UPPSALA
UNIVERSITET

UPTEC IT 24032
Examensarbete 30 hp
Juni 2024

Can AI models solve the programming challenge Advent of Code?

Evaluating state of the art large language models

Johannes Sandström

Civilingenjörsprogrammet i Informationsteknologi

UPPSALA
UNIVERSITET

Can AI models solve the programming challenge Advent of Code?

Johannes Sandström

Abstract

Large Language Models were developed during the 2010s, and chatbots like ChatGPT quickly became popular. The continued development of LLMs led to tools with specific use cases, one of which is software development.

In this study, eight different LLMs are tested on their ability to solve the programming challenge Advent of Code. Advent of Code consists of 25 problems, each with two parts. Each LLM is given five attempts to try to solve the problem by generating Python code, and after each attempt, feedback is provided to the tools on any issues with the solution.

The results show that ChatGPT-4 and Github Copilot generated the most correct solutions, with ChatGPT-4 generating the most correct solutions on the first attempt. The quality of the code is also examined using SonarQube, and ChatGPT-4 is the best in this regard as well. Of the tools tested in this study, Google's Gemini and Gemini Advanced had the fewest correct solutions. Based on the results of this study, it is clear that these LLMs are good at generating code, but Advent of Code 2023 is too difficult to solve. Despite this, these tools demonstrate that they can be useful for programmers.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: Markuss Sprogis Ämnesgranskare: Erik Hallström

Examinator: Lars-Åke Nordén

Sammanfattning

Large Language Models utvecklades under 2010-talet och chatbotar som ChatGPT blev snabbt populära. Den fortsatta utvecklingen av LLMs ledde till verktyg med specifika användningsområden och ett av dessa är mjukvaruutveckling.

I denna studie testas åtta olika LLMs på deras förmåga att lösa programmeringsutmaningen Advent of Code. Advent of Code innehåller 25 problem där varje problem har två delar. Varje LLM får 5 försök på sig att försöka lösa problemet genom att generera python kod och efter varje försök ges respons till verktygen med eventuella problem med lösningen.

Resultaten visar att ChatGPT-4 och Github Copilot genererade flest korrekta lösningar och ChatGPT-4 genererade flest korrekta lösningar på första försöket. Kodens kvalitet undersöks också genom SonarQube och ChatGPT-4 är bäst även där. Av verktygen som testas i denna studie har Googles Gemini och Gemini Advanced minst antal korrekta lösningar. Baserat på resultaten i denna studie är det tydligt att dessa LLMs är bra på att generera kod men att Advent of Code 2023 är för svår att lösa. Trots detta visar dessa verktyg att de kan göra nytta för programmerare.

Contents

1 Introduction

In the 2010's it was discovered that by training deep neural networks on huge amounts of data, it was possible to make tasks such as image classification and text modeling very effective. This resulted in the development of Large Language Models (LLM) and chatbots like ChatGPT. Further work resulted in models adapted to different areas of expertise and one of these was software development. Integrating the use of Large Language Models in software development has led to several tools that are meant to help the developer in the development process. By using LLMs the developer can write code more efficiently and the productivity significantly increases when using such tools [17]. These tools can be used by developers who aim to learn new techniques and programming languages, and they can also serve as a starting point for beginners learning how to code [11].

Using such tools can help productivity but it is still important to retain code quality. Writing quality code is important since it assures the code and the software to be of good quality and useful for a longer period. This will minimize the cost of the software in the long term [12].

In this thesis, multiple LLMs will be used to generate code for a programming quiz called Advent of Code. Advent of Code is similar to an advent calendar where a new problem is released every day from December 1st to December 25th and each problem has two parts. To fully test these tools' capability to generate code the tools will be given five tries to solve each problem. The code quality of the solutions will also be assessed.

The results show that ChatGPT-4 and GitHub Copilot are the best LLMs tested in this study. Claude 3 Opus also performed well, while Gemini and Gemini Advanced encountered difficulties despite their ranking on the chatbot leaderboard [4].

2 Background

In this section, the scope of the project and necessary background information about the LLMs used in this study is presented. The problems from Advent of Code and code quality are also described.

2.1 Scope

Various aspects need to be addressed when integrating the use of AI and LLMs into the development process. The ethical aspect of using generated code is important, as is ensuring code quality and correctness, which are the developers' responsibility.

This thesis will examine multiple LLMs and their ability to solve the complex programming problems of Advent of Code. Since all of these tools can be used to generate code they all have different abilities and numerous use cases which can be useful in different settings. While there are chatbots like ChatGPT which can be used for multiple purposes, others like GitHub CoPilot are directly focused on coding and can be directly integrated into multiple IDEs. All the LLMs that are tested are chatbots which is a prerequisite for the experiments conducted.

2.2 ChatGPT

When Open AI released their first chatbot ChatGPT[1] in 2022 it immediately became a topic for discussion. After five days it already had 1.000.000 users and since its release, it has been a popular tool used worldwide [15]. ChatGPT allows the user to have a conversation with the chatbot asking questions regarding multiple topics and has a wide range of users. ChatGPT is a tool that can be used for much more than just development, however, it is a popular tool to use among developers.

When ChatGPT was released it was built on the GPT-3.5 model. OpenAI released GPT-4 in March 2023 which led to the release of ChatGPT Plus. ChatGPT-3.5 is free to use while ChatGPT-4 or ChatGPT Plus is available for 20 USD per month and is available on the website. With the more capable ChatGPT-4 the user will obtain a more advanced model with support for images and additional tools. It is also possible to utilise the tool through an API key.

2.3 CoPilot

CoPilot[2] is a tool developed by GitHub together with OpenAI and Microsoft and is built upon GPT and utilises OpenAI's Codex model. The model is trained on all programming languages that appear in repositories but since there are varying amounts of code available from different languages the quality of the generated code may vary. Since

[1] https://chatgpt.com/

[2] https://docs.github.com/en/copilot/overview-of-github-copilot/about-github-copilot-individual

CoPilot is developed by GitHub, its generations are based on information from different public GitHub repositories and base the suggestion on those [7].

Copilot is available as an extension in multiple IDEs and can be accessed with an API but also through the chatbot. When it comes to cost, GitHub has different billing plans depending on the user. For individuals, the cost is 10 USD per month which is their cheapest offer.

CoPilot chat can also be used to explain written code and suggest possible improvements. GitHub also states that CoPilot chat should be used as a tool and not as a replacement for a human developer and the developer should be careful using the generated code since the code might be insecure even though it's correct.

2.4 DeepSeek Coder

DeepSeek Coder is a coding tool developed by DeepSeekAI. The model is trained from scratch on 2 trillion tokens and is available in both English and Chinese. The model is trained on natural language to understand human instructions. DeepSeek Coder exists in three different scales with varying amounts of parameters with varying performance. According to the developers, their most advanced model outperforms other open-source code models [9].

DeepSeek Coder can be used for code completion and code generation etc. directly in the IDE through an API, but can also be accessed using the chatbot[3].

2.5 Gemini

Google released their chatbot, Bard, in 2023 as a response to ChatGPT [20]. Bard was built upon the model Gemini and later the name of the chatbot was changed to Gemini[4]. Gemini both has a normal version, available for free, and an advanced model which comes at a cost. Gemini is powered by the model "1.0 Pro" while Gemini Advanced is powered by "1.0 Ultra" which is more powerful and according to Google, their most capable model [23].

[3] https://chat.deepseek.com/

[4] https://deepmind.google/technologies/gemini/

2.6 Le Chat

Mistral AI[5] is a French tech startup releasing both open-source and commercial models. Similar to the other models Mistral AI can be accessed using an API to integrate the models into the IDE.

Besides these Mistral AI also launched their chatbot called ”Le Chat”[6] [19]. The Beta version was released in February 2024. Le Chat can be used with the commercial models Mistral Large, Mistral Small, or their new model Mistral Next.

2.7 Claude

Claude is a family of AI models developed by Anthropic[7] and their first models were released in 2023. In 2024 Claude-3 was released with three models of varying capability. The most powerful one, Opus, was considered the biggest competitor to GPT-4, which is the most capable model on the market [6].

A chatbot with Claude 3 Opus is available at Poe[8], a platform where multiple models can be accessed.

2.8 Advent of Code

Advent of Code is an annual coding challenge created by Eric Wastl and is available on the website[9]. The Challenge can be seen as a competition where each contestant has to solve two programming problems and the goal is to solve the tasks as fast as possible to end up on the leaderboard. The 100 fastest solvers will reach the leaderboard but it is also possible to participate without trying to be one of the fastest solvers. Solving each part will give the solver a star and trying to get all 50 stars is a way of participating without the time pressure, therefore this challenge is for everyone. A new problem is available every day from December 1 to December 25 and each problem is set in a Christmas environment and has two parts. The descriptions of the problems are all written like a short story where the elves have encountered a problem that needs to be solved. There are no specific requirements for participants, and solving the puzzle each day is voluntary.

[5] https://mistral.ai
[6] https://mistral.ai/news/le-chat-mistral/
[7] https://www.anthropic.com
[8] https://poe.com/about
[9] https://adventofcode.com

2.9 Code Quality

There are various ways of measuring the quality of the code and one common concept is ”Code Quality”. Code Quality is a term that refers to measuring the quality of the code and is used to describe how well-written the code is. Measuring the quality is difficult since the perception of what code quality is differs between different groups. There are several metrics to use when measuring code quality including its reusability, reliability, testability, and portability. With these metrics the quality of the code can be measured and examined.

Writing high-quality code is crucial for developers since it enhances the user experience, furthermore, it also minimizes the maintenance cost of a program since quality code ensures the program is reliable and will fill its purpose for a longer period [24].

There exist several services to measure code quality. Sonar's SonarQube[10] and SonarLint[11] are examples of such. SonarLint is a tool that can be integrated into the IDE and identifies possible problems in the code in real-time and provides a guide on how to fix them. SonarQube also analyzes the code and delivers a detailed analysis of the quality and detects issues such as bugs, code smells, and vulnerabilities.

Sonarsource also defines the concept of Clean Code as: ”Clean code whose attributes make your software reliable, secure, and maintainable.” Consistency, intentionality, adaptability, and responsibility are the four attributes of Clean Code and are meant to ensure the high quality of the software.

ISO-5055:2021 is a standard developed by the Consortium for Information and Software Quality (CISQ) together with 31 other companies. From statistical analysis of source code and structure, ISO-5055 defines four quality measures. These measures assess the reliability, security, performance efficiency, and maintainability of software. These measures ”determine how trustworthy, dependable and resilient a software system will be” according to CISQ.

2.9.1 Validity

Writing valid code is a prerequisite for the code to even compile. Today the IDE has multiple tools that will help the developer by pointing out invalid code in the specific programming language. When using code generation tools to write code it could be difficult for the developer to fix validity issues even if the IDE points out what is wrong.

[10] https://www.sonarsource.com/products/sonarqube/
[11] https://www.sonarsource.com/products/sonarlint/

2.9.2 Correctness

Code Correctness is a metric used to measure the correctness of the code. Correct code is code that performs as intended and a way of assuring this is by writing unit tests. If the code passes the tests the code can be considered correct. Writing correct and error-free code is important. Nguyen and Nadim[16] evaluated code correctness by using LeetCode[12], a website with programming problems and test cases to test the solutions. These test cases are used to evaluate correctness.

”Correctness is the ability of software products to perform their exact tasks, as defined by their specification” [14] is a definition of correctness written by Bertrand Meyer.

2.9.3 Reliability

Chen et al. [3] mentions that reliability is often defined as the probability that the program does not fail in a given environment, during a specified exposure time interval. Another definition is ”the probability that a system will operate without failure for a specified time under specified operating conditions” [27]. SonarSource defines reliability as ”a measure of how your software is capable of maintaining its level of performance under stated conditions for a stated period of time”.

2.9.4 Maintainability

SonarSource defines maintainability as ”Maintainability refers to the ease with which you can repair, improve, and understand software code”. Code smells are code that is not faulty but will probably lead to problems maintaining the code. Fowler and Beck describe the term smell and give examples of where refactoring is needed when code smells appear. They use the term smell to describe and indicate weaknesses in the code that are not technically bugs [2]. Technical debt is the cost introduced each time a developer, instead of rewriting the solution, only does a quick fix [22]. Adding more software will increase the technical debt and will be more difficult to solve. SonarSource defines technical debt as ”Estimated time required to fix all maintainability issues and code smells”[13]

[12] https://leetcode.com

[13] https://docs.sonarsource.com/sonarqube/latest/user-guide/concepts/

3 Related Work

With the development of LLMs and AI, the coding process could be simplified and this has led to the development of multiple models and tools. In this section, similar studies are presented.

In a paper written by Feng et al. [8] the value of ChatGPT is discussed. The authors of this paper investigate the use of ChatGPT in the context of code generation based on data from social media platforms. Another paper on this subject, written by Maher et al. [13], investigates the value of ChatGPT for students learning programming.

In an article written by Atkinson [1], the author tests ChatGPT3.5's ability to fulfill a developer's needs when it comes to coding including debugging and correcting errors. In the article, Atkinson concludes that ChatGPT can be a useful tool for developers.

Using tools to generate code comes with risks. In the article "Six Tips for Better Coding with ChatGPT" by Perkel [18] the author mentions some risks with using ChatGPT and writes that users should trust the responses but that it could be useful to verify them. Give feedback to ChatGPT and propose changes if most likely results in the correct code. Another interesting point the author makes is that ChatGPT is good at finding bugs in a code snippet and also translating code from one language to another.

There also exist studies made on the difference between solutions made by a LLM and a developer. In a study made by Vaithilingam et al.[26] the performance and user experience of using CoPilot is assessed. In this study, the 24 participants were asked to solve one task using CoPilot and another task using another code completion tool called Intellisense. Each participant was given two tasks with varying difficulty and was required to solve each task within 20 minutes. After each task, the participants were asked to answer a survey where they reflected on their experience of using respective tools. From these experiments, the authors concluded that there were no major benefits of using CoPilot but the developers preferred using CoPilot since it often provided a good starting point when solving a programming task.

In a study by Yetistiren et al. [28], CoPilot is tested on the HumanEval dataset, which includes 164 test problems, each with a corresponding human-written solution and unit tests for comparison. Each problem comes with a description and a descriptive function name which will help CoPilot solve the problem. The authors test how well CoPilot can generate a solution with different settings including removing the description and using dummy function names. By doing this they are able to test what information is important for CoPilot to know in order to generate a correct solution. In the study, they also examined the code in terms of code correctness, code validity, and code efficiency. Since the dataset contains a solution to the problem the authors can compare the generated

solution with the human-written one. In this study, the authors found that 91.5% of the code was valid, i.e. had correct syntax, but only 28.7% was correct. The code correctness was tested using the unit tests and the authors defined correct code as code that passed all tests. However, the authors conclude that CoPilot is a promising tool to use when coding.

Recent research made at the beginning of 2024 [10], shows concerning trends in terms of maintainability when using AI as a coding tool. More specifically, GitHub CoPilot is investigated. In a study made by GitClear, where the authors aim to investigate the amount of added code and copy/pasted code increasing compared with updated, deleted, and moved code. These three metrics are, according to the authors, describing maintainability which means the developer updates previous code rather than adding new code. In the research, 153 million changed lines of code are evaluated to investigate maintainability. Another point by the authors regarding moved code is that less moved code implies less reuse. Reusing code helps ensure that new features come with documentation and have been written by other developers, thus maintaining code quality. It is also mentioned that copy/pasting code is more likely to cause future problems with the lack of documentation and evaluation of the written code.

According to research by Thomas Dohmke [5], junior developers are more likely to accept code suggestions from CoPilot. In the same paper, it is also concluded that 75% of developers using CoPilot feel more fulfilled at work and 46% of the code written was completed by suggestions from CoPilot.

In a study made by Yetistirien et al.[29] the three LLMs Amazon CodeWhisperer, ChatGPT, and Github CoPilot are evaluated and compared. In this study, the authors evaluate the tools in terms of code quality metrics including Code Validity, Code Correctness, Code Reliability, Code Security, and Code Maintainability on problems from the HumanEval dataset. This dataset contains 164 problems and each problem comes with an explanation of the problem, a function prototype, and unit tests. They also examined the three tools when they either excluded the documentation provided or used a ”dummy” function name. This was done to evaluate what parts are most important for generating a correct solution.

The results from this study show that Github CoPilot managed to generate valid code for 91.5% of the 164 problems. CodeWhisperer had a success rate of 90.2% and ChatGPT, 93.3%. Regarding Code Correctness it was shown that ChatGPT managed to generate the correct code for 65.2% of the problems, CodeWhisperer generated the correct code for 31.1% and CoPilot generated the correct code for 46.3%.

For security and reliability, the authors did not find results upon which they could conclude. For maintainability ChatGPT, CodeWhisperer, and CoPilot performed equally

well. Bugs did exist in the code generated by all tools. For maintainability code from all tools had some code smells and the technical debt for CoPilot was 9.1 minutes, for CodeWhisperer 5.6 minutes, and 8.9 minutes for ChatGPT. Technical debt is defined as the ”estimated time to resolve the issue”.

In the study the authors found ChatGPT to be the most useful while CodeWhisperer performed worst on the test cases. However, from trying different versions of the software, the authors found that CodeWhisperer is evolving and promising for the future.

4 Methods

In this study, eight LLMs will be examined on their abilities to solve Advent of Code. Github CoPilot, ChatGPT-3.5, ChatGPT-4, Gemini, Gemini Advanced, Claude 3 Opus, Le Chat, and DeepSeek Coder are the chatbots to be tested. A ranking of the chatbots can be found on LMSYS Catbot Arena Leaderboard [4].

To examine these tools based on the code generation, some metrics need to be defined. Code quality and code correctness are areas that are of importance when examining code. Furthermore, documentation and explanations are also of interest since writing maintainable code is important. These metrics will be used to compare the tools to understand the strengths and weaknesses.

4.1 Advent of Code

Each problem has a puzzle input provided which should be used as input to the code. This puzzle input leads to an answer, which is the solution to the specific problem. To solve the ”Advent of Code”-problems an account is needed and when a problem is solved the answer is shown on the website. In this study, the same account is used to solve all the problems which means the answer will be shown on the website if a problem already has been solved. To verify the solution, the answer should be entered on the website or compared to the answer shown. ”Part Two” will only be available if the correct answer to ”Part One” is entered.

In this thesis, multiple LLMs will be assessed on their capability of solving the programming problems from Advent of Code. Each tool will receive the exact description from the website together with basic examples, also available in the description. The description of each problem is written like a story which could be a challenge for the models since there will be much information to process. The description of the problems might

be modified to optimize the code generated from the models.

Since the challenge is a competition among the fastest solvers the author of Advent of Code strongly advises the participants who want to use tools, like ChatGPT, to solve the puzzles to wait until the leaderboard is full. Solving the puzzles with a tool is considered cheating and the participants who aim to make the leaderboard are supposed to create their own solutions. This thesis is written after the 2023 Advent of Code and the 100 fastest solvers have already been crowned, therefore solving these challenges will be the goal of this thesis.

There are some challenges for the tools to correctly solve these problems. The problem description of the Advent of Code problems is written like a short story therefore it requires that the tool can sort out the important parts of the description. This will also show potential use cases of the tool and which type of problems they can manage to solve. Another challenge is if the tools can, based on the description with examples, generate correct code that follows the rules explained in the description.

4.2 Code Quality

For the Advent of Code challenge the code generated by the models will be assessed based on validity, correctness, readability, and reliability.

The tools will also be tested on how many tries they need to generate a correct solution. This will be done by randomly selecting problems upon which the tools struggled to create a correct solution. From this, we can conclude how to fully exploit the tools. There are cases where the user might want to modify the solutions from the code-generating tools. If the tool understood the description wrong or if something in the solution needs to be corrected. Examining the possibility of the tools to fix their problems provides an additional aspect to consider when comparing the tools.

4.2.1 Code Validity

One major requirement we have on the code-generating tools is that they generate valid code. This is essential for the developer to fully rely on the tools and their suggestions. So for the tools to be useful the validity of the code is an important metric.

VSCode together with SonarLint is used to evaluate the validity. This is done by checking if the generated code is valid in terms of the language's syntax rules. The IDE will point out if there are any problems with the code. In the occasion of an error found, this is interpreted as if the code is not valid.

4.2.2 Code Correctness

Measuring code correctness is done by entering the result into the Advent of Code website. The Advent of Code website will only accept the answer if it is correct but will give feedback on the answer if it is wrong. In this context, we evaluate the correctness of the code based on the correct answer according to the website. There could however still be some corner cases or specific inputs for which the solution is not correct. We will solely rate correctness based on the website.

4.2.3 Reliability and Maintainability

Reliability and maintainability will be tested using SonarQube. SonarQube will point out issues related to these categories. For maintainability, SonarQube will point out potential code smells, technical debt and give a maintainability score for the code. Reliability issues will be graded based on bugs and the severity of them. Since it could be difficult to point out these problems in code without thorough code reviews SonarQube will be used. SonarSource, the company behind SonarQube, has definitions for these metrics which can be read about in section 2.

These metrics are considered important because they consider the code quality from a longer time perspective. We want to ensure reliability to have the code work correctly over time without forcing changes. Maintainability is important when the code eventually needs to be changed or rewritten.

4.3 Programming Language

The tools will be asked to generate Python code. Python is chosen since it is one of the most popular programming languages [21, 25] and most models are trained to generate Python code.

4.4 Approach

The description of the Advent of Code problems is written as stories with the task description hidden within the story. The problem descriptions, along with the instructions to generate Python code, are entered into each of the chatbots. If the chatbot manages to generate code, the code will be copied into VSCode[14], which is the IDE used for these

[14]https://code.visualstudio.com

experiments. The IDE will then show potential validity issues. If there are no errors shown by the IDE, the code is considered valid.

The problem description includes a small example. This example is used to test the code before using the puzzle input. If the generated code can solve the example the puzzle input will be used and if this results in the correct result, the code is considered correct. The puzzle input is written in a specific way and it is important to make sure the code will handle this input in the same way it handled the example. In some cases, the code needs to be modified to run correctly with the larger puzzle input.

The output of running the code with the puzzle input is entered into the website and if the answer is correct ”Part Two” is unlocked. If it is not correct or if the code terminated with an error, follow-up questions are asked to help the tool solve the problem. If the result is still wrong after five tries the tool is considered unable to solve the problem.

Code correctness is examined based on the above criteria. Each problem will be rated on how many tries each tool needs to solve the problem. There are three levels:

- The code is correct on the first try
- If the code is correct in 2-5 tries. Follow-up questions are asked to optimize the tool’s chances to solve the problem.
- The tool could not provide a correct solution after five tries.

Each tool will be tested on all available problems from ”Part Two”. However, since Part Two only is available if Part One is correct, the amount of problems tested depends on how well the tools can solve the first part.

After each problem is tested on all tools, SonarQube is used to examine reliability and maintainability. The code that will be run through SonarQube is the final version. This means if the code is not considered correct within 5 tries, the code of the last try will be used to examine reliability and maintainability. If the code is correct then the correct version is used.

5 Results

LLM	Part	Valid	Correct	Problems
CoPilot	Part One	23	11	25
	Part Two	13	5	15
ChatGPT-3.5	Part One	25	5	25
	Part Two	15	2	15
ChatGPT4	Part One	25	11	25
	Part Two	15	5	15
DeepSeek	Part One	25	6	25
	Part Two	15	1	15
Gemini	Part One	23	3	25
	Part Two	13	0	15
GeminiADV	Part One	22	4	25
	Part Two	15	0	15
Le Chat	Part One	23	7	25
	Part Two	14	2	15
Opus	Part One	23	9	25
	Part Two	14	4	15

Table 1 Valid is the number of Advent of Code Problems where no errors were pointed out by the IDE, Correct is the number of Advent of Code Problems the LLM successfully solves. Problems is the number of problems tested on the tool

In table 1 the number of valid and correct solutions on all Advent of Code Problems tested is shown. CoPilot and ChatGPT-4 both solved 11 problems each for Part One and 5 for Part Two while ChatGPT-3.5 could solve 4 Part One-problems and 1 Part Two-problems.

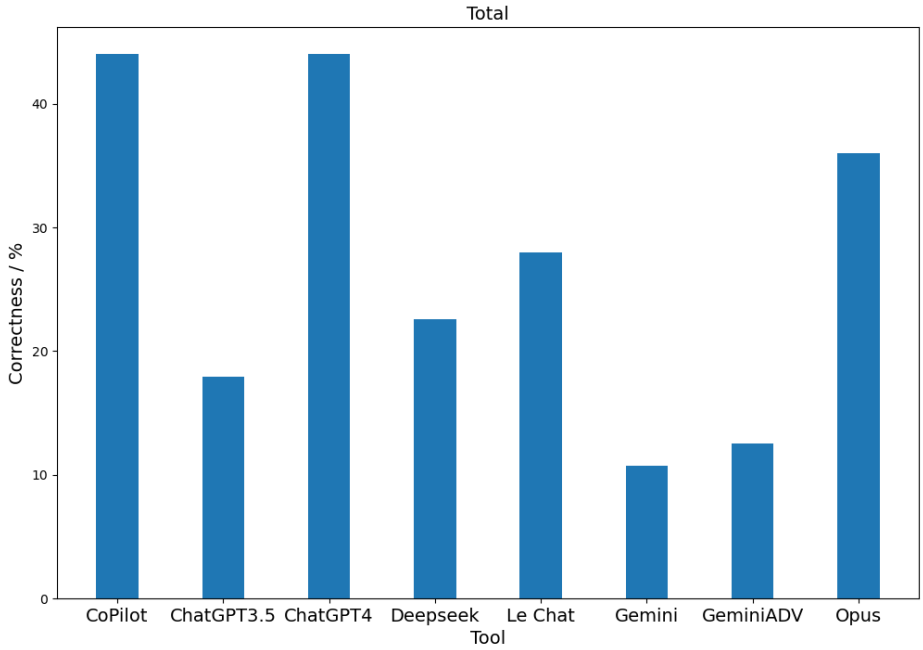

Figure 1 The plot shows the percentage of problems the respective LLM can solve. Problems from both part one and part two in Advent of Code are included

Figure 1 shows the correctness percentage of the solutions generated by the tools. This percentage is calculated by dividing the amount of correct solutions by the total amount of code generations.

Each tool was given five tries to solve each problem. Figure 2 shows the number of tries needed to generate a correct solution for Part One. The figure shows which tools managed to solve more problems on the first try and which tools managed to fix the code from the feedback given. Claude-3-Opus and ChatGPT-4 both solved 7 problems on the first try while Copilot solved 6 problems on the first try. ChatGPT-4 could solve 3 problems on the second try and Gemini and Copilot both generated 2 correct solutions on the second try. Gemini is the only tool that solved more problems on the second try than on the first try.

The result of each try for Part Two is shown in figure 3. ChatGPT-4 generated a correct solution on the first try to 4 of the problems. CoPilot, ChatGPT-3.5, Le Chat, and Claude 3 Opus managed to solve 2 problems on the first try. Gemini and Gemini Advanced were not able to solve any problem from Part Two.

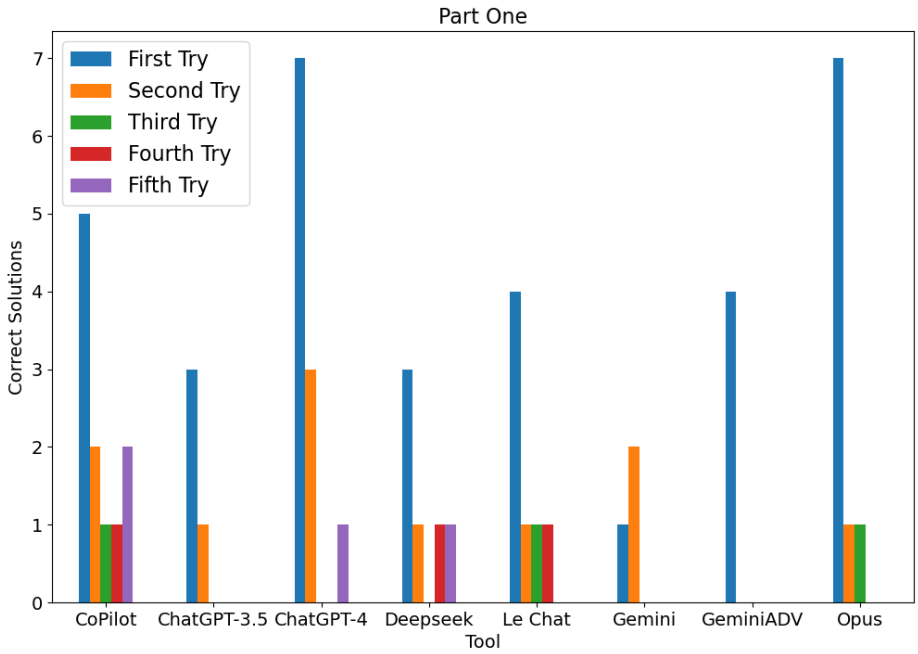

Figure 2 The number of tries required to generate a correct solution for Part One in Advent of Code. Each bar represents how many problems were solved in the respective try. Bars are grouped after the LLM.

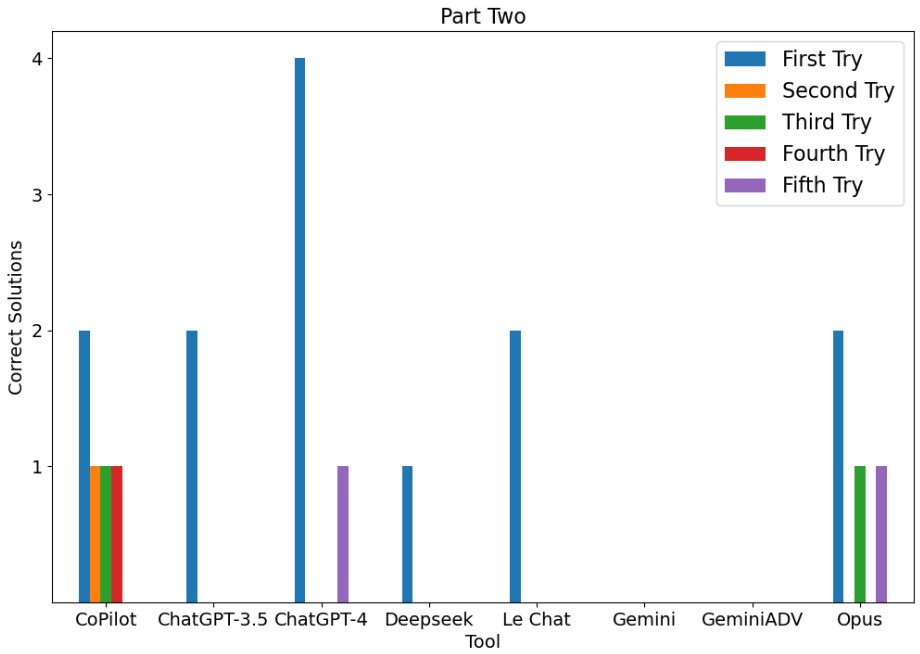

Figure 3 The number of tries required to generate a correct solution for Part Two in Advent of Code. Each bar represents how many problems were solved in the respective try. Bars are grouped after the LLM.

Figure 4 shows the number of correct solutions generated by each tool on the respective part. ChatGPT-4 and CoPilot both generated 11 correct solutions on Part One. These two also performed best on Part Two with five correct solutions. The figure also shows that Gemini with 3 correct solutions for Part One and zero for Part Two, was the tool that managed to generate the least amount of correct solutions.

Dividing the problems of Part One into 3 parts, shown in figure 5 shows that all tools solved more problems in the first 9 days compared to days 10-17 and day 18-25. CoPilot and ChatGPT-4 both solved 8 of the first 9 problems. Days 10-17, all tools managed to solve at least one problem, CoPilot and Opus solved 3 of these problems. Days 18-25, Deepseek and ChatGPT-4 were the only tools generating a correct solution.

Figure 2 shows the number of tries required to generate a correct solution. Each new try was based on feedback from the earlier tries. If the answer was still wrong after 5 tries, the problem was considered unsolvable.

Below, the problems are briefly described and information about each try is shown.

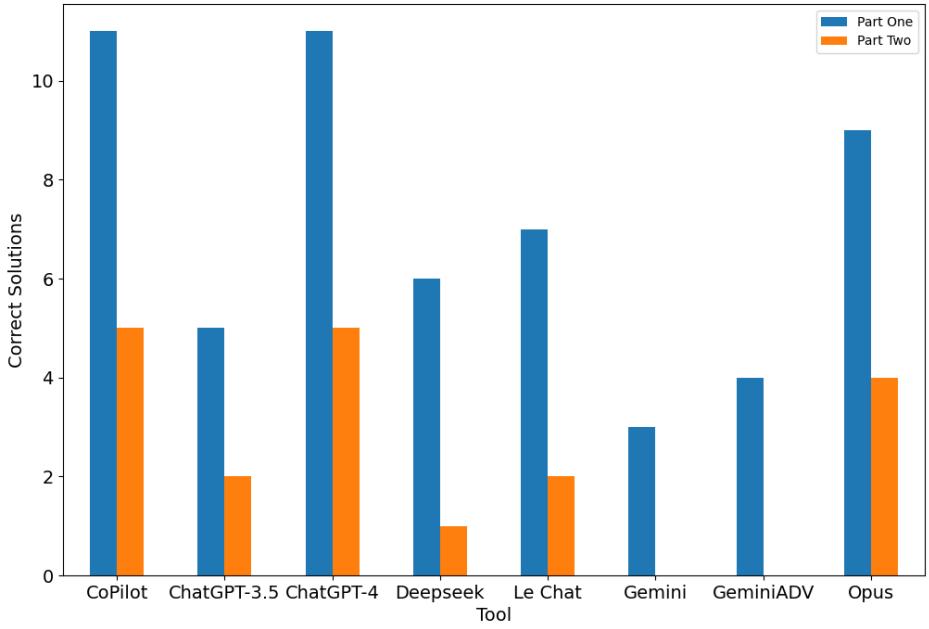

Figure 4 Number of correct solutions of both parts

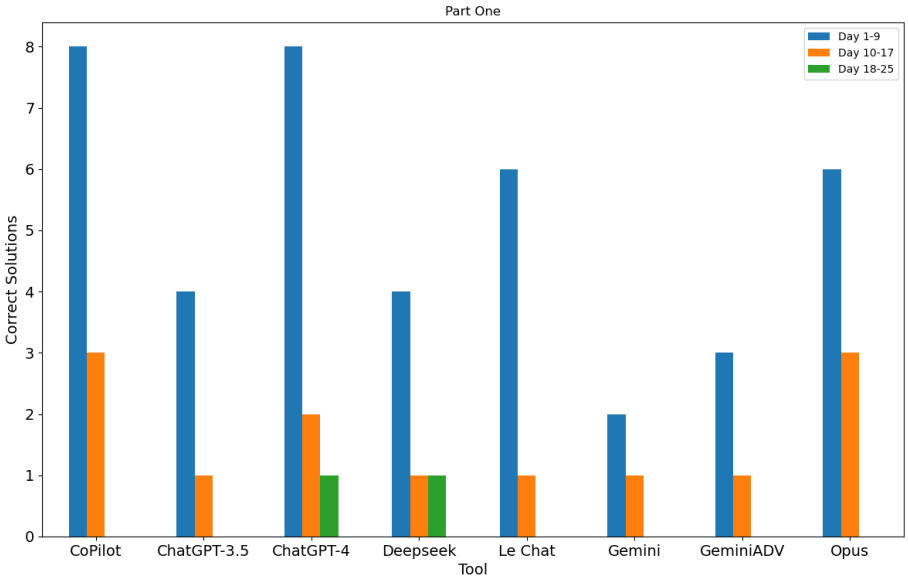

Figure 5 Correct solutions of Part One divided into three parts.

There are several possible outcomes of each try shown in the tables:

- **Error**: Running the provided code resulted in an error.
- **WA**: The output were the wrong answer
- **Correct**: The code were correct

Some days the task was to loop over very large numbers which resulted in the code not terminating. If the solution failed to terminate within 10 minutes the try was considered as the wrong answer and the tool was asked to optimise the solution.

5.1 Problems Part 1

5.1.1 Day 1[15]

The first problem is about finding the first and last integer in a string and creating a two-digit number. If there only is one integer in the string, that integer would be counted as both first and last. These integers should then be added together to a resulting number. Table 2 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	WA	WA	WA	ValueError	**Correct**	ValueError	**Correct**
2	-	WA	WA	WA	WA	-	**Correct**	-
3	-	WA	WA	WA	**Correct**	-	-	-
4	-	WA	WA	**Correct**	-	-	-	-
5	-	WA	WA	-	-	-	-	-

Table 2 Output of each try for Day 1 Part One

5.1.2 Day 2[16]

From a bag with cubes of three different colors, a random amount is picked. For Day 2 the task is to check if a possible game is possible or not based on a fixed amount of cubes and a game record. If the game record contains 14 red cubes but the bag only has 12 red cubes, the game is not possible. Table 3 shows the output of each try from all the tools.

[15]https://adventofcode.com/2023/day/1
[16]https://adventofcode.com/2023/day/2

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	ValueError	**Correct**	ValueError	**Correct**	**Correct**	**Correct**	ValueError
2	ValueError	ValueError	-	**Correct**	-	-	-	**Correct**
3	TypeError	ValueError	-	-	-	-	-	-
4	**Correct**	WA	-	-	-	-	-	-
5	-	WA	-	-	-	-	-	-

Table 3 Output of each try for Day 2 Part One

5.1.3 Day 3[17]

The task is to find out if numbers are adjacent to symbols in a pattern of numbers and symbols. The numbers that are adjacent to a symbol are then added together producing the sum which is the answer. Table 4 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	WA	WA	WA	WA	IndexError
2	WA	WA	WA	WA	WA	**Correct**	WA	WA
3	WA	WA	WA	WA	WA	-	WA	WA
4	WA	WA	WA	WA	WA	-	WA	WA
5	**Correct**	WA	WA	WA	WA	-	WA	WA

Table 4 Output of each try for Day 3 Part One

5.1.4 Day 4[18]

Day 4 is about the winning numbers of scratchcards. Each input includes two ”lists” of numbers where the first list is the winning numbers and the other list is your numbers. The task is to find out how many winning numbers you have. Table 5 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	WA	ValueError	**Correct**	**Correct**	WA	**Correct**	**Correct**
2	-	**Correct**	WA	-	-	**Correct**	-	-
3	-	-	WA	-	-	-	-	-
4	-	-	WA	-	-	-	-	-
5	-	-	WA	-	-	-	-	-

Table 5 Output of each try for Day 4 Part One

[17] https://adventofcode.com/2023/day/3
[18] https://adventofcode.com/2023/day/4

5.1.5 Day 5[19]

The task is to correctly map and convert numbers from one category to another and find out the lowest location which is the last step of the conversions. Table 6 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	TypeError	**Correct**	WA	**Correct**	**Correct**	TypeError	WA
2	-	TypeError	-	WA	-	-	WA	ValueError
3	-	Attribute	-	WA	-	-	ValueError	WA
4	-	WA	-	WA	-	-	WA	WA
5	-	WA	-	WA	-	-	WA	WA

Table 6 Output of each try for Day 3 Part One

5.1.6 Day 6[20]

The problem is about toy boats and how to beat the record. Each race has a specified duration and the the boat accelerates faster if you hold the boat in the beginning, therefore the task is to calculate all the possible solutions to beat the record. Table 7 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	WA	ValueError	**Correct**	**Correct**	**Correct**	**Correct**	WA
2	-	WA	ValueError	-	-	-	-	WA
3	-	WA	ValueError	-	-	-	-	WA
4	-	WA	WA	-	-	-	-	WA
5	-	WA	WA	-	-	-	-	WA

Table 7 Output of each try for Day 6 Part One

5.1.7 Day 7[21]

The task is to calculate the total winnings in a game which is similar to poker. The goal is to calculate the highest hands and bids based on the rules of the game. Table 8 shows the output of each try from all the tools.

[19] https://adventofcode.com/2023/day/5
[20] https://adventofcode.com/2023/day/6
[21] https://adventofcode.com/2023/day/7

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	SyntaxError	WA	TypeError	NameError	TypeError	WA	WA	KeyError
2	WA	WA	NameError	ValueError	WA	WA	WA	WA
3	WA	WA	ValueError	ValueError	WA	WA	WA	WA
4	WA	WA	ValueError	ValueError	TypeError	WA	WA	WA
5	WA	WA	ValueError	ValueError	WA	WA	WA	WA

Table 8 Output of each try for Day 7 Part One

5.1.8 Day 8[22]

This day the task is to navigate from a starting point to the goal. Navigating is done by following ”Right” and ”Left” instructions and following these given instructions you will eventually end up at the goal. The goal is to find out how many steps are needed to reach the goal. Table 9 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	**Correct**	**Correct**	**Correct**	**Correct**	WA	ValueError
2	**Correct**	**Correct**	-	-	-	-	WA	WA
3	-	-	-	-	-	-	WA	WA
4	-	-	-	-	-	-	WA	WA
5	-	-	-	-	-	-	WA	**Correct**

Table 9 Output of each try for Day 8 Part One

5.1.9 Day 9[23]

The task is to extrapolate numbers based on a given sequence. This is done by adding rows with the difference of the numbers in the original input. The extrapolated value is calculated when the difference is zero. Table 10 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	IndexError	WA	WA	WA	WA	WA
2	WA	WA	WA	WA	WA	**Correct**	WA	WA
3	WA	WA	WA	**Correct**	WA	-	WA	WA
4	WA	WA	IndexError	-	WA	-	WA	WA
5	**Correct**	WA	WA	-	IndexError	-	WA	WA

Table 10 Output of each try for Day 9 Part One

[22]https://adventofcode.com/2023/day/8
[23]https://adventofcode.com/2023/day/9

5.1.10 Day 10[24]

Here the task is to follow a pattern of symbols finding a loop connected to the starting point. Finding the farthest point from the starting position and calculating the steps needed to reach that point is the goal of the task. Table 11 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	KeyError	WA	WA	WA	WA	WA	RecursionError	KeyError
2	TypeError	WA	WA	WA	WA	WA	WA	WA
3	**Correct**	WA	WA	WA	WA	WA	WA	WA
4	-	WA	WA	WA	RecursionError	WA	WA	WA
5	-	WA	WA	WA	WA	WA	WA	WA

Table 11 Output of each try for Day 10 Part One

5.1.11 Day 11[25]

The task of Day 11 is to find pairs of ”galaxies” and calculate the shortest path between them. Summing the shortest paths of all pairs results in the answer. Table 12 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	TypeError	WA	IndexError	TypeError	WA	WA	WA
2	WA	KeyError	WA	IndexError	**Correct**	**Correct**	WA	WA
3	KeyError	KeyError	WA	IndexError	-	-	WA	WA
4	WA	WA	WA	IndexError	-	-	WA	WA
5	WA	WA	WA	IndexError	-	-	WA	WA

Table 12 Output of each try for Day 11 Part One

5.1.12 Day 12[26]

This task is about figuring out which arrangements of springs fit the criteria. Both the springs and the criteria are found in the input. Table 13 shows the output of each try from all the tools.

[24]https://adventofcode.com/2023/day/10
[25]https://adventofcode.com/2023/day/11
[26]https://adventofcode.com/2023/day/12

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	IndexError	WA	ValueError	**Correct**	Error	WA	WA
2	WA	IndexError	WA	ValueError	-	Error	WA	WA
3	IndexError	IndexError	WA	ValueError	-	Error	WA	WA
4	WA	IndexError	WA	WA	-	Error	WA	WA
5	WA	IndexError	WA	IndexError	-	Error	WA	WA

Table 13 Output of each try for Day 12 Part One

5.1.13 Day 13[27]

Here the task is to find a line of reflection vertically or horizontally. The input is a pattern and the line of reflection is found where two columns or rows are a reflection of each other. Table 14 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	IndexError	NameError	WA	WA	Error	WA	WA
2	IndexError	IndexError	NameError	WA	WA	Error	WA	WA
3	WA	IndexError	WA	WA	WA	Error	WA	WA
4	IndexError	NameError	WA	WA	WA	Error	WA	WA
5	WA	IndexError	WA	WA	WA	Error	WA	WA

Table 14 Output of each try for Day 13 Part One

5.1.14 Day 14[28]

The task is to tilt a platform with rocks and calculate the load after the tilt. The platform contains round rocks that roll when the platform is tilted and cube-shaped rocks that will stay where they are. Table 15 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	IndexError	WA	WA	WA	WA
2	**Correct**	WA	WA	IndexError	WA	WA	WA	WA
3	-	WA	WA	WA	WA	WA	WA	WA
4	-	WA	ULE	WA	WA	WA	WA	WA
5	-	WA	ULE	WA	WA	WA	WA	WA

Table 15 Output of each try for Day 14 Part One. ULE is UnboundLocalError

[27] https://adventofcode.com/2023/day/13
[28] https://adventofcode.com/2023/day/14

5.1.15 Day 15[29]

For Day 5 the task is to calculate the resulting numbers using a HASH algorithm. The input contains combinations of letters, symbols, and numbers and corresponds to a value. These values are summed to produce the result. Table 16 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	**Correct**	**Correct**	**Correct**	**Correct**	**Correct**	**Correct**	**Correct**
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-

Table 16 Output of each try for Day 15 Part One

5.1.16 Day 16[30]

This problem is about finding out how many tiles become energized after a beam of light enters a grid. This grid contains different symbols making the light bounce around. Tiles are energized if the beam passes through them. Table 17 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	TypeError	RecursionError	TypeError	WA	WA	WA	WA	KeyError
2	KeyError	WA	RecursionError	WA	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	WA	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 17 Output of each try for Day 16 Part One

5.1.17 Day 17[31]

Here the task is about minimizing heat loss when carrying lava from one point to another. The input consists of a map where the heat loss is marked. The lowest possible heat loss is the answer to the task. Table 18 shows the output of each try from all the tools.

[29] https://adventofcode.com/2023/day/15
[30] https://adventofcode.com/2023/day/16
[31] https://adventofcode.com/2023/day/17

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	IndexError	NameError	WA	SyntaxError	WA	WA	WA
2	WA	IndexError	WA	WA	WA	WA	WA	WA
3	WA	IndexError	WA	WA	WA	WA	IndexError	WA
4	WA	AttributeError	WA	ULE	WA	WA	WA	WA
5	WA	AttributeError	WA	ULE	WA	WA	WA	WA

Table 18 Output of each try for Day 17 Part One. ULE is UnboundLocalError

5.1.18 Day 18[32]

The problem is about calculating the amount of lava that will fit in a lagoon after following a dig plan. Each row of the dig plan, which is the input, contains directions and how many steps to take in the specified direction. This results in a lagoon and the task is to calculate the size of the lagoon. Table 19 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	RecursionError	ValueError	WA	ValueError	WA	WA	WA	ValueError
2	RecursionError	ValueError	ValueError	NameError	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	WA	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 19 Output of each try for Day 18 Part One

5.1.19 Day 19[33]

The task of Day 19 is about finding the rating of the accepted parts. This is done by following a workflow and part ratings. These are the input. Following the workflow the parts will be accepted or not, then the sum of the ratings of the accepted parts is calculated. Table 20 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	NameError	KeyError	ValueError	ValueError	ValueError	Error	ValueError	WA
2	IndexError	ValueError	WA	ValueError	ValueError	WA	ValueError	KeyError
3	IndexError	ValueError	WA	KeyError	KeyError	WA	WA	WA
4	IndexError	ValueError	ValueError	WA	WA	WA	WA	WA
5	SyntaxError	ValueError	ValueError	ValueError	WA	**Correct**	WA	WA

Table 20 Output of each try for Day 19 Part One

[32] https://adventofcode.com/2023/day/18
[33] https://adventofcode.com/2023/day/19

5.1.20 Day 20[34]

After pushing a button, pulses are sent. Based on a module configuration, the input, the task is to find out the number of low pulses and high pulses sent after pushing the button 1000 times. Table 21 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	TypeError	ValueError	WA	KeyError	KeyError	WA	TypeError	IndexError
2	TypeError	AttributeError	TypeError	KeyError	RuntimeError	WA	KeyError	ULE
3	AttributeError	KeyError	WA	KeyError	WA	WA	WA	ULE
4	KeyError	WA	WA	TypeError	WA	WA	NameError	KeyError
5	KeyError	WA	WA	WA	WA	WA	WA	KeyError

Table 21 Output of each try for Day 20 Part One. ULE is UnboundLocalError.

5.1.21 Day 21[35]

Here the task is about finding out how many garden plots can be reached in 64 steps. The input is a map with a starting position, garden plots, and rocks. The goal is to visit as many garden plots as possible in 64 steps. Table 22 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	-	IndexError	WA	WA	WA	WA	WA	WA
2	-	IndexError	WA	WA	WA	WA	WA	WA
3	-	IndexError	WA	WA	WA	WA	WA	WA
4	-	WA	WA	WA	WA	WA	WA	WA
5	-	WA	WA	WA	WA	WA	WA	WA

Table 22 Output of each try for Day 21 Part One

5.1.22 Day 22[36]

The task is to find out which bricks can be safely disintegrated. Based on the input, which is a snapshot of the bricks while falling, the goal is to find out where they will land and if they are safe to disintegrate. A brick is safe to disintegrate if no other bricks fall if it is removed. Table 23 shows the output of each try from all the tools.

[34]https://adventofcode.com/2023/day/20
[35]https://adventofcode.com/2023/day/21
[36]https://adventofcode.com/2023/day/22

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	TypeError	IndexError	WA	WA	WA	WA	WA
2	WA	TypeError	KeyError	WA	WA	WA	WA	WA
3	WA	TypeError	WA	WA	WA	WA	WA	WA
4	WA	ValueError	WA	WA	WA	WA	WA	WA
5	WA	ValueError	WA	WA	WA	WA	WA	WA

Table 23 Output of each try for Day 22 Part One

5.1.23 Day 23[37]

The task is to calculate the longest possible hike based on a map of hiking trails. The map contains the paths along with forest and steep slopes which can only be used in one direction. Table 24 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	ULE	ULE	WA	WA	WA	WA	WA
2	WA	ULE	ValueError	WA	WA	WA	WA	WA
3	WA	ULE	WA	WA	WA	WA	WA	WA
4	WA	ULE	WA	WA	RecursionError	WA	WA	**Correct**
5	WA	ULE	NameError	WA	WA	WA	WA	-

Table 24 Output of each try for Day 23 Part One. ULE is UnboundLocalError.

5.1.24 Day 24[38]

Here the task is about finding out if the paths of hailstones will cross within a test area. The input contains the position of the hailstone and its speed. Based on this information their paths can be calculated and the intersections can be found. Table 25 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	ValueError	WA	TypeError	WA	ValueError	WA	WA	WA
2	WA	WA	WA	AttributeError	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	WA	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 25 Output of each try for Day 24 Part One

[37] https://adventofcode.com/2023/day/23
[38] https://adventofcode.com/2023/day/24

5.1.25 Day 25[39]

The last task is about finding out which wires need to be disconnected to separate components into two separate groups. The input is a wiring diagram showing all components and their connections. After separating the components the size of the groups is multiplied resulting in the answer. Table 26 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	KeyError	WA	ValueError	RuntimeError	ValueError	WA	KeyError	WA
2	WA	WA	KeyError	ValueError	ValueError	WA	KeyError	WA
3	RecursionError	WA	NameError	Runtime	IndexError	WA	KeyError	WA
4	TypeError	WA	WA	WA	WA	WA	KeyError	RuntimeError
5	WA	WA	WA	WA	KeyError	WA	KeyError	WA

Table 26 Output of each try for Day 25 Part One

5.2 Problems Part Two

15 of the 25 Part One problem were solved therefore 15 Part Two problems were tested.

5.2.1 Day 1

This task is to change the calculation from Part One to account for digits that are spelled out as words. This will result in another calibration value. Table 27 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	WA	WA	WA	WA	WA
2	WA	WA	WA	ValueError	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	ValueError	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 27 Output of each try for Day 1 Part Two

[39] https://adventofcode.com/2023/day/25

5.2.2 Day 2

The task is to calculate the least amount of cubes needed to play the game. The answer is now the amount of red, green, and blue cubes multiplied together. Table 28 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	WA	WA	**Correct**	**Correct**	**Correct**	**Correct**	**Correct**
2	-	WA	WA	-	-	-	-	-
3	-	WA	WA	-	-	-	-	-
4	-	WA	WA	-	-	-	-	-
5	-	WA	WA	-	-	-	-	-

Table 28 Output of each try for Day 2 Part Two

5.2.3 Day 3

For Part Two the task is to calculate the gear ratio of all the gears and add them together. Table 29 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	WA	TypeError	WA	WA	WA
2	WA	WA	WA	WA	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	WA	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 29 Output of each try for Day 3 Part Two

5.2.4 Day 4

For Part Two there are n longer any points, instead, you win more scratchcards. After no more scratchcards are won the amount of scratchcards you end up with is the answer. Table 30 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	ValueError	WA	WA	WA	WA	WA
2	WA	WA	WA	WA	WA	WA	WA	WA
3	WA	WA	WA	WA	**Correct**	WA	WA	WA
4	**Correct**	WA	WA	WA	-	ValueError	WA	WA
5	-	KeyError	TypeError	WA	-	**Correct**	WA	WA

Table 30 Output of each try for Day 4 Part Two

5.2.5 Day 5

Now the conversion calculations are different from Part One. All seeds can be seen as pairs where the first number is the start point and the other number is the length of the range resulting in many more seeds. Table 31 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	ValueError	WA	WA	IndexError	WA
2	WA	WA	WA	IndexError	WA	WA	WA	ValueError
3	WA	WA	WA	ValueError	WA	WA	WA	KeyError
4	WA	WA	WA	ValueError	WA	ValueError	WA	WA
5	WA	WA	WA	ValueError	WA	WA	WA	KeyError

Table 31 Output of each try for Day 5 Part Two

5.2.6 Day 6

Now there is only one race and the input should be put together to only produce one big number. The task is still to calculate how many ways you can beat the record. Table 32 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	**Correct**	WA	WA	**Correct**	**Correct**	**Correct**	**Correct**	WA
2	-	WA	WA	-	-	-	-	WA
3	-	WA	NameError	-	-	-	-	WA
4	-	WA	WA	-	-	-	-	WA
5	-	WA	WA	-	-	-	-	WA

Table 32 Output of each try for Day 6 Part Two

5.2.7 Day 8

There are multiple starting nodes and there are as many starting nodes as there are end nodes. Now the task is to start at all starting nodes simultaneously. Table 33 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	WA	WA	WA	WA	WA
2	WA	WA	WA	WA	WA	WA	WA	WA
3	WA	WA	WA	WA	WA	WA	WA	WA
4	WA	WA	WA	WA	WA	WA	WA	WA
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 33 Output of each try for Day 8 Part Two

5.2.8 Day 9

For Part Two the task is to extrapolate backwards finding the first values of the sequences. Table 34 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	IndexError	WA	WA	**Correct**	WA	WA
2	**Corerct**	WA	WA	WA	WA	-	WA	WA
3	-	WA	WA	WA	WA	-	WA	WA
4	-	WA	WA	WA	WA	-	WA	KeyError
5	-	WA	WA	WA	WA	-	WA	WA

Table 34 Output of each try for Day 9 Part Two

5.2.9 Day 10

Now the task is to calculate how many tiles are enclosed by the loop. Table 35 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	WA	WA	WA	WA	WA	KeyError
2	WA	WA	WA	WA	WA	WA	WA	KeyError
3	WA	WA	WA	WA	WA	WA	WA	TypeError
4	WA	WA	WA	WA	WA	WA	WA	KeyError
5	WA	WA	WA	WA	WA	WA	WA	WA

Table 35 Output of each try for Day 10 Part Two

5.2.10 Day 11

The galaxies are further apart therefore the map needs to be 1 million times larger. The task is still to calculate the sum of the shortest paths. Table 36 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	-	WA	WA	WA	WA	AttributeError	WA
2	WA	-	WA	WA	WA	WA	AttributeError	WA
3	WA	-	WA	WA	WA	WA	WA	WA
4	NameError	-	WA	WA	WA	WA	IndexError	WA
5	WA	-	WA	WA	WA	WA	IndexError	WA

Table 36 Output of each try for Day 11 Part Two

5.2.11 Day 12

For Part Two there are more possible arrangements. The input should be replaced with five copies of itself. The task is to calculate the sum of possible arrangements. Table 37 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	TypeError	TypeError	IndexError	WA	WA	WA	WA
2	WA	IndexError	TypeError	WA	IndexError	WA	TypeError	WA
3	WA	WA	TypeError	WA	IndexError	WA	ValueError	WA
4	WA	WA	TypeError	WA	WA	WA	WA	WA
5	WA	IndexError	TypeError	WA	IndexError	WA	WA	WA

Table 37 Output of each try for Day 12 Part Two

5.2.12 Day 14

Now the platforms are tilted in a cycle, north, west, south, and east. The task is to calculate the load after 1000000000 cycles. Table 38 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	WA	NameError	WA	WA	WA	WA	WA
2	WA	WA	NameError	WA	WA	WA	WA	WA
3	WA	WA	NameError	WA	WA	WA	WA	WA
4	WA	WA	NameError	WA	WA	WA	WA	WA
5	WA	WA	NameError	WA	WA	WA	WA	WA

Table 38 Output of each try for Day 14 Part Two

5.2.13 Day 15

The task is still to follow an algorithm but the procedure is a bit different. Now the task is to calculate the focal length of the lenses. If the combinations of letters and symbols include an equal sign the number following is the focal length. The task is to calculate the focusing power of the lens configuration. Table 39 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	NameError	WA	WA	WA	WA	**Correct**	WA	WA
2	NameError	WA	WA	WA	WA	-	WA	WA
3	**Correct**	WA	WA	WA	ValueError	-	WA	WA
4	-	WA	WA	WA	WA	-	WA	WA
5	-	WA	WA	WA	**Correct**	-	WA	WA

Table 39 Output of each try for Day 15 Part Two

5.2.14 Day 19

Now the task is to find all the parts that will be accepted of all possible combinations of ratings. Table 40 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	-	WA	ValueError	-	ValueError	WA	ValueError	TypeError
2	-	WA	ValueError	-	WA	WA	IndexError	ValueError
3	-	WA	ValueError	-	WA	WA	WA	ValueError
4	-	WA	ValueError	-	WA	WA	IndexError	ValueError
5	-	WA	ValueError	-	WA	WA	IndexError	ValueError

Table 40 Output of each try for Day 19 Part Two

5.2.15 Day 23

For Part Two the slopes are treated as normal paths meaning they can be climbed both ways. The task is still to find the longest possible hike. Table 41 shows the output of each try from all the tools.

Try	CoPilot	Gemini	GeminiADV	Le Chat	Opus	GPT-4	GPT-3.5	DeepSeek
1	WA	IndexError	WA	NameError	WA	WA	WA	ValueError
2	WA	IndexError	WA	WA	WA	WA	WA	WA
3	WA	IndexError	WA	TypeError	WA	WA	WA	WA
4	WA	IndexError	WA	WA	WA	WA	WA	WA
5	RecursionError	IndexError	WA	RecursionError	WA	WA	WA	WA

Table 41 Output of each try for Day 23 Part Two

5.3 CoPilot

CoPilot failed to generate code for two problems. To Day 21 Part One CoPilot couldn't generate code because the response was blocked. The following error message was returned: "Sorry, the response matched public code so it was blocked. Please rephrase your prompt.". To Day 19 Part Two no code was generated due to the high complexity of the problem. Of the 38 solutions where code where generated, 36 were valid.

In the first 11 days, CoPilot could solve 10 of the Part One problems within five tries. Of these 10 correct solutions, four were correct on the first try. For Part Two 5 correct solutions were generated, two of them on the first try. In total 40 solutions were generated, 24 for Part One and 15 for Part Two. 38 of them could be compiled and run.

5.4 ChatGPT-3.5

ChatGPT managed to generate code for all problems. All solutions were valid, and no errors were pointed out by the IDE, but two of the solutions had some warnings. ChatGPT managed to generate the correct solution for four problems in Part One and one in Part Two. Most of the generated solutions returned the wrong answer even after five tries. The correct solutions were on the first try except for the first problem where there was a ValueError that ChatGPT-3.5 could solve. 40 solutions were generated, 25 "Part One" and 15 "Part Two". ChatGPT only solved four of the "Part One"-problems therefore only four "Part Two"-problems were tested.

5.5 ChatGPT-4

ChatGPT-4 also managed to generate code for all problems. ChatGPT-4 also analyzed the generated code in case of errors. If an error existed it tried to re-generate until no errors were found. Some Run-time errors still existed in some cases.

40 solutions were generated, 25 ”Part One” and 15 ”Part Two”. All of the solutions were valid because no errors were found by the IDE.

11 of the ”Part One”-problems could be solved by ChatGPT-4, seven of them on the first try. For the Part Two problems, 5 could be solved and four of them on the first try. The tool failed to generate correct answers to 24 of the 40 problems. In 11 of those problems, the solution still resulted in the wrong answer after 5 tries. The remaining nine problems either had errors that the tool could not fix on its own or the solution used to many resources for the solution to even terminate.

5.6 Deepseek

Deepseek managed to generate valid code for all the problems. For Part One, six solutions were correct and three of them were generated on the first try. For Part Two only one solution was correct. For the incorrect answers, most of the solutions led to the wrong answer even after 5 tries. In some cases, similar to the other tools, the code did not terminate. In total Deepseek solved 7 of 40 problems. No errors were pointed out by the IDE which means all solutions were valid.

5.7 Le Chat

Of the 40 problems tested, Mistral’s chatbot, ”Le Chat”, generated 37 valid solutions in total. Nine were correct and six of them on the first try. Considering only Part One, ”Le Chat” generated 23 valid solutions where seven were correct. Four of these correct solutions were correct on the first try. For Part Two 14 solutions were valid and two were correct, both on the first try. 18 shows that the two last tries resulted in ULE, UnboundLocalError. These errors occur when a local variable is referenced before assignment.

5.8 Gemini

Gemini generated 38 solutions and 35 of them were valid. On day 2 Part Two and day 19 Part Two, Gemini failed to generate a complete solution. Out of these 38 solutions, three were correct and one was correct on the first try. Considering only the 25 problems from Part One, 23 of the solutions were valid and three were correct. For Part Two 14 solutions were generated of which 13 were valid but none were correct. On day 23, Gemini got UnboundLocalError on all tries shown as ULE in 24. These errors occur when a local variable is referenced before assignment.

Gemini Advanced generated valid code in 37 of 40 solutions. Of these 37 solutions, 4 were correct and all 4 were correct on the first try. For Part One 22 of 25 solutions were valid and 4 were correct and for Part Two all 15 were valid but none were correct.

5.9 Opus

Claude 3 Opus managed to generate valid code for 37 of the 40 problems tested. In total 13 correct solutions were generated and nine were correct on the first try. 23 of 25 solutions were valid for Part One and nine correct solutions were generated, seven of them on the first try. For Part Two 14 solutions were valid and four were correct, two of them on the first try.

5.10 Reliability & Maintainability

Running the code from all solutions through SonarQube resulted in 223 issues with a technical debt of 4 days and 7 hours. Of these 223 issues, 14 were related to reliability and 209 to maintainability, and table 42 shows the distribution of issues on respective tool. The reliability issues are considered bugs while the maintainability issues are code smells. Technical debt and code smells are defined in section 2.9.4.

SonarQube also points out the severity of the issues which are ranked as High, Medium, and Low. Of the 209 maintainability issues, 96 were ranked as high, 60 as medium, and 53 as low. Of the reliability issues 11 were ranked as high and 3 as medium.

The Reliability of the code received rating E and Maintainability rating A. The reliability rating is E because of at least one issue with high impact.

Each issue is also categorized with a Clean Code Attribute[40] Table 43 shows all 223

[40]https://docs.sonarsource.com/sonarqube/10.2/user-guide/clean-code/

Type	CoPilot	Gemini	GeminiADV	LeChat	Opus	GPT-4	GPT-3.5	DeepSeek	**Total:**
Reliability	1	2	9	1	0	1	0	0	14
Maintainability	22	28	46	23	29	16	23	22	209

Table 42 Software quality issues from respective LLM. Total is the sum of each issue type.

issues related to clean code divided into adaptability, consistency, and intentionality. Gemini Advanced had the highest number of issues overall, particularly in terms of adaptability and intentionality. Figure 6 shows an example of an adaptability issue and figure 7 shows an intentionality issue.

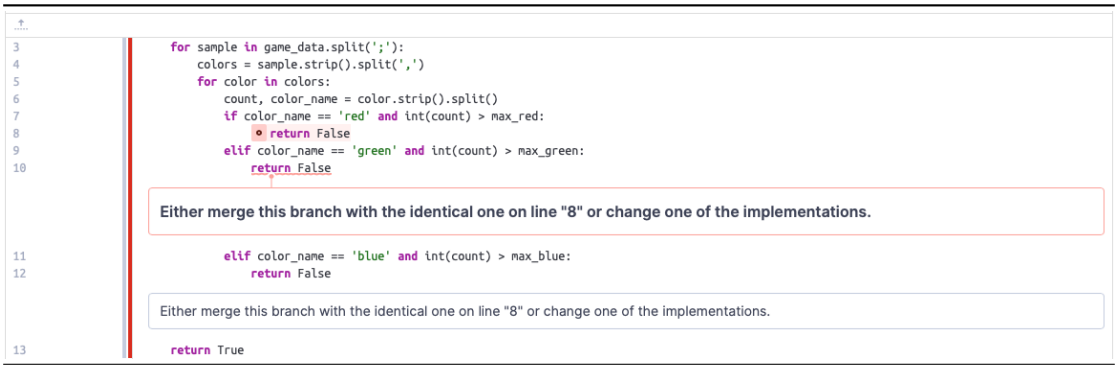

Figure 6 Adaptability issue introduced by Gemini Advanced, Day 2 Part One

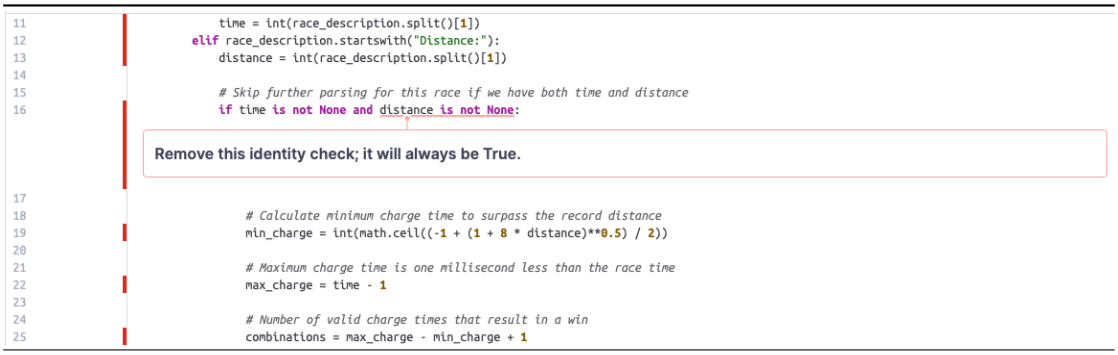

Figure 7 Intentionality issue introduced by Gemini Advanced, Day 6 Part Two

The most present issues found by SonarQube were the adaptability issues and most of these are due to the high cognitive complexity. The highest acceptable number of cognitive complexity is 15, according to SonarQube.

The consistency issues were all due to not following the naming convention. SonarQube marked those issues as code smells.

Type	CoPilot	Gemini	GADV	LeChat	Opus	GPT-4	GPT-3.5	DeepSeek	**Total:**
Adaptability	7	13	21	18	10	12	13	15	109
Consistency	2	0	2	0	8	0	4	0	16
Intentionality	14	17	32	6	11	5	6	7	98

Table 43 Clean code issues pointed out by sonarsource

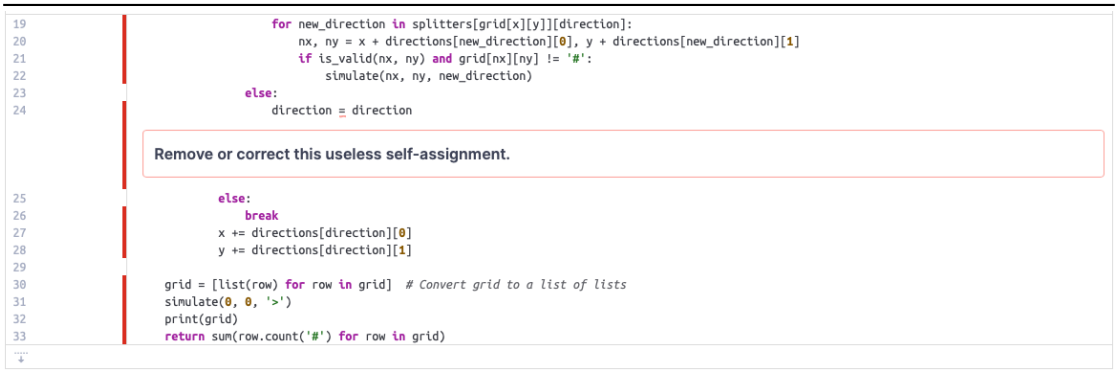

Figure 8 Bug found in code generated by CoPilot

Figure 9 Bug in code by ChatGPT-4, Day 13 Part One

From the 223 issues found in the code, 14 were labeled as bugs. These 14 bugs were found in 8 different solutions. All these bugs appeared on faulty code where no correct solutions were given. SonarQube explains the bug and how to fix it. Figure 9 shows a bug in the code from ChatGPT-4 trying to solve Day 13 Part One.

Figure 8 shows a bug introduced by CoPilot. This bug is found in the else statement where a self-assignment occurs.

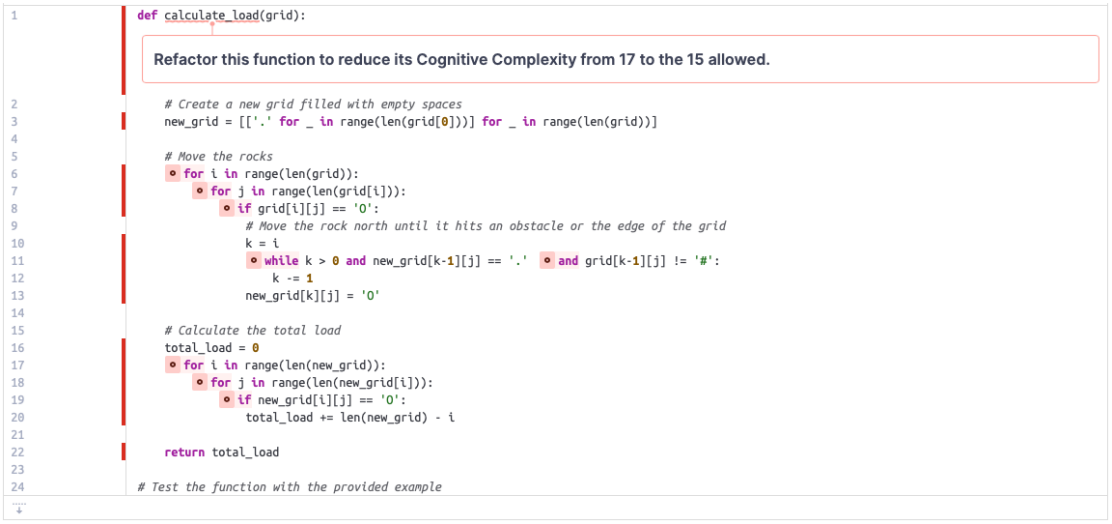

Figure 10 Cognitive complexity-Issues from CoPilot, Day 14 Part One

In figure 10, a cognitive complexity issue is shown in code generated by CoPilot. The figure shows all the parts of the code where the cognitive complexity increases which results in a total sum of 17. SonarQube also provides information on how to reduce cognitive complexity by refactoring.

6 Discussion

After testing these LLMs on Advent of Code it is clear that some tools showed better ability in understanding the problem descriptions as well as generating code. The initial problem for the tools was to process the problem descriptions for each day. When it comes to generating code, all the tools evaluated in this thesis were somewhat equal since code could be generated in almost all cases. Since some tools performed better in this study, additional research and comparisons should be conducted before concluding which tool is best. Among the tools tested here, ChatGPT-4 and Copilot performed best at generating correct solutions for Advent of Code. The other tools might be more beneficial in different settings and for solving other types of problems.

In this study, some LLMs are specifically developed for developers, while others are general chatbots that can solve problems across multiple topics. Among the LLMs tested, Deepseek Coder and GitHub Copilot are both designed to generate code and assist developers. Of these two, it is clear that Copilot is more favorable. The other LLMs have wider areas of use, which might explain why some performed poorly in

this study. However, ChatGPT-4, Claude 3 Opus, and Le Chat all performed better than Deepseek, indicating that these general-purpose tools can be used instead of those specifically developed for developers.

Today, a common way of recruiting developers is by having them solve coding tests. These tests are often fairly simple programming problems designed to assess if the developer is suitable for the position. If tools like the ones tested in this study are able to solve these kinds of programming problems, the recruitment process may need to be reconsidered. Allowing LLMs to solve these coding tests could enable developers with less knowledge to secure jobs, potentially lowering the quality of the workplace. In this study, the tools performing best can solve simpler problems but struggle with more complex ones. If the continued development of these tools leads to more capable LLMs that can solve more complex problems, rethinking the development process could be beneficial.

If tools like the ones tested in this study can be used to solve more complex problems, the need for human programmers may decrease. However, using LLMs as pair programming tools can be beneficial for developers. These tools can efficiently write code and assist with simple and repetitive tasks, making them useful in the development process. The results of this study show that while the tested LLMs are not fully capable of solving Advent of Code challenges, they can still be helpful for developers.

One potential benefit of developers using these chatbots is that the feedback given can be more specific, which can help generate correct code. If developers provide feedback on how to improve the solution, the performance of the tools could improve. In cases where the chatbots are not able to provide a correct solution, more specific feedback can help the tool generate accurate code, demonstrating where human developers can be particularly useful.

7 Conclusion

The main goal of this thesis was to determine if LLMs can solve the complex programming problems from Advent of Code. The results clearly show that the tools struggle with solving the majority of the problems, though some LLMs perform better than others.

Copilot and ChatGPT-4 performed the best in terms of correctness, but ChatGPT-4 had more correct solutions on the first try, making it the better performer. ChatGPT-4 also was the best tool for generating valid code as most failing solutions were due to the solution giving the wrong answer rather than errors. Claude 3 Opus also showed good

results and can be considered the third best LLM in this study.

Since most tools performed better on the first 9 problems of Part One, it is clear that these problems can be considered easier and that the difficulty level of Advent of Code increases thereafter.

Of all the LLMs tested, Gemini and Gemini Advanced were the least successful in solving Advent of Code. Neither of them could generate a correct solution to any of the Part Two problems, and they also produced the least number of correct solutions for Part One.

The reliability issues found were all bugs and all these issues were found in code which did not generate a correct answer. These bugs were found in 8 solutions but there are far more than 8 incorrect solutions. The 209 maintainability issues show that these LLMs are not generating maintainable code. Even though SonarQube can be used to detect issues there are still issues in the code which were not detected.

Although ChatGPT-4 and Copilot performed the best among the tools, they still fall short of completing Advent of Code. While using LLMs to generate code enhances productivity, they encounter challenges in solving complex problems similar to those in Advent of Code. Nonetheless, given their proficiency in code generation, I believe that integrating them as pair programming tools could effectively incorporate these tools into the development process.

8 Future Work

To further develop this study there are multiple things that could be done. Including testing these LLMs on other problems to perhaps receive a wider understanding in their respective strengths.

In this thesis, we tested LLMs to see how well they could handle the challenges of Advent of Code 2023. One interesting aspect is how the competition has evolved since its debut in 2015. As LLM technology has advanced, more people have tried using similar tools to tackle the competition. How has Advent of Code changed over time in terms of solving problems with these tools, and are the challenges from previous years easier to overcome with LLM assistance?

As the LLM market evolves, it would be interesting to assess the performance of new models and tools in solving Advent of Code 2023, as demonstrated in this study. Given the ongoing evolution of LLMs, it would be compelling to observe how future iterations handle the challenges presented by Advent of Code 2023.

This thesis investigates how LLMs perform when generating code in Python. To expand this analysis, it would be valuable to examine how the choice of programming language affects the results. This could involve investigating how the solutions differ based on the programming language and identifying which LLMs perform better with specific languages

References

[1] C. F. Atkinson, "Chatgpt and computational-based research: benefits, drawbacks, and machine learning applications," *Discover Artificial Intelligence*, vol. 3, no. 1, pp. 42–18, 2023.

[2] K. Beck, M. Fowler, and G. Beck, "Bad smells in code," *Refactoring: Improving the design of existing code*, vol. 1, no. 1999, pp. 75–88, 1999.

[3] M.-H. Chen, M. Lyu, and W. Wong, "Effect of code coverage on software reliability measurement," *IEEE Transactions on Reliability*, vol. 50, no. 2, pp. 165–170, June 2001.

[4] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica, "Chatbot arena: An open platform for evaluating llms by human preference," 2024.

[5] T. Dohmke, "Github copilot for business is now available," 2023, last accessed 7 February 2024. [Online]. Tillgänglig: https://github.blog/2023-02-14-github-copilot-for-business-is-now-available/

[6] A. Drapkin, "Chatgpt vs claude 3 test: Can anthropic beat openai's superstar?" March 2024, last accessed 10 April 2024. [Online]. Tillgänglig: https://tech.co/news/chatgpt-vs-claude-3#:~:text=According%20to%20Google%2Dbacked%20Anthropic,written%20and%20easier%20to%20read.

[7] N. A. Ernst and G. Bavota, "Ai-driven development is here: Should you worry?" *IEEE Software*, vol. 39, no. 2, pp. 106–110, 2022.

[8] Y. Feng, S. Vanam, M. Cherukupally, W. Zheng, M. Qiu, and H. Chen, "Investigating code generation performance of chat-gpt with crowdsourcing social data," in *Proceedings of the 47th IEEE Computer Software and Applications Conference*, 2023, pp. 1–10.

[9] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. K. Li, F. Luo, Y. Xiong, and W. Liang, "Deepseek-coder: When the large language model meets programming – the rise of code intelligence," 2024.

[10] W. Harding and M. Kloster, "Coding on copilot," 2024, last accessed 7 February 2024. [Online]. Tillgänglig: https://gitclear-public.s3.us-west-2.amazonaws.com/Coding-on-Copilot-2024-Developer-Research.pdf

[11] M. Kazemitabaar, X. Hou, A. Henley, B. J. Ericson, D. Weintrop, and T. Grossman, "How novices use llm-based code generators to solve cs1 coding tasks in a self-paced learning environment," in *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '23. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Tillgänglig: https://doi.org/10.1145/3631802.3631806

[12] H. Keuning, B. Heeren, and J. Jeuring, "Code quality issues in student programs," in *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 110–115. [Online]. Tillgänglig: https://doi.org/10.1145/3059009.3059061

[13] M. Maher, Y. Tadimalla, and D. Dhamani, "Is chatgpt good for your students? a study design of the impact of ai tools on the student experience in learning java," in *EDULEARN23 Proceedings*, ser. 15th International Conference on Education and New Learning Technologies. IATED, 3-5 July, 2023 2023, pp. 5702–5709. [Online]. Tillgänglig: https://doi.org/10.21125/edulearn.2023.1493

[14] B. Meyer, *Object-oriented software construction*. Prentice hall Englewood Cliffs, 1997, vol. 2.

[15] O. Mortensen, "How many users does chatgpt have? statistics & facts (2024)," 2024, last accessed 7 May 2024. [Online]. Tillgänglig: https://seo.ai/blog/how-many-users-does-chatgpt-have

[16] N. Nguyen and S. Nadi, "An empirical evaluation of github copilot's code suggestions," in *Proceedings of the 19th International Conference on Mining Software Repositories*, ser. MSR '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–5. [Online]. Tillgänglig: https://doi-org.ezproxy.its.uu.se/10.1145/3524842.3528470

[17] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The impact of ai on developer productivity: Evidence from github copilot," 2023.

[18] J. M. Perkel, "Six tips for better coding with chatgpt," *Nature (London)*, vol. 618, no. 7964, pp. 422–423, 2023.

[19] A. Rowe, "What is mistral ai? le chat and mistral large explained," March 2024, last accessed 10 April 2024. [Online]. Tillgänglig: https://tech.co/news/what-is-mistral-ai-le-chat

[20] H. R. Saeidnia, "Welcome to the gemini era: Google deepmind and the information industry," *Library Hi Tech News*, no. ahead-of-print, 2023.

[21] K. Srinath, "Python–the fastest growing programming language," *International Research Journal of Engineering and Technology*, vol. 4, no. 12, pp. 354–357, 2017.

[22] G. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for software design smells: managing technical debt*. Morgan Kaufmann, 2014.

[23] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[24] I. E. Team, "Code quality: What it is and how to measure it (with tips)," 2023, last accessed 7 May 2024. [Online]. Tillgänglig: https://www.indeed.com/career-advice/career-development/what-is-code-quality

[25] N. Thaker and A. Shukla, "Python as multi paradigm programming language," *International Journal of Computer Applications*, vol. 177, no. 31, pp. 38–42, 2020.

[26] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Tillgänglig: https://doi.org/10.1145/3491101.3519665

[27] A. Wood, "Predicting software reliability," *Computer*, vol. 29, no. 11, pp. 69–77, Nov 1996.

[28] B. Yetistiren, I. Ozsoy, and E. Tuzun, "Assessing the quality of github copilot's code generation," in *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 62–71. [Online]. Tillgänglig: https://doi.org/10.1145/3558489.3559072

[29] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, "Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt," 2023.