A Preliminary Analysis on the Code Generation Capabilities of GPT-3.5 and Bard AI Models for Java Functions

Giuseppe Destefanis[1,*], Silvia Bartolucci[2] and Marco Ortu[3]

[1]*Dept. of Computer Science, Brunel University London, UK*
[2]*Dept. of Computer Science, University College London, UK*
[3]*Department of Business and Economics Sciences, University of Cagliari, IT*
* Corresponding author: `giuseppe.destefanis@brunel.ac.uk`

Abstract. This paper evaluates the capability of two state-of-the-art artificial intelligence (AI) models, GPT-3.5 and Bard, in generating Java code given a function description. We sourced the descriptions from CodingBat.com, a popular online platform that provides practice problems to learn programming. We compared the Java code generated by both models based on correctness, verified through the platform's own test cases. The results indicate clear differences in the capabilities of the two models. GPT-3.5 demonstrated superior performance, generating correct code for approximately 90.6% of the function descriptions, whereas Bard produced correct code for 53.1% of the functions. While both models exhibited strengths and weaknesses, these findings suggest potential avenues for the development and refinement of more advanced AI-assisted code generation tools. The study underlines the potential of AI in automating and supporting aspects of software development, although further research is required to fully realize this potential.

1 Important Notice

It is crucial to highlight that the experiments conducted in this study took place between the 10th and the 15th of May 2023. The AI systems analyzed—GPT-3.5 and Bard—are continually learning and evolving systems. As they are fed more data and undergo further training, their performance and capabilities can significantly change over time. Consequently, the results and comparisons drawn in this study are, by their nature, temporally bound and may not hold true in the future. Furthermore, it should be noted that Bard is an experimental tool, as explicitly stated by the provider. Therefore, its performance, behavior, and even availability may fluctuate. As such, readers are advised to interpret the results and conclusions of this paper in light of these considerations.

2   Introduction

Artificial Intelligence (AI) has made significant strides in recent years, particularly in the area of natural language understanding and generation. With the widespread proliferation of large language models capable of performing intricate tasks, research efforts have been undertaken to benchmark and compare the performance of these models [1]. Our analysis focuses on two specific AI models, GPT-3.5 and Bard, investigating their capabilities to generate Java code based on function descriptions obtained from CodingBat.com.

In previous research, the mathematical reasoning capabilities of Generative Pre-trained Tranformers (GPTs) were assessed against state-of-the-art large language models (LLMs) [2]. The system was presented with a set of mathematics questions taken, and the output was rated manually to assess their performance. In [3] GPT-3.5, 4 and Bard's performances in answering medical questions from a neurosurgery question bank were compared, showing that GPT models systematically outperformed Bard. GPT-3.5 and 4 were also shown to be able to pass complex tests, such as the Bar Exam in U.S., with GPT-4 consistently surpassing its LLMs predecessors as well as previous recorded students' performance [4]. GPT-3 models were also tested for code vulnerability and bug detection, showing that their capabilities were limited [5]. Similarly, GPT-3.5 and GPT-4 abilities were tested against other LLMs on coding tasks [6], performing translation in different languages [8], and stock return forecasting [7].

2.1   GPT-3.5

The GPT-3.5 model is a variant of the transformative Generative Pretrained Transformer 3 (GPT-3) model. This language prediction model utilizes a transformer architecture to understand and generate human-like text based on given prompts. GPT-3.5 has 175 billion parameters and has been trained on a diverse range of internet text. Its large scale allows it to generate coherent and contextually relevant sentences over long passages. However, it is important to note that GPT-3.5 does not have any specific knowledge or understanding of the content; it generates responses based on patterns learned during its training.

2.2   Bard

Bard is another advanced AI model known for its exceptional natural language generation capabilities. While details of its architecture and training data are proprietary, Bard has demonstrated strong performance in various natural language understanding and generation tasks. It is particularly noted for its ability to generate creative and contextually nuanced text. Similar to GPT-3.5, Bard does not have a conscious understanding of the content it generates but instead identifies and follows patterns in the data it was trained on.

2.3   CodingBat.com

CodingBat is an online platform that offers coding practice problems to help students learn and master various programming concepts. The platform primarily focuses on Java and Python problems. The problems on CodingBat.com cover a wide range of difficulty levels

and programming concepts, making it an ideal resource for learners of all levels. The solutions to problems are checked and run in real-time, providing immediate feedback to the learners.

This analysis assesses the capabilities of the GPT-3.5 and Bard AI models by feeding them with function descriptions from CodingBat.com. The generated Java code from both models is subsequently evaluated exclusively in terms of correctness, assessed using the real-time code checking system of CodingBat.com. The objective of this research is to analyse the respective strengths and weaknesses of GPT-3.5 and Bard in terms of generating correct Java code. The insights derived from this preliminary study could be helpful in the creation of more refined, AI-assisted code generation tools in the future.

3 Methodology

3.1 Data Collection

For this study, we selected Java function descriptions from five distinct sections of CodingBat.com, each section representing a different level of problem complexity and topic. This selection aimed at ensuring a diverse range of problems that would challenge the code generation capabilities of both the GPT-3.5 and Bard AI models.

The selected sections were as follows:

- **Warmup**: This section contains basic Java functions that are designed as warm up on programming skills. Despite their simplicity, these problems provide a good starting point for testing the AI models' grasp of fundamental programming concepts.
- **String-3**: This section presents harder string problems that usually involve the use of two loops. The challenges in this section test the AI models' ability to handle more complex string manipulation tasks and control structures.
- **Array-3**: This section contains harder array problems, also involving two loops and more complex logic. These problems allow us to evaluate how well the AI models can generate code for more advanced array manipulation tasks and complex logical operations.
- **Functional-2**: This section includes functional filtering and mapping operations on lists using Java lambdas. These tasks test the AI models' capability to generate code using functional programming concepts in Java, which differ significantly from imperative programming tasks.
- **Recursion-2**: This section contains harder recursion problems. These problems challenge the AI models to generate code that implements recursive solutions, which are significantly different in structure from iterative solutions and can be particularly challenging to generate.

From each of these five sections, we collected all the Java function descriptions, resulting in a total of 64 function descriptions for our study. Each of these descriptions was provided as a prompt to both the GPT-3.5 and Bard AI models to generate the corresponding Java code. The generated code was subsequently evaluated based on its correctness, with the results provided by the real-time code checking system of CodingBat.com.

4 Result

In this analysis, we evaluated the Java code generated by GPT-3.5 (ChatGPT) and Bard AI models using function descriptions from five distinct sections of CodingBat.com. Each piece of code was evaluated for correctness, with '1' denoting correct code and '0' representing incorrect code.

The results for the category **WarmUp** are summarized in Table 1, as shown below:

Function	GPT-3.5	Bard
sleepIn	1	1
diff21	1	1
parrotTrouble	1	1
makes10	1	1
nearHundred	1	1
missingChar	1	1
frontBack	1	0
front3	1	0
backaround	1	1
or35	1	1
front22	1	0
startHi	1	1
icyHot	1	1
in1020	1	1
hasTeen	1	1
IoneTeen	1	1
delDel	1	0
mixStart	1	1
startOz	1	0
intMax	1	1
close10	1	1
in3050	1	1
max1020	1	1
stringE	1	1
lastDigit	1	1
endUp	1	1
everyNth	1	1

Table 1. Comparison of Java Code Correctness between GPT-3.5 and Bard for the category WarmUp.

From these results, it can be observed that GPT-3.5 was able to generate correct Java code for all the given function descriptions. On the other hand, Bard had a few instances where it produced incorrect code. Specifically, Bard was unable to generate correct code for the function descriptions `frontBack, front3, front22, delDel, and startOz`.

In the **String-3** category, presented in Table 2, which involves more complex string manipulation problems, the GPT-3.5 model outperformed Bard. GPT-3.5 correctly generated

code for all but one of the given function descriptions. In contrast, Bard only correctly generated code for two functions, `gHappy` and `sumDigit`. Notably, both models failed to generate correct code for the `notReplace` function, suggesting that this problem posed a significant challenge.

Function	GPT-3.5	Bard
countYZ	1	0
withoutString	1	0
equalIsNot	1	0
gHappy	1	1
counTriple	1	0
sumDigit	1	1
sameEnds	1	0
mirrorEnds	1	0
maxBlock	1	0
sumNumber	1	0
notReplace	0	0

Table 2. Comparison of Java Code Correctness for String-3 category between GPT-3.5 and Bard.

In the **Array-3** category, Table 3, which contains harder array problems requiring more complex logic and double loops, GPT-3.5 demonstrated stronger performance compared to Bard. GPT-3.5 correctly generated code for five out of nine function descriptions. On the other hand, Bard struggled with these more complex tasks and was unable to correctly generate code for any of the problems in this category. Both models, however, were unable to generate correct code for the function descriptions `fix34, fix45`, and `squareUp`.

Function	GPT-3.5	Bard
maxSpan	1	0
fix34	0	0
fix45	0	0
canBalance	1	0
linearIn	1	0
squareUp	0	0
seriesUp	1	0
maxMirror	1	0
countClumps	1	0

Table 3. Comparison of Java Code Correctness for Array-3 category between GPT-3.5 and Bard.

In the **Functional-2** category, Table 4, which includes functional filtering and mapping operations on lists with lambdas, both GPT-3.5 and Bard demonstrated high levels of competence and managed to correctly generate code for all the function descriptions. These results suggest that both GPT-3.5 and Bard have robust capabilities when it comes to handling functional programming tasks.

Function	GPT-3.5	Bard
noNeg	1	1
no9	1	1
noTeen	1	1
noZ	1	1
noLong	1	1
no34	1	1
noYY	1	1
two2	1	1
square56	1	1

Table 4. Comparison of Java Code Correctness for Functional-2 category between GPT-3.5 and Bard.

In the **Recursion-2** category, Table 5, which involves harder recursion problems, GPT-3.5 demonstrated relative strength with correct code generated for five out of the eight function descriptions provided. Bard, on the other hand, only managed to correctly generate code for the function `groupSum`. Notably, both models struggled with the problems `groupSum5` and `split53`, suggesting a higher level of difficulty in these problems.

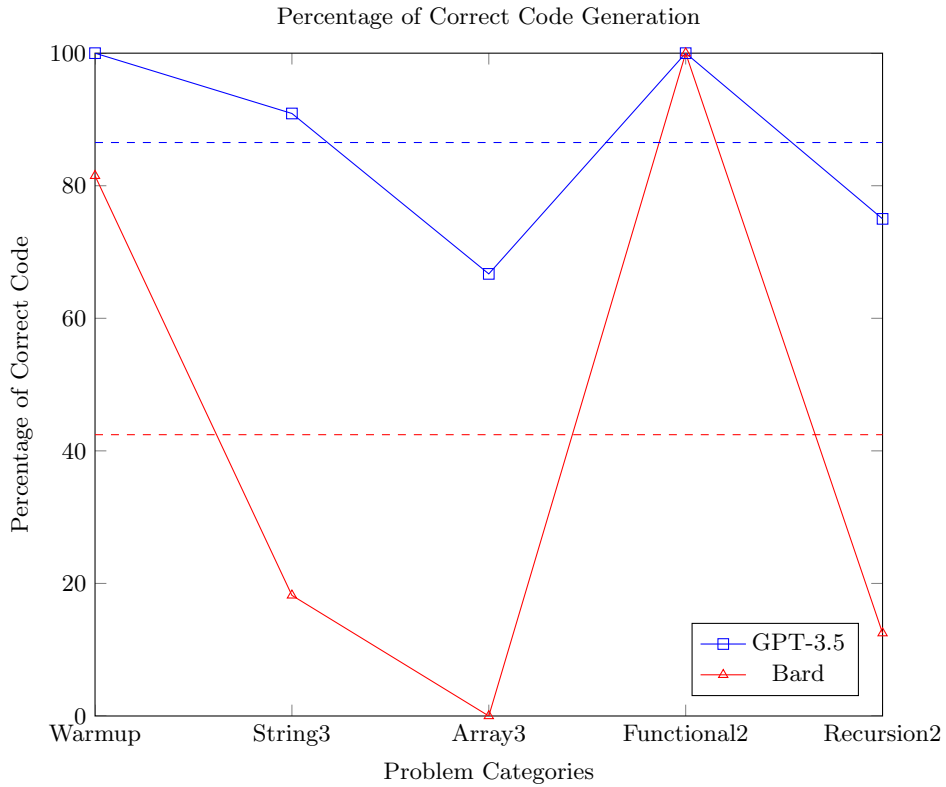| Function	GPT-3.5	Bard
groupSum	1	1
groupSum6	1	0
groupNoAdj	1	0
groupSum5	0	0
groupSumClump	1	0
splitArray	1	0
splitOdd10	1	0
split53	0	0

Table 5. Comparison of Java Code Correctness for Recursion-2 category between GPT-3.5 and Bard.

In summary, across the four categories of problems Warmup, String-3, Array-3, Recursion-2, GPT-3.5 consistently outperformed Bard. For the category Functional-2, both tools generated correct code.

GPT-3.5 generated correct code for 90.6% of the problems, compared to Bard's 53.1%. These results suggest that GPT-3.5 has a stronger ability to generate correct Java code based on function descriptions from a diverse range of problem types and complexities. This analysis also highlights that both models have room for improvement, particularly in handling certain types of complex problems.

Figure 4 illustrates the results. The plot provides a clear visual representation of the performance of GPT-3.5 and Bard across five problem categories, namely Warmup, String3, Array3, Functional2, and Recursion2. For GPT-3.5, the model achieves near-perfect performance in the Warmup and Functional2 categories, demonstrating its effectiveness in handling relatively simple and functional problems. Its performance on String3 and Recursion2

problems is also good, with a slight dip observed in the Array3 category. Contrarily, Bard's performance varies more significantly across categories. This model performs well in the Warmup and perfectly in the Functional2 categories but struggles with more complex categories such as String3, Array3, and Recursion2. The Array3 category proves particularly challenging for Bard, where it fails to generate any correct code. The graph overall highlights the superior overall performance of GPT-3.5 across four categories. However, as stated above, the performance of both models suggests room for improvement, particularly in handling complex problem categories.

Fig. 1. Comparison of the percentage of corrected code generated respectively by GPT-3.5 (ble line) and Bard (red line) in five different problem categories. Dashed lines represent the average performance of GPT-3.5 (blue dashed line) and Bard (red dashed line).

Despite GPT-3.5 outperforming Bard across 4 out of 5 categories, it is evident that both models exhibit a similar performance pattern. This trend indicates a shared difficulty in tackling certain types of tasks, specifically those involving complex array operations and recursion. This shared challenge underscores the complexity and intricacy of these tasks even for sophisticated AI tools. The congruence in their performance trends suggests that

8

these tasks tap into areas where both models, despite their distinct underlying architectures, face significant difficulties. Consequently, these areas represent critical points of attention for future enhancements in AI-assisted code generation.

We also performed a McNemar test [10] to probe the difference between the two AI models, GPT-3.5 and Bard, in terms of their success rates on the same set of function descriptions. McNemar test is suited for this purpose as it is designed for paired nominal data, such as our case where each function description serves as a paired observation evaluated by both AI models. In our study, McNemar test was applied to a 2x2 contingency table, representing the success and failure rates of both AI models. The null hypothesis for McNemar test is that the marginal probabilities of success for the two AI models are equal.

The McNemar Chi-square statistic was found to be equal to 14.062 with 1 degree of freedom, and the associated p-value was 0.0001768. Given this p-value is substantially less than the conventional threshold for statistical significance (0.05), we reject the null hypothesis of equal success rates. This result indicates a statistically significant difference in the success rates of GPT-3.5 and Bard in generating correct Java code. In particular, GPT-3.5 demonstrated a higher success rate compared to Bard.

This outcome underlines the differing strengths of the AI models and encourages further investigation into the factors contributing to these differences.

4.1 Examination of Unsuccessful Code Generation

As an example, we report below the generated function `front3` of the Warmup section for which Bard provides an incorrect solution.

CodingBat asks the following:

Given a string, we'll say that the front is the first 3 chars of the string. If the string length is less than 3, the front is whatever is there. Return a new string which is 3 copies of the front.

front3("Java") ...."JavJavJav"
front3("Chocolate") ... "ChoChoCho"
front3("abc") ... "abcabcabc"

Bard prompt produced the following code:

```
public String front3(String str) {
  if (str.length() < 3) {
    return str;
  }

  return str.substring(0, 3) + str.substring(0, 3) + str.substring(0,
      3);
}
```

which produces the results on CodingBat illustrated in fig. 2

GPT3.5 produced the following code:

```
public String front3(String str) {
        if (str.length() < 3) {
            return str + str + str;
```

Fig. 2. Example of incorrect code generated by Bard.

```
4          } else {
5              String front = str.substring(0, 3);
6              return front + front + front;
7          }
8      }
```

which passed all tests performed by CodingbBat.

5 Threats to Validity

As with any empirical study, several threats to the validity of our results exist.

5.1 Internal Validity

The main internal threat comes from the fact that this study assumes that the AI models generate code that is correct based on the test cases provided by CodingBat.com. However, it is possible that the code could behave unexpectedly with other inputs, leading to potential overestimation of the models' capability. Additionally, the correctness of the generated code is determined only by the test cases from CodingBat.com. If these test cases are incomplete and do not cover all possible cases, this could also affect the results.

5.2 External Validity

In terms of external validity, while we have strived to ensure diversity in our function descriptions, they are nonetheless limited to the Java problems available on CodingBat.com. As such, these results may not generalize to other programming languages, problem domains, or more complex coding tasks. It is also worth noting that both AI models have been trained on diverse datasets, and their performance in this study might not reflect their performance in different contexts or tasks.

5.3 Construct Validity

This study assumes that the quality of code can be measured solely in terms of its correctness. However, other important aspects such as code efficiency, readability, maintainability, and adherence to best practices have not been considered. The absence of these metrics limits the comprehensive evaluation of the code quality generated by the AI models.

Despite these limitations, this study provides valuable insights into the capabilities of current AI models in generating Java code from function descriptions. Future work could focus on addressing these limitations by incorporating a wider range of problem types, considering other programming languages, and including more comprehensive measures of code quality.

6 Future Work

While this analysis provides a preliminary comparison of the code generation capabilities of GPT-3.5 and Bard, there are several directions for future research that could provide more comprehensive insights. One critical aspect that needs further investigation is the complexity of the generated code. Evaluating the generated code not just on correctness but also on factors such as its efficiency, readability, and maintainability could provide a more nuanced understanding of the AI models' code generation capabilities. Advanced tools and techniques for evaluating code complexity should be utilized in future studies to assess code quality. Furthermore, our study was restricted to Java programming language and the function descriptions available on CodingBat.com. Future studies should consider evaluating the AI models' performance with other programming languages and a wider range of problem types and complexities. This would allow for a more comprehensive evaluation of the code generation capabilities of these AI models.

It would also be beneficial to perform a longitudinal study, tracking the progress of these AI models over time. As these models continue to evolve and improve, it would be insightful to understand how their code generation capabilities change with each iteration.

7 Conclusion

This analysis sought to assess the capabilities of two AI models, GPT-3.5 and Bard, in generating Java code from function descriptions. Function descriptions were collected from five diverse categories of problems on CodingBat.com, and the AI models' output was evaluated based on correctness, as verified by the platform's own test cases.

Our findings reveal that GPT-3.5 outperformed Bard across four out of five categories, exhibiting overall a higher rate of correctly generated code. Specifically, GPT-3.5 achieved an overall correctness rate of approximately 90.6%, compared to Bard's 53.1%. This suggests that, at least within the context of this study, GPT-3.5 possesses a stronger ability to generate correct Java code based on diverse function descriptions.

However, it should also be noted that neither model was consistently correct in generating their output. Both models demonstrated shortcomings, particularly when faced with more complex problems such as those in the Recursion-2 category. This highlights the need for continued development and refinement of AI code generation models.

Despite the inherent limitations and threats to validity, we believe that this study contributes valuable insights to the field of AI-assisted code generation. It provides a comparative evaluation of two popular AI models, and highlights areas of strength and potential improvement.

Looking forward, we envision that these findings will stimulate further research and advancement in the field. Specifically, we see potential for studies that explore a wider range of problem types, include other programming languages, and incorporate more comprehensive measures of code quality. Ultimately, this research will possibly contribute to the development of AI models that are capable of reliably assisting humans in diverse coding tasks, thereby enhancing productivity and fostering innovation in software development.

References

1. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. arXiv preprint arXiv:2303.18223.
2. Frieder, S., Pinchetti, L., Griffiths, R. R., Salvatori, T., Lukasiewicz, T., Petersen, P. C., Chevalier, A. & Berner, J. (2023). Mathematical capabilities of chatgpt. arXiv preprint arXiv:2301.13867.
3. Ali, R., Tang, O. Y., Connolly, I. D., Fridley, J. S., Shin, J. H., Zadnik Sullivan, P. L., ... & Asaad, W. F. (2023). Performance of ChatGPT, GPT-4, and Google Bard on a Neurosurgery Oral Boards Preparation Question Bank. medRxiv, 2023-04.
4. Katz, D. M., Bommarito, M. J., Gao, S., & Arredondo, P. (2023). GPT-4 passes the bar exam. Available at SSRN 4389233.
5. Cheshkov, A., Zadorozhny, P., & Levichev, R. (2023). Evaluation of ChatGPT Model for Vulnerability Detection. ArXiv preprint arXiv:2304.07232.
6. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., ... & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712.
7. Lopez-Lira, A., & Tang, Y. (2023). Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models. ArXiv preprint arXiv:2304.07619.
8. Jiao, W., Wang, W., Huang, J. T., Wang, X., & Tu, Z. (2023). Is ChatGPT a good translator? A preliminary study. ArXiv preprint arXiv:2301.08745.
9. Open AI (2023) GPT-4 Technical Report. Available at `https://cdn.openai.com/papers/gpt-4.pdf`.
10. Lachenbruch, P. A. (2014). McNemar test. Wiley StatsRef: Statistics Reference Online.