explain what the data set is, what GPT version we use and which prompting technique we use. A visual summary of the methodology section is presented in the flowchart below.

3.5 Finding themes

Processing all the assignment and categorizing them as right or wrong can give us valuable insights, but to get to those insights it is essential to have a closer look at the results and identify the specific themes in which the GPT performed well and those in which it did not. This enables us to find where it goes wrong in certain types of questions, rather than the sole true or false result. Identifying significant variations in the performance of the GPT model across different types of questions within a course reveals distinct themes where the model either excels or falls short.

These themes allow for a more focused analysis of the model's strengths and weaknesses in relation to specific question types, enabling to gain a deeper understanding of its capabilities and areas that require improvement.

4 Results

The first course we evaluated was the introductory programming course named object oriented programming. The course is a foundation of the bachelor's as it introduces the concepts of object oriented programming. Throughout the course we learned the Java programming language.

We evaluated the whole course and found interesting results. The most interesting results can be found in the table below. Overall, it passed all individual exams and obtained an average of 61%, which is a passing result but below the average of the course edition during the 2020-2021 year, which was 64%.

The most interesting findings were that ChatGPT lacks to check if code compiles and if certain things happen on compile time or happen during run time for the Java programming language. Possible reasons for this behaviour will be discussed in the discussion section. The worst performance was in large code understanding. For these kinds of questions, we gave ChatGPT a large (i.e. +50 lines) code base and asked questions based on those which require code understanding. In these kinds of questions, ChatGPT even performed less well than a random strategy which would achieve a 25% score since we were evaluating a 4 answer based on a multiple choice exam.

Object oriented programming	
Questions containing compiling understanding	42%
Questions containing large code understanding	17%
Overall score on MCQ questions	61%

Table 1: Most relevant results obtained from evaluating the multiple choice exams of the course object oriented programming

Another course under investigation is the software engineering and methods course. In this course, students learn to create a software architecture and requirements based on a scenario, learn the use of design patterns and learn to analyse their code and improve it.

Since this course gave the most material out of all courses, it was possible to thoroughly investigate the course. We split exam questions into different categories, which often occurred. We also evaluated multiple lab assignments. These assignments are hard to give an exact score as the rubric was not complete and left room for teaching assistants' input. I decided to evaluate them on a scale from 1 to 5. In this system, 1 corresponds to very poor, 3 corresponds to average and 5 corresponds to very good.

Interesting findings were that in both theory and in the practical lab, it performed very well on the design patterns. While looking at the grade distribution per question for the course, design patterns was the lowest scoring part for students. On the other hand, students were very good in the mutation part component, while AI failed to do this. These findings provide a great opportunity for collaborative learning with AI, which is the intent of the research.

Another interesting fact is that GPT-3.5 performs well on software analytics in practice but fails to answer theory about this. This is interesting as the practical aspect builds upon the theoretical background.

Software engineering and methods		
Exam (theory)	Design patters	100%
	Code smells	100%
	Sofware analytics	8%
	Mutation testing	50%
	Overall	69%
Lab (practical)	Domain Driven Design	5
	Design Patterns	4
	Software analytics + Code smells	4
	Mutation Testing	1

Table 2: Most relevant results obtained from evaluating the exams and lab work of the course software engineering and methods

On the evaluation of the course Computer Organisation we saw no particular topics where it performed very well (i.e. above 80%) or parts where it scored very mediocre (i.e. under 50%). With an average performance of 55%, we can see it did not perform very well on this course, but we did not find specific topics it would fail consistently on or get great scores on consistently.

The investigation of the course Operating Systems was really interesting because the open question exam labelled the questions already for us. In this manner, we can easily identify the topics of weakness and strengths for this course.

From the table underneath, it can clearly be seen that it scored pretty good in most aspects with a final grade, taking grade distribution into account, of 7.9. The highest score obtained on a course. Looking at the questions more in depth, it scored very well on questions based on pure definitions and terms but failed hard on questions containing some more challenging mathematics which were mainly found in the I/O, Storage and File Systems section.

Operating systems	
Definitions and concepts	68%
Processes and Threads	50%
Security and Protection	100%
Memory and Virtual Memory	60%
I/O, Storage and File Systems	29%
Connecting concepts	100%
Final score	7.9

Table 3: Results per topic obtained from evaluating the multiple choice exams of the course operating systems

The course "Software Quality and Testing" aimed to provide students with a comprehensive understanding of theoretical testing principles and their practical application. From this evaluation, we can derive the AI's quality in code testing. Something which was only lightly covered during the investigation of the course object oriented programming.

In terms of theoretical testing principles, AI demonstrated significant capabilities. AI algorithms were effective in solving theoretical questions about testing principles and were of great help with coming up with possible test cases.

However, when it came to converting theoretical principles into practical testing, AI failed repeatedly. One of the main issues encountered was the generation of incorrect tests. Despite the promising theoretical foundation, the generated tests often failed to produce the desired results in practice. Many of these tests did not even compile, let alone execute successfully. Even when the generated tests were executable, they frequently scored low on test coverage tests. Test coverage is a crucial metric used to determine the extent to which the software has been tested. Scoring low on this means that the generated tests were not meaningful to ensure bug-free code and thus failed to serve their purpose.

Software quality and testing	
Theory (MCQ)	82%
Theory (Open questions)	71%
Practical testing - Did not compile	62%
Practical testing - 0% score	19%
Practical testing - Avergage score excluding did not compile and 0%	34%

Table 4: Results from evaluation of the course software quality and testing categorized into theory and practical questions.

It appears that for the course Concepts of programming languages material has been removed from public access, resulting in a scarcity of available assignments. As a consequence, results obtained from this investigation may not be significant and do not give meaningful insight.

From the small number of assignments, we found that GPT has difficulties understanding the later stages of the course. This includes mutation and type checking. Earlier content was based on basic parsing, desugaring and interpreting scores better. Again, it is important to notice that these findings are based on a sample size that cannot provide any significance.

In our research, we decided to give GPT another chance when it was wrong to see if it could correct itself based on the information that itss first response was wrong. We found that only with the prompt of stating it was wrong, it had too little information to correct itself. In MCQs, it sometimes apologizes for the mistake and then gives back the exact same answer with a different wording. An example can be found in Appendix B. After analysis, we found that in only 52% of the multiple choice cases, it gave the right result after stating it was wrong. 33% would be achieved by guessing, because out of four options, one was eliminated. In open questions, we only saw a mediocre 12% improvement after stating that it was wrong.

During the research, we have encountered certain themes where the model consistently failed to answer a certain type of question, regardless of the content. Two notable areas wherein the model struggled continuously were large code understanding and complex mathematical reasoning.

When it came to large code understanding, the model often failed to understand code structures, advanced algorithms, or complex software architecture. The model's performance tended to deteriorate as the codebase grew larger and more intricate. In such cases, it often provided incorrect responses with reasoning that made sense for non-programmers but was easily spotted to be wrong for computer scientists.

Similarly, complex mathematical reasoning posed a big challenge for GPT. While it could handle relatively straightforward mathematical questions and calculations, its ability to reason through advanced mathematical concepts was limited. When faced with problems, it often failed to take out the important data correctly, which inevitably led to wrong responses.

On the other hand, a great strength of ChatGPT is its proficiency in addressing questions that can be easily googled, such as retrieving definitions and explaining fundamental concepts. In these cases, the model can provide concise and accurate responses.

Moreover, ChatGPT goes beyond mere information retrieval by demonstrating an impressive ability to connect different concepts and seemingly establish a deeper understanding of course topics. It shows off a remarkable capacity to contextualize information and provide coherent explanations that integrate multiple ideas seamlessly.

This capability to connect concepts is very valuable in educational contexts, where students often struggle to grasp the relationships between different topics. ChatGPT's ability to bridge these gaps and provide comprehensive explanations helps learners develop an understanding of the subject.

It is important to note that while ChatGPT excels in these aspects, it still relies on the information available within its training data and may not always provide the most up-to-date or comprehensive answers. Verification and cross-referencing with trusted sources remain important steps to ensure correctness.

In conclusion, the model consistently struggles with certain question types, irrespective of the course or content. Large code understanding and complex mathematical reasoning pose significant challenges. However, ChatGPT excelled in definitions and explaining fundamental concepts. It also demonstrated the ability to connect concepts, aiding learners in understanding course topics.

Good in	Bad in
Defenitions & Concepts	Large code understanding
Connections between different topics	Complex mathematical reasoning

Table 5: Summary of the themes found in the performance of ChatGPT

5 Responsible Research

Responsible research practices were followed throughout the course of this study, ensuring ethical considerations and consent from all involved parties. In particular, the acquisition of material from professors for research purposes involved a careful approach.

To do ethical research, a consent-seeking process was undertaken to request permission from professors to utilize their material. During this process, clear communication was established, outlining the objectives of the research and the intended use of the data. It is worth noting that professors have the autonomy to decide the extent of their participation based on their preferences and concerns.

Some professors granted permission to use their material under the condition that only publicly available information was processed. They were worried about training a large language model, such as ChatGPT, on their specific answers. These professors knew that the publicly available data would most probably already be entered on platforms as ChatGPT by students undertaking the course, so they saw no harm in the research doing the same, but did not want to train ChatGPT with additional answers.

On the other hand, there were professors who willingly agreed to provide answers and were aware that their responses would be utilized to train the ChatGPT model. They understood the implications and recognized the value of contributing to the advancement of research in natural language processing. Their informed consent and cooperation enabled a richer and more diverse data set for my studies.

By respecting the decisions of the professors and addressing their individual concerns, this research adhered to responsible practices following the standards of the TU Delft.

6 Discussion and limitations

6.1 Discussion

From the results we found many interesting insights. One of the key findings from our study is that GPT-3.5 fails when it comes to understanding and analysing large code samples. When presented with questions or problems that involve extensive code comprehension, the model's responses were often inaccurate or simply wrong. This limitation suggests that GPT-3.5 does not understand programming concepts.

On the other hand, we observed that GPT-3.5 was able to generate code. The model was able to provide correct code snippets when prompted with programming-related tasks, as long as it was a text based prompt with clear instructions on what to program and thus not needing any code understanding.

Another noteworthy observation is that GPT-3.5 occasionally confuses the concepts of compile time and run time. GPT-3.5's confusion may arise due to the model's lack of context-awareness specific to each language. In these cases, GPT may confuse programming languages with each other.

An interesting finding in the context of the software engineering methods course was the successful collaboration between AI and humans. AI and humans scored well in areas where the other performed poorly, indicating the potential for integrating AI into the curriculum. This finding encourages further exploration of how AI can supplement human instruction, aiding students in areas where they typically struggle and enhancing the overall learning experience.

In the course on operating systems, the strength of AI was evident in its ability to connect different concepts and synthesize information. This leads to believe that GPT has great strengths in contextualizing content based on the relation between different concepts. However, when faced with questions involving complex mathematical concepts, the language model struggles to provide accurate answers. This limitation can possibly be explained by the fact that ChatGPT does not understand certain concepts but only sees certain numbers/characters in a certain context and reproduces based on this. It can do basic maths as it is trained on this, but the moment understanding of the topic comes into play to solve this math-based question, it cannot perform well any more.

In our results we often saw that on a retry the model didn't really improve much. From this, we led to believe that the model chose a new option at random. To check this hypothesis, we stated that it was wrong on a question it was right on from the first try. This to see what GPT would do. As expected, GPT took our assumption as the truth and came with a wrong answer. The full conversation can be found in Appendix C. From the improvement of 52% instead of 33%, it is a possible hypothesis to state that ChatGPT can probably eliminate a few options on some questions, which leads to a more 50/50 chance of improving rather than 33%, but when we go to open questions we see only an 12% improvement as the model just has too less information to correct itself.

The research also revealed the difficulty AI encountered in conducting proper software testing. This failure can be attributed to its limited understanding of code principles. As AI lacks a comprehensive understanding of programming concepts, it is unable to effectively test code as humans would. To our belief, this leads back again to the same problem as with large code understanding, since it fails to understand code and thus cannot test it as well.

In conclusion, while the model excels at generating code, it struggles with understanding and analysing large codebases, possibly due to not understanding coding principles. This also makes it impossible to solve questions related to the understanding of code, such as software testing. Additionally, GPT-3.5 occasionally confuses compile time and run time, potentially due to a lack of language-specific context. Even though GPT is great at connecting concepts and giving definitions, it fails to do complicated maths. These findings underscore the need for further research and development to refine the capabilities of language models in computer science and engineering.

6.2 Limitations

Dataset limitations

A potential limitation of this research lies in the scope of the research, which only focuses on a small number of courses from the computer science and engineering bachelor. This narrow sample may not fully capture the diverse range of educational approaches, instruction writing, and course content present in other courses or universities. As a result, the findings and conclusions drawn from this study may not be entirely applicable to the broader curriculum of the bachelor or to other universities.

ChatGPT limitations

During the study, it was revealed that GPT models failed to answer when faced questions that involve images. The inability to process image-based questions reduces the model's overall performance and leads us to only process text-based questions. Incorporating image processing capabilities into language models like ChatGPT could be a valuable addition, considering the large number of image-based questions. This would improve the utility and applicability tremendously.

Furthermore, the research highlighted that GPT models tend to select incorrect answers to multiple-choice questions (MCQs), despite providing the right answer/reasoning in the rest of the prompt. This observation suggests that while LLM models may have sophisticated reasoning abilities, they may still struggle to accurately identify the correct answer among the options presented, as in the learning set GPT might often have seen other options close to the actual answer. An example of this can be found in Appendix D

7 Conclusions and Future Work

7.1 Conclusion

In our research, we looked towards *how well does GPT-3.5 perform on problems from the TU Delft Computer science and engineering Bachelor*. The aim is to identify the types of problems where the model performs well and those where it performs less effectively. The integration of AI technologies in education has great potential, and it is important to understand the capabilities and limitations of AI models like GPT within the context of education. By categorizing the problems into themes, the goal is to understand where these capabilities and shortcomings lye.

For our research, we collected assignments related to the CSE bachelor of the TU Delft. Two options were given to professors. They could either give out material or allow me to scrape content from Brightspace. After we collected these assignments, we processed them through ChatGPT, based on GPT-3.5, because this is the most used LLM worldwide. The originality of the questions was preserved, with only minor changes made for clarity. This to remove bias in prompting. By using this approach, we aimed to enabled reproducibility and peer review.
We found limitations in GPT's ability to answer image-based questions and select correct answers to multiple-choice questions, even with the right reasoning. The field where AI struggles the most is the fields of large code understanding and complex mathematical reasoning. However, AI performed very well at defining concepts and explaining fundamental ideas, showcasing an impressive ability to connect different concepts. Overall, we found that ChatGPT shows great possibilities in the educational field, as other research papers already mentioned in the related work section.

We discussed possible evidence on why the results came out like this and came up with the key finding that ChatGPT does not understand coding principles and thus fails to understand code the way it does with text. This is probably because the training data did not have enough code in it, since it was mostly text. The low score based on difficult maths questions can be explained by the reasoning that it does not actually understand things, but can only place certain characters and words in context.

7.2 Future work

This research paper only focuses on the performance of AI based on the TU Delft CSE bachelor. However, it is important to acknowledge the potential for further exploration beyond GPT-3.5 and compare it with other large language models, such as the upcoming GPT-4. Comparing different models can provide insights into the advancements and improvements made in natural language processing. This could show the potential for upcoming large language models.

As discussed, bias in this research may come from the fact that we have only sampled from the TU Delft CSE bachelor. Exploring different universities and different educational fields may lead to different interesting findings. Since content, way of examining and teaching may differ per university.

Another aspect worth further investigating is the impact of different prompting techniques on the responses generated by the language model. Throughout the research, it was observed that the model's outputs were often close to the correct answer, but occasional mistakes could be easily identified by humans. Exploring the effect of different prompts and approaches to refining the model's responses can help uncover strategies to enhance its accuracy and reliability. Finding good techniques to prompt would greatly improve the AI's overall performance.

References

[1] D. Nunes, R. Primi, R. Pires, R. Lotufo, and R. Nogueira. Evaluating gpt-3.5 and gpt-4 models on brazilian university admission exams, 2023. unpublished.

[2] D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo. Gpt-4 passes the bar exam, 2023. unpublished.

[3] J. Savelka, A. Agarwal, C. Bogart, Y. Song, and M. Sakr. Can generative pre-trained transformers (gpt) pass assessments in higher education programming courses?, 2023. unpublished.

[4] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather, and B. A. Becker. My ai wants to know if this will be on the exam: Testing openai's codex on cs2 programming exercises. In *Proceedings of the 25th Australasian Computing Education Conference*, ACE '23, page 97–104, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, A. Li H. He, M. He, Z. Liu, Z. Wu, D. Zhu, N. Qiang X. Li, D. Shen, T. Liu, and B. Ge. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models, 2023. unpublished.

[6] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler, J. Weller, J. Kuhn, and G. Kasneci. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274, 2023.

[7] K. Roose. The brilliance and weirdness of chatgpt, 2023.

[8] D. Baidoo-Anu and L. O. Ansah. Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning, 2023. unpublished.

[9] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prathers, and E. A. Santos. Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 500–506, New York, NY, USA, 2023. Association for Computing Machinery.

[10] J. Robinson, C. M. Rytting, and D. Wingate. Leveraging large language models for multiple choice question answering, 2023. unpublished.

[11] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023. unpublished.

Appendix

A Not all information present in sub questions

Question

A UNIX file system uses 512 byte (29B) blocks and 8 byte disk addresses. Each inode contains 16 direct entries, one (singly-) indirect block, and one doubly-indirect block.

i. What is the maximum file size in this file system?

ii. If half of the files are exactly 384 Byte $(1.5 \times 2^8 B)$ and the other half are exactly 512 Byte large, what fraction of the disk space would be wasted? Consider only blocks used to store data.

iii. Based on the same conditions as in the question above (ii.), does it help to reduce the fraction of wasted disk space if we change the block size to 256 Bytes?

B Giving back the same response

Question
Given a MouseListener, we can say:
a. MouseListener is an interface; if it contains ¿ 1 method, there is an abstract MouseAdapter class

b. MouseListener is an interface; an abstract MouseAdapter class might exist.

c. MouseListener is a class; if it contains ¿ 1 method, there is a MouseAdapter interface

d. MouseListener is a class; a MouseAdapter interface might exist.

Original responds
The correct answer is:
b. MouseListener is an interface; an abstract MouseAdapter class might exist.

Explanation:
MouseListener is an interface in Java's Swing library, which defines a set of methods that respond to mouse events such as mouse clicks, mouse moves, etc. If a class implements the MouseListener interface, it must implement all the methods in the interface. MouseAdapter, on the other hand, is an abstract class that implements the MouseListener interface. It provides empty implementations for all the methods in the interface, so a programmer can extend MouseAdapter and override only the methods that are necessary for the class, instead of implementing all the methods in the interface.

Therefore, option b is the correct answer.

New responds
Apologies for the incorrect response. Let's reassess the options: a. MouseListener is an interface; if it contains ¿ 1 method, there is an abstract MouseAdapter class.

b. MouseListener is an interface; an abstract MouseAdapter class might exist.

c. MouseListener is a class; if it contains ¿ 1 method, there is a MouseAdapter interface.

d. MouseListener is a class; a MouseAdapter interface might exist.

The correct answer is:
b. MouseListener is an interface; an abstract MouseAdapter class might exist.

Explanation:
The 'MouseListener' is indeed an interface in Java. It defines several methods such as 'mouseClicked', 'mousePressed', 'mouseReleased', etc., that need to be implemented by any class that implements the 'MouseListener' interface. The 'MouseAdapter' class is also a class in Java, but it is not an interface. It is an abstract class that provides empty implementations for all the methods of the 'MouseListener' interface. This allows you to extend the 'MouseAdapter' class and override only the methods you need, rather than implementing all the methods of the 'MouseListener' interface.

So, option (b) correctly states that MouseListener is an interface, and an abstract MouseAdapter class might exist.

C Example correcting right responds

Question
Method overloading means (which statement is the most correct one):
a. In 1 class you can define 2 or more methods with the same name.

b. In 1 class you can define 2 or more methods with the same name, but the names of the parameters need to be different.

c. In 1 class you can define 2 or more methods with the same name, but the types of the parameters need to be different.

d. In 1 class you can define 2 or more methods with the same name, but the parameter lists have to be different in terms of the number of parameters. If the number of parameters is equal, than the types of the parameters need to be different or the order of the types needs to be different.

Original responds

The most correct statement about method overloading is:

d. In 1 class you can define 2 or more methods with the same name, but the parameter lists have to be different in terms of the number of parameters. If the number of parameters is equal, then the types of the parameters need to be different or the order of the types needs to be different.

Explanation:
Method overloading allows you to define multiple methods in a class with the same name but with different parameter lists. The parameter lists must differ in terms of the number of parameters or the types of the parameters, or both. If the number of parameters is different, regardless of their types, you can define multiple methods with the same name.

For example:

```java
public void exampleMethod(int x) {
    // Method implementation
}

public void exampleMethod(int x, int y) {
    // Method implementation
}
```

In the example above, the two methods have the same name, but their parameter lists differ in terms of the number of parameters. If the number of parameters is the same, the types of the parameters must be different or the order of the types needs to be different.

For example:

```java
public void exampleMethod(int x) {
    // Method implementation
}

public void exampleMethod(String x) {
    // Method implementation
}
```

In the example above, the two methods have the same name and the same number of parameters (one parameter), but the types of the parameters differ. Overall, method overloading allows you to define multiple methods with the same name, providing flexibility and allowing for different ways of calling methods based on the arguments provided.

Corrected responds

Apologies for the incorrect response. Let's evaluate the statements again. The most correct statement about method overloading is:

a. In 1 class you can define 2 or more methods with the same name.

Explanation:
Method overloading allows you to define multiple methods in a class with the same name but with different parameter lists. This allows you to provide different implementations of a method based on different parameter types, number of parameters, or both. In method overloading, the names of the methods can be the same, and the parameters can have different names, types, or order. The return type of the method does not play a role in method overloading.

For example:

```java
public class MyClass {
    public void myMethod(int x) {
        // Method implementation
    }

    public void myMethod(String s) {
        // Method implementation
    }

    public void myMethod(int x, String s) {
        // Method implementation
    }
}
```

In the example above, the class 'MyClass' defines three methods with the same name 'myMethod', but each method has a different parameter list. You can then call the appropriate method based on the arguments you pass:

```java
MyClass obj = new MyClass();
obj.myMethod(5);                // Calls
    myMethod(int x)
obj.myMethod("Hello");          // Calls
    myMethod(String s)
obj.myMethod(10, "World");      // Calls
    myMethod(int x, String s)
```

Therefore, statement a. In 1 class you can define 2 or more methods with the same name is the most correct statement regarding method overloading.

D Example failing MCQ with right reasoning

Question
When an exception is thrown (which statement is correct):
a. Execution in the surrounding try-block stops immediately (i.e., any statements after the statement raising the exception are skipped) and control resumes in the catch-block that matches the exception. After completing the catch-block, the finally block is executed (if present). Depending on the situation, control resumes after the catch block (or the finally block if present).

b. Execution in the surrounding try-block stops immediately (i.e., any statements after the statement raising the exception are skipped). Control resumes in the finally-block (if present), once the statements in the block have been executed, control goes to the catch-block that matches the exception. Depending on the situation, control resumes after the catch block (or the finally block if present).

c. Execution in the surrounding try-block stops immediately (i.e., any statements after the statement raising the exception are skipped) and control resumes in the catch-block that matches the exception. After completing the catch-block, the finally block is executed (if present). Depending on the situation, control resumes at the statement immediately after the statement in the try block raising the exception.

d. Execution in the surrounding try-block stops immediately (i.e., any statements after the statement raising the exception are skipped). Control resumes in the finally-block (if present), once the statements in the block have been executed, control goes to the catch-block that matches the exception. Depending on the situation, control resumes at the statement in the try block immediately after the statement raising the exception.

Responds
c. Execution in the surrounding try-block stops immediately (i.e., any statements after the statement raising the exception are skipped) and control resumes in the catch-block that matches the exception. After completing the catch-block, the finally block is executed (if present). Depending on the situation, control resumes at the statement immediately after the statement in the try block raising the exception.