The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)
ISSN: 2602-3199

The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2024

Volume 28, Pages 326-341

ICBASET 2024: International Conference on Basic Sciences, Engineering and Technology

Performance Comparison of AI Platforms in Solving Computer Science Problems

Huseyin Karacali
TTTech Auto Turkey

Efecan Cebel
TTTech Auto Turkey

Nevzat Donum
TTTech Auto Turkey

Abstract: In the rapidly evolving landscape of artificial intelligence (AI), its significance and accelerated development are undeniable. AI has emerged as a cornerstone technology with profound implications across various domains, driving innovation and reshaping the way we approach complex problems. Particularly, the utilization of AI in coding tasks has garnered substantial attention, given its potential to streamline development processes and enhance the efficiency of software engineering practices. Against this backdrop, this paper presents a detailed comparative analysis of four different AI platforms, namely ChatGPT, Gemini, Blackbox, and Microsoft Copilot, in addressing key challenges within the realm of computer science, spanning natural language processing, image processing, and cybersecurity. The study focuses on leveraging the C++ programming language to develop solutions for these multifaceted problems across the aforementioned platforms. Each platform's outputs are meticulously evaluated on various parameters including accuracy, execution time, code size, and time complexity to provide a comprehensive understanding of their performance. Furthermore, an iterative optimization methodology is employed, entailing three rounds of refinement for the code produced by each platform, with the resultant outputs subjected to comparative analysis in each iteration. Through this rigorous approach, the paper not only elucidates the efficacy of different AI platforms in addressing diverse computational challenges but also underscores the iterative enhancement process on AI platforms for refining code quality and performance across multiple domains within computer science.

Keywords: AI-driven development, AI-driven chatbot, AI Platforms, Computer science

Introduction

In recent years, the integration of AI techniques has become increasingly pervasive across numerous domains, revolutionizing the way complex challenges in computer science are approached, natural language processing (NLP), image processing, and cybersecurity. Amidst this landscape, the emergence of sophisticated AI platforms such as ChatGPT, Gemini, Blackbox, and Microsoft Copilot has offered promising avenues for addressing multifaceted problems through innovative methodologies and advanced algorithms.

This study endeavors to undertake a meticulous comparative analysis of these four prominent AI platforms, with a specific focus on their efficacy in tackling fundamental challenges spanning the aforementioned domains. By delving into the intricate nuances of each platform's capabilities and performance characteristics, developers aim to provide valuable insights into their applicability and suitability for addressing real-world problems across diverse contexts.

Central to this comparative analysis is the adoption of the C++ programming language as a unifying framework for solution development. Recognized for its efficiency, versatility, and widespread usage in various domains, C++ serves as an ideal platform for evaluating and benchmarking the performance of AI algorithms and methodologies across different AI platforms.

The evaluation criteria employed in this study encompass a comprehensive array of parameters, including but not limited to accuracy, execution time, code size, and time complexity. Through a rigorous examination of these metrics, study aims to elucidate the strengths and limitations of each AI platform in addressing complex challenges, thereby facilitating informed decision-making regarding their deployment in practical settings.

Furthermore, to ensure a thorough understanding of each platform's performance, a three-round iterative optimization methodology is employed. This iterative approach necessitates successive refinement of the outputs generated by each platform, followed by comparative analysis in each iteration. By subjecting the refined outputs to systematic evaluation, the goal is to capture the iterative evolution of solutions produced by these AI platforms and assess their adaptability and robustness in response to optimization efforts.

Ultimately, the insights derived from this comparative analysis are expected to enrich our understanding of the capabilities and limitations of AI platforms in addressing multifaceted challenges across diverse domains. By shedding light on the relative strengths and weaknesses of each platform, this study aims to empower researchers, practitioners, and decision-makers with valuable knowledge for leveraging AI-driven solutions effectively in real-world scenarios.

Since the day AI chat boxes started to be released, comparisons have begun on many issues such as which one has better understanding ability, which one gives more relevant answers, which one is more up-to-date. Each released AI chat box has been directly compared with others. A comparison of the most well-known of these, 'Bard', now known as 'Gemini', and ChatGPT was made by Waisberg and his colleagues. In this study, a one-to-one comparison was made and the comparison was detailed on a topic in the field of medicine (Waisberg et al., 2023).

These developed AI platforms were used for purposes such as generating code and even completing some projects without any human development. It has even been seen as a threat to the profession of computer programmers. In an article written by Brett A. Becker and his colleagues, the educational opportunities, and difficulties in producing code with AI are mentioned (Becker et al., 2023). In short, researchers have always observed the situation of coding with artificial intelligence and continued their studies.

Finally, there are studies similar to this study that make calculations and comparisons on code quality. One of these is the code quality calculation in artificial intelligence-assisted code development made by Yetiştiren and his friends. In this study, experimental measurements were made. Copilot, CodeWhisperer, and ChatGPT were used (Yetistiren et al., 2023).

With this effort, a study has been produced that will contribute to the advancement of artificial intelligence research and applications, encouraging innovation and informed decision-making in the development and application of AI technologies.

Materials

ChatGPT

ChatGPT, developed by OpenAI, is an advanced natural language processing model designed to generate human-like text responses based on input prompts. It is built upon the GPT (Generative Pre-trained Transformer) architecture, specifically the GPT-3.5 variant, which utilizes deep learning techniques to understand and generate text (OpenAI, 2024).

Unlike traditional chatbots that rely on predefined responses or rule-based systems, ChatGPT employs a machine learning approach called unsupervised learning. This means it learns from vast amounts of text data from the internet without explicit human supervision. As a result, it can produce contextually relevant and coherent responses across a wide range of topics.

One of the key features of ChatGPT is its ability to understand and generate text in multiple languages, making it accessible to diverse linguistic communities worldwide. Additionally, it can mimic the style and tone of input prompts, allowing for more personalized interactions.

ChatGPT finds applications in various fields such as customer service, content generation, language translation, and educational tools. Its versatility and adaptability make it a valuable asset for businesses, researchers, and developers seeking to leverage natural language processing capabilities. Moreover, ChatGPT has undergone continuous refinement and improvement through iterations, enhancing its performance and capabilities over time. OpenAI regularly updates and fine-tunes the model to ensure its effectiveness and reliability in various contexts.

Figure 1. The logo of the chatGPT (ChatGPT, n.d.)

In summary, ChatGPT represents a significant advancement in natural language processing technology, offering a powerful tool for generating human-like text and facilitating seamless interactions between humans and machines (ChatGPT, n.d.). In this study, ChatGPT 3.5, recognized as one of the most utilized AI platforms, has been chosen as one of the subjects for evaluation.

Gemini

Gemini, developed by Google AI, is a groundbreaking innovation in the world of AI. Described as a great language model, Gemini has many capabilities such as creating text, translating languages, producing creative content, and answering your questions in an informative way.

Figure 2. The logo of the gemini (Gemini, n.d.)

Although still in development, Gemini has learned to perform many types of tasks. For example, it can create different text formats such as codes, scripts, musical pieces, emails, and letters. It also translates languages, facilitating communication between people speaking different languages.

One of Gemini's most important characteristics is that he can answer your questions comprehensively and informatively, even if they are open-ended, challenging, or strange. Thanks to this feature, its usability increases in various fields such as research, education, and entertainment (Gemini", 2023). In this study, Gemini (formerly Bard), is another selected AI platform for comparison. Since Gemini is developed by Google, it is trusted for these kinds of studies.

Copilot

Microsoft Copilot stands at the forefront of this transformation, offering a tool with the potential to revolutionize the coding process. Leveraging a large language model and diverse coding techniques, Copilot is designed to assist developers in their coding endeavors. By providing code suggestions, identifying errors, and optimizing code, it empowers programmers to write code faster and more efficiently ("Microsoft Copilot," n.d.).

Figure 3. The logo of the microsoft copilot ("Microsoft Copilot," n.d.)

Basic properties include code suggestions, debugging assistance, code optimization, code generation, and multi-language support. Copilot holds immense potential to serve as a valuable asset for developers, potentially enhancing the coding process in various aspects. However, it is crucial to recognize Copilot as a tool, not a replacement for human expertise. Responsible usage of Copilot and maintaining complete control over the code necessitate continued manual code review and editing. Microsoft Copilot exemplifies the impact of AI in the software development processes. As technology advances, such tools are poised to evolve further, empowering developers to become more productive and creative ("Microsoft Copilot," n.d.). In this study, Copilot was chosen as one of the AI platforms to be compared. Microsoft Copilot will be a high-potential choice for this study, which will involve calculations and developments related to computer science.

Blackbox

Blackbox AI is another platform in the realm of AI-powered coding tools. Designed to assist developers in writing, debugging, and optimizing their code, Blackbox AI leverages a large language model and various coding techniques to provide real-time suggestions and support. The tool's ability to enhance developer productivity, improve code quality, and facilitate learning has propelled it to popularity. However, for effective and ethical coding practices, it's crucial to utilize Blackbox AI responsibly and acknowledge its limitations.

Figure 4. The logo of the blackbox ai (Tech Insider Buzz, 2023)

Blackbox AI boasts a range of functionalities that empower developers. One key feature is its ability to translate natural language descriptions into code, making it easier for beginners or those unfamiliar with a specific language to get started. The tool also provides real-time insights on current events, technological advancements, and product launches, keeping developers updated and informed. The most impactful feature for speeding up development is Blackbox AI's code completion. By automatically suggesting the next lines of code based on the coding context, it helps developers maintain their coding flow and minimize errors. Blackbox AI even caters to non-coders to a certain extent. Users can generate code from visual inputs like code screenshots or product sketches, enabling the creation of simple code ("BlackboxAI," n.d.). Since the Blackbox AI is becoming increasingly popular for use in software development processes, it is the last selected AI chat box for the study.

Development and Testing Platform

The execution and compilation of each code response given iteratively by the AI chat boxes was done on a single computer to ensure consistent and accurate comparison. This computer is a Linux-based virtual machine with Ubuntu-20.04.1 operating system with 5.15.0-101-generic kernel. It has 64-bit architecture with x86_64. All solutions by AI platforms are produced in C++ programming language and these codes are compiled with g++. The version of g++ is also 9.4.0.

Method

In this study, the extent to which 4 different AI platforms can successfully write a program under the desired conditions is compared with each other. Since these comparisons will be based on programming ability, performance, artificial intelligence's ability to solve problems that may arise in the program, and performance improvement capabilities, experiments have been made to find solutions to certain predetermined computer science problems. These computer science problem types belong to the fields of natural language processing (NLP), cybersecurity (CS), and image processing (IP), respectively.

These topics are very popular today and are actively developing areas. The solutions that the mentioned AI platforms would provide to the problems identified in these areas were measured. Evaluations were made in line with some generally determined metrics such as accuracy, execution time, executable size, and time complexity. In this study, it is not enough to see how well the artificial intelligence platforms solve the problems in one go, it is requested to interact 3 more times after the first response and to correct any errors if there is an error, or to improve the application in terms of performance.

Selected Computer Science Topics

Natural Language Processing

Natural language processing is the first of the selected topics. Under this topic, a problem that examines the frequency of letters in the text message entered by the user and gives the output in order from the most frequently used to the least frequently used, was requested by artificial intelligence platforms. It is planned that the answers and improvements to be given by artificial intelligence may be effective in realizing this problem, due to the diversity of the processing of characters of the input data, the method of counting letters, and the ways of storing this data at run-time. In addition, it can be easily observed in this study whether it works consistently or not. Apart from consistency, the main purpose is to observe the change in time complexity since there are many different methods to perform this operation. The input text specified by the user can be of any length and content. The program to be developed must be able to handle this situation and is expected to respond correctly.

Cybersecurity

Another selected topic is cybersecurity. The problem under this topic is that the text data in 'test.txt' given as input is encrypted using the AES-128 algorithm and this encrypted data is written in a file called 'encrypted.txt'. The key and initialization vector to be used for this process are predefined. In general, since each character will be counted one by one in the program, much variation in terms of time complexity is not expected. However, it is a topic that can be a reference in comparison in terms of accuracy, and executable size.

Image Processing

The last computer science topic chosen is image processing. The problem under this topic is that edge detection can be made in an example image named 'example.jpg'. Since it is a more demanding task in terms of performance compared to the first two topics, it is suitable for use in run-time comparison.

Figure 5. Example image for image processing problem (Peter Kovesi, n.d.)

As a solution to this problem, after successful edge detection, a file named 'result.jpg' should be produced as output. This output should be an output indicating the edges of the objects in the first image. In addition, the program is responsible for checking the existence of the image to be given as input.

Measurement Metrics

Another part of carrying out the study is evaluation and measurement. Separate answers for each AI platform for all problems in terms of accuracy, time, size, and time complexity were observed and recorded in each iteration. A total of 192 different values were examined under 3 different topics, with 4 different AI platforms, when responses were received 4 times and observed with 4 different metrics.

Correctness and Accuracy

The first metric observed during the evaluation process of the study is whether it works correctly or not. Since a software solution is offered to the problems, results such as compilation errors, faulty operation, inconsistent operation, and correct operation can be seen under this metric. Although it works correctly, especially for performance in image processing, the output quality varies. The problems encountered for the faulty situations under this heading are also stated in the results section.

Execution Time

Another measurement value is execution time. For all iterations that did not have a compilation error (an executable file was produced), the time from the moment it was run from the command line to its termination was recorded. This represents how effectively a solution is offered in terms of performance. The 'time' command, which is built-in in Linux, was used to measure this value. The command executed as 'time ./executable_name' both runs the application offered by the artificial intelligence and prints the elapsed time on the terminal screen.

Many parameters are involved in measuring this value. Parameters such as I/O operations, operations of other applications running in the background, and input delays can change this time. Therefore, a script was prepared that records the resulting time using the 'time' command and repeats this process 100 times and averages the

resulting time values. Thanks to this script, each iteration will be run 100 times and the average value will be determined as the execution time. Thus, the deviation of other processes on this measurement is greatly reduced.

Executable Size

Another observation value is the size of this executable file (the executable file resulting from compilation) in bytes. The size of the outputs is of great importance, especially in sectors where there are many memory-related limitations, such as embedded systems. Therefore, it is quite significant that the concept of size can be included in performance comparison. For this value, the size of each compiled output was observed and recorded. All codes produced by AI were compiled with g++ and the '-O0' flag, which is the 'no optimization' option, was used. This allows it to be compiled without any optimization by the compiler. Thus, the compiler performance impact on the work is eliminated.

Time Complexity

The last value to check is time complexity. In computer science, performance independent of sub-branches is very important and this performance directly depends on the number of operations and the size of the data to be processed. Avoiding repetitive operations, especially loops, usually has positive results in terms of performance. Time complexity is one of the most important performance values since the largest software is considered. For this reason, evaluating each response given by AI in terms of time complexity is also an important task for the purpose of comparing these platforms.

Results and Discussion

The applications created with the solutions provided by AI platforms for each problem described in the Method section were noted, taking into account the above-mentioned values. The results for each problem will be discussed separately in this section. All 4 answers to a problem will be presented in the form of tables, and based on these values, platforms can be compared.

Natural Language Processing

Iteration 1

Table 1. First iteration for NLP problem solution

	1	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
NLP	ChatGPT	Compile Error	Missing library included.		
	Gemini	Compile Error	Some functions used are undefined.		
	Copilot	Compile Error	Syntax error.		
	Blackbox	Compile Error	Missing library included.		

As seen in Table 1, none of the results requested from AI for the NLP problem could produce a successful output. The project was not compiled because both ChatGPT and Blackbox did not add the 'vector' library required for the vector type they use to work. The code produced by Gemini did not have definitions for some of the functions it used, and Copilot made a very simple cursor error and gave a compilation error. Based on these results, a data set to use for comparison could not be obtained.

Iteration 2

Table 2. Second iteration for NLP problem solution

	2	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
NLP	ChatGPT	Success	3	71504	O(n*logn)
	Gemini	Success	3	73704	O(n*logk)
	Copilot	Compile Error	Syntax error.		
	Blackbox	Success	2	67488	O(n)

Since there were errors in the code produced by all AI platforms in the first iteration, each of them was notified of their errors and asked to solve the problem. As a result, Copilot could not solve the syntax error it caused, while all other platforms managed to produce the program that worked correctly.

In the second iteration, all platforms that successfully solve the problem have execution times that are very close to each other. There is an almost incomparable difference in terms of performance. In terms of size, there is a difference of approximately 4 kilobytes between Blackbox and ChatGPT, and 2 kilobytes between ChatGPT and Gemini, from least to most of each. Blackbox has achieved a better result in terms of size compared to other platforms, albeit slightly.

Additionally, the most obvious difference in this iteration emerged in the time complexity metric. It is greater than O(n) in time complexity for both ChatGPT and Gemini because there is a logarithmic expression next to n in the multiplication case. Here the letter n indicates the number of characters in the input. The letter 'k' represents repeating numbers. Although Gemini is slightly higher performing than ChatGPT, Blackbox has achieved a significantly better result compared to other platforms in the second iteration for the NLP problem.

Iteration 3

Table 3. Third iteration for NLP problem solution

	3	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
NLP	ChatGPT	Success	< 1	52560	O(n)
	Gemini	Compile Error	Wrong template usage.		
	Copilot	Compile Error	Syntax error.		
	Blackbox	Incorrect Result	1	56736	O(n)

In the third iteration, platforms that worked correctly in the previous round were asked to improve their performance, while Copilot was given a prompt to resolve the error. As a result, Copilot could not find a solution to the syntax error by giving the exact same answer. In addition, Gemini, which tried to add a template structure to the code, encountered a compilation error because it responded with the wrong use of the template expression.

Blackbox, which had the best results in terms of performance in the previous iteration, also preserved time complexity, halved the execution time, and managed to reduce the code size by approximately 11 kilobytes. However, it was observed that some letters of the alphabet could not be detected in the output of the program. No matter how many times the letters 'z' and 'a' were repeated, they could not be detected in the text given as input by the application.

Finally, ChatGPT demonstrated a very successful performance increase in this iteration. Executable file size is reduced by almost 19 kilobytes, reduced the average execution time to less than 1 millisecond, and made the time complexity linear to the input O(n).

Iteration 4

Table 4. Fourth iteration for NLP problem solution

	4	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
NLP	ChatGPT	Success	2	45120	O(n*logk)
	Gemini	Incorrect Result	4	74568	O(n)
	Copilot	Success	3	61368	O(n*logk)
	Blackbox	Success	3	61368	O(n*logk)

For the last iteration of this problem, all AI platforms provided compilable answers. Copilot produced a compilable piece of code for the first time. This piece of code, working correctly, had very average performance. It has average values in terms of code size, running time, and time complexity. Blackbox, whose application ran incorrectly in the previous iteration, produced exactly the same answer as Copilot. Therefore, all values are the same. It lost its performance in running time and optimization in time complexity in the second iteration. In summary, the desired improvement in terms of performance could not be achieved with Blackbox AI.

Gemini, on the other hand, has solved the compilation error caused by using the wrong draft in the previous round, but although the piece of code it produced can be compiled, it does not work correctly. It shows completely random values for the letters and characters used once input is given. For example, it detects that the letter 't' occurs 98 times in the input message given as 'test'.

Finally, ChatGPT has again produced a solution that works correctly. Although it lost some performance in runtime in this iteration, it performed very well in terms of code size. Compared to the first successful iteration, it showed good improvement in all values.

Cybersecurity

Iteration 1

Table 5. First iteration for CS problem solution

	1	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
CS	**ChatGPT**	Compile Error	Syntax error.		
	Gemini	Incorrect Result	5	23880	O(n)
	Copilot	Incorrect Result	3	33848	O(n)
	Blackbox	Compile Error	memset is not defined.		

In the first answers to this problem, which asked to encrypt a sample text with the AES-128 algorithm, the codes produced by ChatGPT and Blackbox caused compilation errors. While the code produced by ChatGPT gives a syntax error, the library that will define the memset function has not been added in Blackbox's. Although the codes produced by Gemini and Copilot passed the compilation phase successfully, they did not give correct results when they were run. In the output of the code produced by both of them, no new file was created for the encrypted data. Therefore, a successful coding could not be done by AI for the first round.

Iteration 2

Table 6. Second iteration for CS problem solution

	2	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
CS	**ChatGPT**	Success	2	23248	O(n)
	Gemini	Success	4	23880	O(n)
	Copilot	Incorrect Result	3	34792	O(n)
	Blackbox	Success	3	18720	O(n)

In the second round of this problem, all compilation errors encountered in the previous iteration have been resolved. At the same time, the code generated by Gemini worked correctly, but even though Copilot made some changes to the code, the file containing the encrypted data was still not created. For all AI platforms, time complexity is equal, each O(n), directly related to the length of the data to be encrypted that given in the example file. Apart from this, Blackbox has produced code with an average execution time but is much smaller in size than other AIs. There is a half-and-half execution time difference between the outputs of ChatGPT and Gemini, which were almost the same size at the time. ChatGPT has the best uptime performance.

Iteration 3

Table 7. Third iteration for CS problem solution

	3	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
CS	**ChatGPT**	Success	2ms	23240	O(n)
	Gemini	Incorrect Result	4ms	19080	O(n)
	Copilot	Incorrect Result	3ms	34792	O(n)
	Blackbox	Compile Error	Wrong function usage with too many arguments.		

Copilot reproduced exactly the same code as it produced in the previous iteration. Therefore, the incorrect output has not changed in this round either. On the other hand, the output of Gemini, which worked successfully in the previous round, works incorrectly in this round. The text file it creates to hold encrypted data is empty. It

provided a significant reduction in code size, but this disrupted the operation of the code. Also, Blackbox produced code that could not be compiled due to changes made while trying to improve performance, and a compilation error occurred. ChatGPT, the only AI platform that continues to operate successfully in this iteration, has produced an output that performs almost identically to the performance it produced in the previous iteration.

Iteration 4

Table 8. Fourth iteration for CS problem solution

	4	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
CS	ChatGPT	Compile Error	Type conversion error.		
	Gemini	Incorrect Result	4ms	23240	O(n)
	Copilot	Incorrect Result	3ms	34792	O(n)
	Blackbox	Incorrect Result	3ms	18512	O(n)

In the last round of the Cybersecurity problem, Copilot produced the same answers as in the previous 2 iterations and failed to make any changes. Even though Gemini made changes to the code, the code he produced in the previous round was working incorrectly, it still could not produce a successful result and the file it produced for the encrypted data was empty. The Blackbox output, which produced a compilation error in the third iteration, resolved this error but produced an empty file, just like Gemini. While ChatGPT was improving performance, it produced a code that made a type error in the data and caused a compilation error.

Image Processing

Iteration 1

Table 9. First iteration for IP problem solution

	1	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
IP	ChatGPT	Success	32	66496	O(n)
	Gemini	Success	31	66720	O(n)
	Copilot	Success	33	66688	O(n)
	Blackbox	Success	112	66472	O(n)

In the first answers received for the image processing problem, all generated codes were successfully compiled and edge detection was successfully performed. All time complexities are equal and O(n). Here, the letter n represents the number of pixels to be processed. Additionally, all outputs produced are almost identical in size. The single most obvious difference in the output created by AIs is Blackbox's run time. It has more than 3 times the execution time of others. Below are the edge detection outputs produced by 4 different AI.

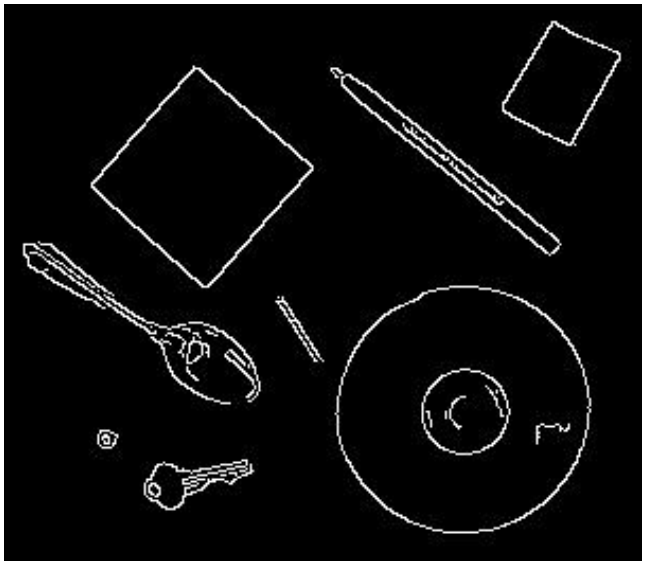

Figure 6. Output of ChatGPT for the first iteration

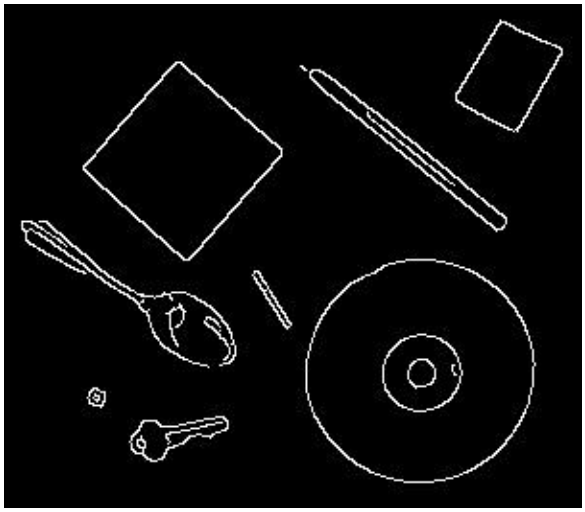

Figure 7. Output of Gemini for the first iteration

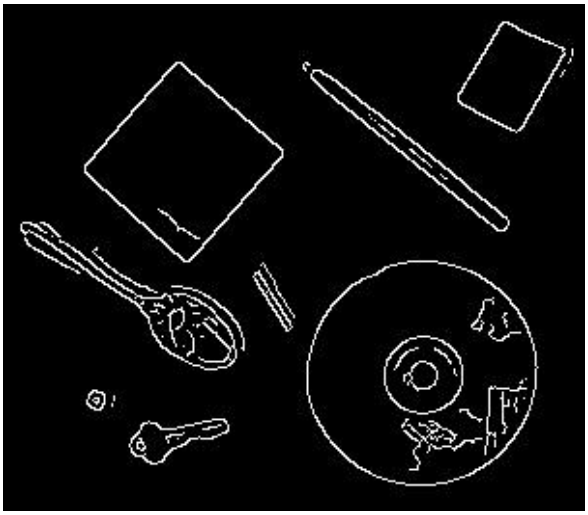

Figure 8. Output of copilot for the first iteration

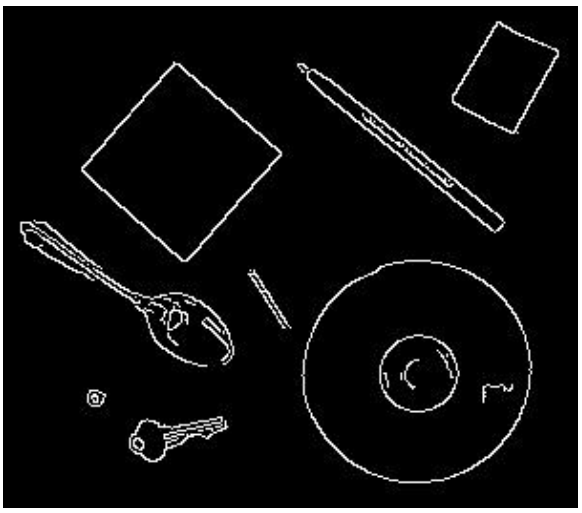

Figure 9. Output of blackbox for the first iteration

When looking at the outputs given in Figure 6, Figure 7, Figure 8, and Figure 9, it is observed that Gemini gave the best result for the first iteration. It is the one that best indicates the real edges of the shapes in the image and can distinguish the parts that are not edges. While ChatGPT and Blackbox produce similar outputs, non-edge details are shown as edges in the output of the code produced by the Copilot platform. Although all AI platforms produce very close results, Gemini has the fastest and highest quality output in the first iteration.

Iteration 2

Table 10. Second iteration for IP problem solution

	2	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
IP	ChatGPT	Compile Error	Invalid headers included.		
	Gemini	Compile Error	Undeclared built-in functions.		
	Copilot	Success	33	66688	O(n)
	Blackbox	Success	33	66320	O(n)

In the second iteration of the image processing problem, both ChatGPT and Gemini caused a compilation error due to the code fragments they added while improving performance. Therefore, they could not produce output. Since Copilot produces exactly the same code as its previous answer, it is same as the visual output in Figure 8. The output of Blackbox is also very similar with the first iteration output of itself.

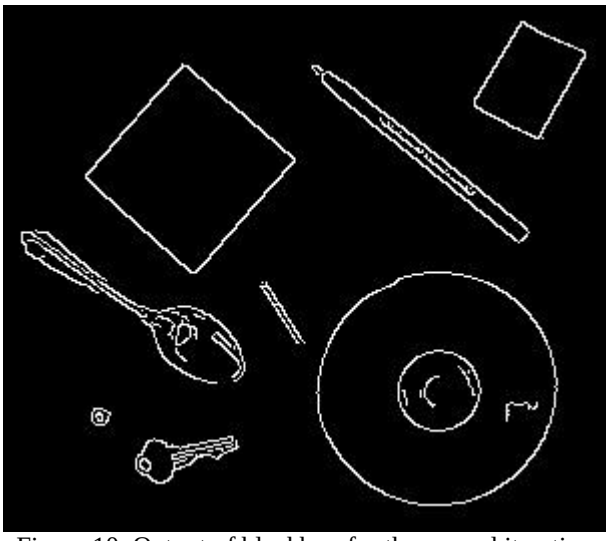

Figure 10. Output of blackbox for the second iteration

Iteration 3

Table 11. Third iteration for IP problem solution

	3	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
IP	ChatGPT	Success	34	66776	O(n)
	Gemini	Success	27	75600	O(n)
	Copilot	Success	29	66864	O(n)
	Blackbox	Success	27	66504	O(n)

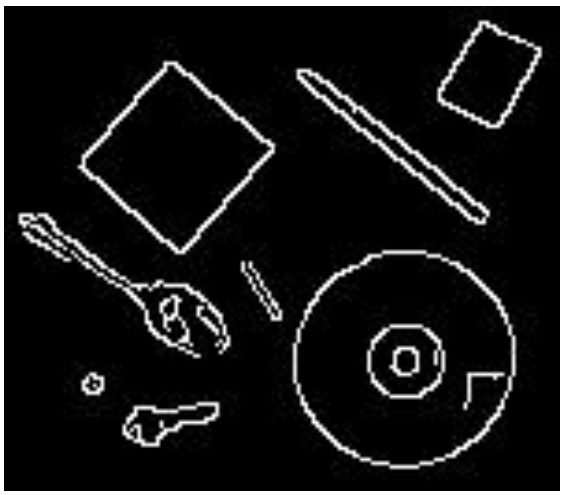

Figure 11. Output of chatGPT for the third iteration

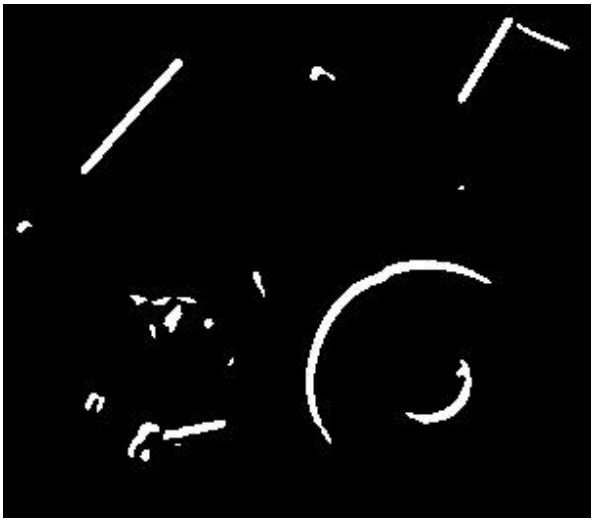

Figure 12. Output of gemini for the third iteration

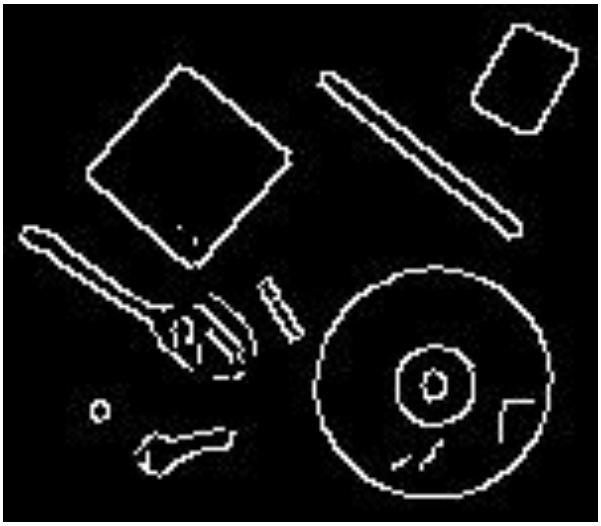

Figure 13. Output of copilot for the third iteration

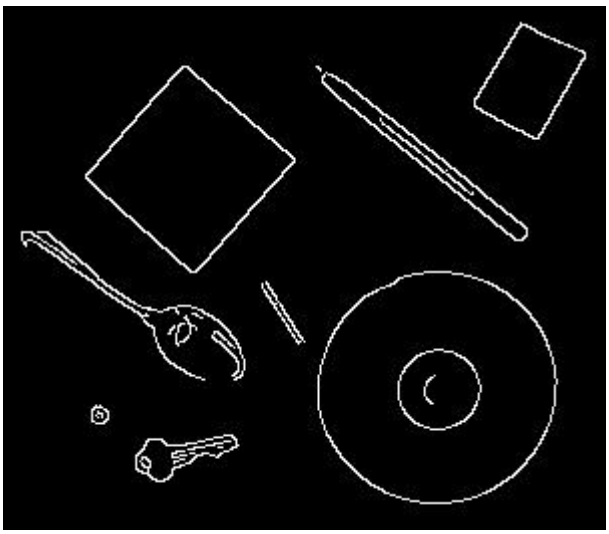

Figure 14. Output of blackbox for the third iteration

In the third round of this problem, both ChatGPT and Gemini managed to resolve the compilation errors from the previous round. The edge detection output produced by ChatGPT, which produces an output with similar values to the first round, is given in Figure 11. Gemini, on the other hand, shortened the execution time

considerably but created a significant growth in size. The output produced is given in Figure 12. Both the shortest execution time and the smallest size were produced by Blackbox and their output is shown in Figure 14. Copilot managed to change the code in this iteration and detected an edge as in Figure 13. According to the outputs given in Figure 11, Figure 12, Figure 13, and Figure 14, ChatGPT and Copilot outputs were reduced in size. Blackbox has developed a very successful edge detection algorithm. ChatGPT and Copilot produced an average level of output, but in the application produced with the code answered by Gemini, the edge detection algorithm worked very poorly and no object could be distinguished.

Iteration 4

Table 12. Fourth iteration for IP problem solution

	4	Correctness	Execution Time (ms)	Code Size (bytes)	Time Complexity
IP	ChatGPT	Success	27	121152	O(n)
	Gemini	Compile Error	Undeclared built-in functions & variables.		
	Copilot	Success	29	66864	O(n)
	Blackbox	Incorrect Result	Segmentation fault.		

Gemini, which produced a very bad output in the previous iteration, used built-in functions in this last round, but these caused a compilation error. Blackbox, which produced a very good output in the previous round, produces an output that can be compiled, but this output cannot detect an edge due to the segmentation fault.

On the other hand, ChatGPT and Copilot have managed to write codes that produce an output. However, since Copilot produced the exact same code as in the previous iteration, the output in Figure 13 was also generated in the last round. The code in ChatGPT's response has almost doubled in size compared to its last version and produces a result like Figure 15.

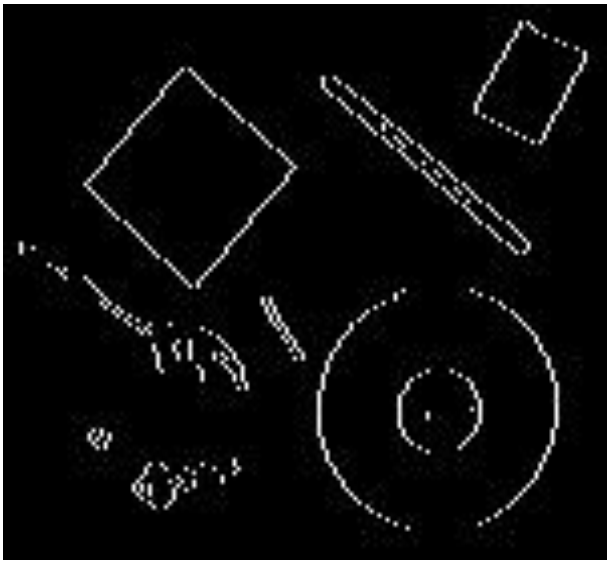

Figure 15. Output of ChatGPT for the fourth iteration

As seen in the ChatGPT output shown in Figure 15, the edge detection algorithm has a very poor performance. The edges of some objects could be transferred as cut-off, some objects became unidentifiable. Therefore, performance improvement efforts did not yield positive results.

Conclusion

In conclusion, this study has undertaken a comprehensive comparative analysis of four prominent AI platforms—ChatGPT, Gemini, Microsoft Copilot, and Blackbox—within the context of addressing key challenges in computer science, encompassing natural language processing (NLP), image processing, and cybersecurity. Through meticulous evaluation and benchmarking, this research aimed to provide valuable insights into the performance, capabilities, and limitations of each platform in generating solutions for multifaceted problems.

To summarize the results of the platforms, Copilot shows the poorest performance as a code developer. There have been many cases where he repeated the previous answer multiple times when he encountered an error or when performance improvement was requested. In this case, if there is an error, it can complete 4 iterations without any solution, just like in cybersecurity. Blackbox AI, on the other hand, can analyze and solve compilation errors caused by it very well. In no case has it produced code with compilation errors twice in a row and it is the best of all in terms of time complexity. It demonstrated a performance similar to Gemini in terms of code generation capability. Although Gemini, like Blackbox, successfully produces solutions to compilation errors, one of Gemini's shortcomings is that the compiled projects do not give correct outputs. Executable also produces more inefficient outputs than other platforms in terms of size. Finally, ChatGPT gave the best results among these AI platforms. What distinguishes it from others is that it produces a significant number of code outputs that work correctly and does not produce any output that gives incorrect results. Although it caused as many compilation errors as other platforms, it managed to solve them in the next iteration.

Throughout the iterative optimization process, each platform's performance was systematically evaluated and refined, highlighting the iterative evolution of AI-driven solutions and their adaptability to optimization efforts. Furthermore, the evaluation metrics employed—accuracy, execution time, executable size, and time complexity—provided a comprehensive understanding of each platform's performance characteristics and comparative advantages.

By elucidating the relative strengths and limitations of each platform, this research aims to inform practitioners, researchers, and decision-makers in leveraging AI-driven solutions effectively and ethically. Moving forward, further research and development efforts are warranted to explore the evolving capabilities of AI platforms, address existing limitations, and foster innovation in AI-driven development practices.

Scientific Ethics Declaration

The authors declare that the scientific ethical and legal responsibility of this article published in EPSTEM journal belongs to the authors.

Acknowledgements or Notes

* This article was presented as an oral presentation at the International Conference on Basic Sciences, Engineering and Technology (www.icbaset.net) held in Alanya/Turkey on May 02-05, 2024.

* We would like to express our sincere gratitude to Software Architect Huseyin Karacali for his extraordinary mentorship and inspiring influence. We would also like to thank TTTech Auto Turkey for its invaluable assistance throughout the development stages of this project.

References

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023, March). Programming is hard-or at least it used to be: Educational opportunities and challenges of Aİ code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V, 1*, 500-506.

Blackbox. (n.d.). *Chat blackbox: AI code generation, code chat, code search*. Retrieved from https://www.blackbox.ai/

ChatGPT. (n.d.). *Chatgpt*. Retrieved from https://openai.com/chatgpt

Pichai, S., & Hassabis, D. (2023, December 6). *Introducing Gemini: Our largest and most capable AI model*. Retrieved from https://blog.google/technology/ai/google-gemini-ai/#sundar-note

Tech Insider Buzz. (2023, November 24). *Blackbox AI: Ai code assistant*. Retrieved from https://techinsiderbuzz.com/blog/blackbox-ai-ai-code-assistant/

Waisberg, E., Ong, J., Masalkhi, M., Zaman, N., Sarker, P., Lee, A. G., & Tavakkoli, A. (2023). Google's AI chatbot "Bard": A side-by-side comparison with CHATGPT and its utilization in ophthalmology. *Eye*, *38*(4), 642–645

Wikimedia Foundation. (2024a, March 27). *Gemini.* Retrieved from https://tr.wikipedia.org/wiki/Gemini

Wikimedia Foundation. (2024a, March 31). *Chatgpt.* Retrieved from https://en.wikipedia.org/wiki/ChatGPT

Wikimedia Foundation. (2024c, April 6). *Microsoft copilot*. Retrieved from https://en.wikipedia.org/wiki/Microsoft_Copilot

Yetistiren, B., Ozsoy, I., Ayerdem, M., & Tuzun, E. (2023a, October 22). Evaluating the code quality of AI-Assisted Code Generation Tools: An empirical study on github copilot, Amazon codewhisperer and Chatgpt. *Arxiv*, *1*,1078.

Author Information

Huseyin Karacali
TTTech Auto Turkey,
Türkiye

Efecan Cebel
TTTech Auto Turkey,
Türkiye
Contact e-mail: e*fecan.cebel@tttech-auto.com*

Nevzat Donum
TTTech Auto Turkey,
Türkiye

To cite this article:

Karacali, H., Cebel, E., & Donum, N. (2024). Performance comparison of AI platforms in solving computer science problems. *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 28,* 326-341.