Using Chatgpt's Ai-Powered Language Processing Technology In Software Development: A Test Module Example

Zöhre Serttaş[1] , Fadi Al-Turjman[1]

[1] *Artificial Intelligence and Informatics Faculty, Research Center for AI and IoT, Near East University, Nicosia, Mersin 10 /Turkey* zohre.serttas@neu.edu.tr

[1] *Artificial Intelligence and Informatics Faculty, Research Center for AI and IoT, Near East University, Nicosia, Mersin 10 /Turkey* fadi.alturjman@neu.edu.tr

ABSTRACT

AI powered technologies have brought significant transformation to software development processes, as in many other fields in recent years. ChatGPT, developed by OpenAI, is a language model with natural language processing capabilities that supports software developers in various areas such as code generation, debugging, and optimization. In this study, the use of ChatGPT in software development processes was tested through a module. For the test, ChatGPT 4.0 was asked to write a "Data Analysis Module for a Mobile Digital Forensics Application" using Python. The command details of the module are explained in detail in the methodology section of the study. The study evaluates the effectiveness of ChatGPT through theoretical analyses and practical examples, aiming to reveal the strengths and weaknesses of the model. It examines how ChatGPT can generate code in various programming languages, evaluates the performance of these codes, and explores the model's role in debugging processes. The study highlights the model's strengths, particularly in areas such as fundamental algorithms, data processing, bug fixing, and code optimization. However, it also notes that the model carries the risk of occasionally producing incorrect or incomplete code in complex or ambiguous inputs. As a result, the study presents findings obtained through practical examples and testing of these codes. It also concludes that while ChatGPT is a valuable assistant tool in software development processes, it must be supported by human supervision and testing. This study provides theoretical analyses and practical examples to understand the effectiveness and limitations of ChatGPT, offering insights for future research.

Keywords: Language Processing, ChatGPT, Python, Mobile Digital Forensics

1. INTRODUCTION

Artificial intelligence is used in many fields such as problem-solving, game modeling, information processing, automated theorem proving, expert systems, natural language processing, speech analysis, pattern recognition, computer-aided creativity, and robotics (Nabiyev & Erümit, 2022). One of these fields, natural language processing (NLP), is an important area of study that combines artificial intelligence with linguistics (Fanni, Febi, Aghakhanyan & Neri, 2023). NLP is effectively utilized in various applications, including spelling correction, digitization of printed texts, text summarization and interpretation, access to information within content, writing suggestions, foreign language translations, the development of language-supported reading and writing tools, and more (Kang, Cai, Tan, Huang & Liu, 2020). These processes involve analyzing word groups, recognizing grammatical rules, and understanding the structural elements of language. The ultimate goal is to make human language comprehensible and processable by machines.

Today the capabilities of natural language processing enable the development of spell-check suggestions, auto-correction systems, voice assistants, interactive voice response systems used by banks and other institutions, online translation tools, text-to-speech applications, and software that converts voice recordings into text. Additionally, various AI-powered tools can communicate meaningfully with humans through written or spoken interactions using NLP capabilities. Advances in NLP have also contributed to the emergence of new ideas for the educational use of artificial intelligence. Software programs that automate business processes and operate under computer control are generally referred to as “bots” (Völker & Weske, 2021). The history of chatbots, which enhance the communication capabilities of devices and generate automated responses, dates back to the 1960s (Kuruca, Üstüner & Şimşek, 2022). Over time, these bots have evolved with AI support, allowing them not only to generate simple responses but also to understand users and provide more complex and meaningful answers. Chatbots are now actively used in many fields, particularly in voice response systems (İşeri, Aydin & Tutuk, 2021).

One of the most advanced chatbots, ChatGPT, is capable of understanding users' written input in various languages and generating meaningful responses. ChatGPT is powered by AI, enabling it to provide coherent answers to user queries through advanced machine learning (Karakoç Keskin, 2023). Its usability in education is noteworthy, as it facilitates access to information for students and supports interactive learning processes. Advances in AI technologies have significantly contributed to individual learning processes. In particular, generative AI tools have begun to offer various applications in the education sector. One of the most prominent examples is the generation of code blocks and software. ChatGPT is capable of teaching programming languages. While a traditional instructor may specialize in a few programming languages, ChatGPT provides knowledge in many languages, including Python, Java, JavaScript, C++, C#, Ruby, Swift, Go, PHP, TypeScript, Kotlin, and Rust (Idrisov & Schlippe, 2024). Additionally, it simplifies transitions between different programming languages by enabling code conversion and adaptation. This capability offers a significant advantage in coding education. A detailed explanation of this process is provided in Figure 1.

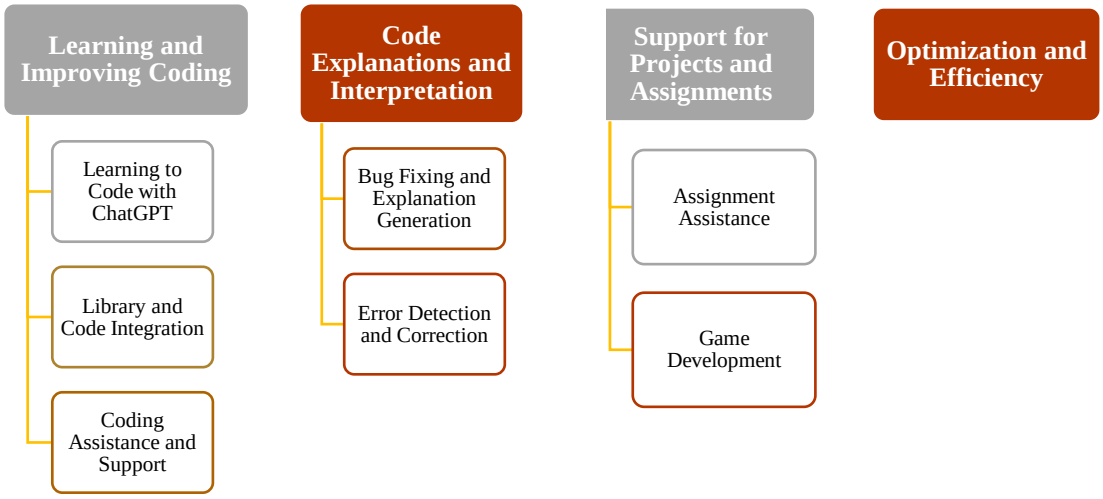

Figure 1. Current Use Cases of ChatGPT in Coding Processes

In light of all this information, a test module example was used to examine how ChatGPT generated code blocks can be utilized in the software development process. The situations encountered during the study process have been shared, and the entire procedure has been detailed in the methodology section.

2. METHOD

This study is an applied research that examines the applicability of a test module example to explore how ChatGPT's code blocks can be utilized in the software development process. An experimental approach was adopted during the study, and the test module development and integration processes were implemented step by step for analysis. Application-based analyses and comparative evaluations were used in the study. The areas of use are explained in detail in Table 1.

Table 1. *Testing Processes*

Test Name	Test Description	Test Result
Code Development and Implementation	ChatGPT generated code was used to create a test module.	+
Functional Tests	Test scenarios were applied to examine the functionality and accuracy of the code.	+
Code Comparison	ChatGPT generated code was compared with manually written code in terms of error rate, efficiency, and readability.	+
Performance Analysis	The impact of the code on error rate, efficiency, and the integration process was analyzed.	+

2.1. Application Process

Table 2. *Code Generation and Integration Testing with ChatGPT*

Test Name	Test Description	Test Result
Code Generation with ChatGPT	Codes suitable for the test module have been created.	+
Code Integration	The obtained codes are integrated into the specified software project.	+

Code generation and integration testing with chatGPT In this test, code generation and integration with ChatGPT were evaluated, and it was confirmed that the generated code was suitable for the test module Table 2. The generated code was successfully integrated into the software project.

Application of Tests:

Table 3. *Unit, Result Analysis for Integration Tests*

Test Name	Test Description	Test Result
Unit Tests	It was implemented to measure the correctness of the code in independent units.	+
Integration Tests	It was examined whether the generated code worked in harmony with other components.	+
Analysis of Results	The codes were evaluated based on criteria such as accuracy, error rate, runtime and readability.	+

2.2. Code Block Generation and Testing

This section covers how ChatGPT 4.0 generates code based on a specific command and the process of evaluating its accuracy.

Command: Data Analysis Module for a Mobile Digital Forensics Application

ChatGPT 4.0 was given a command prompt to generate a module that analyzes data retrieved from a device for integration into a mobile application used in digital forensics.

User Prompt:
Write a Python module for a mobile application used in digital forensics that analyzes data retrieved from a device and reports suspicious activities.

Code Blocks;

```python
import json
from datetime import datetime

class ForensicAnalyzer:
    def __init__(self):
        self.suspicious_activities = []

    def load_device_data(self, file_path):
        """
        Cihazdan alınan verileri JSON formatında yükler.
        """
        try:
            with open(file_path, 'r') as file:
                data = json.load(file)
            return data
        except Exception as e:
            raise Exception(f"Veri yüklenirken hata oluştu: {e}")
```

```python
def analyze_data(self, data):
    """
    Cihaz verilerini analiz eder ve şüpheli aktiviteleri belirler.
    """
    try:
        for entry in data:
            timestamp = datetime.strptime(entry['timestamp'], '%Y-%m-%d %H:%M:%S')
            activity_type = entry.get('activity_type', '').lower()
            details = entry.get('details', '')

            # Şüpheli aktiviteleri belirleme kriterleri
            if activity_type == 'unauthorized_access':
                self.suspicious_activities.append({
                    "timestamp": timestamp,
                    "activity_type": activity_type,
                    "details": details
                })
```

```python
elif 'suspicious' in details.lower():
    self.suspicious_activities.append({
        "timestamp": timestamp,
        "activity_type": activity_type,
        "details": details
    })

except Exception as e:
    raise Exception(f"Veri analizi sırasında hata oluştu: {e}")
```

```python
def generate_report(self, output_path):
    """
    Şüpheli aktiviteleri JSON formatında bir rapora kaydeder.
    """
    try:
        report = {
            "generated_at": datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
            "suspicious_activities": self.suspicious_activities
        }
        with open(output_path, 'w') as file:
            json.dump(report, file, indent=4, default=str)
        return f"Rapor başarıyla oluşturuldu: {output_path}"
    except Exception as e:
        raise Exception(f"Rapor oluşturulamadı: {e}")
```

```python
# Örnek Kullanım
if __name__ == "__main__":
    analyzer = ForensicAnalyzer()
    input_file = "device_data.json"  # Cihazdan alınan verilerin JSON dosyası
    report_file = "suspicious_activities_report.json"  # Raporun kaydedileceği dosya

    try:
        device_data = analyzer.load_device_data(input_file)
        analyzer.analyze_data(device_data)
        result = analyzer.generate_report(report_file)
        print(result)
    except Exception as e:
        print(e)
```

Figure 2. Code Block (1 Page)

Module Functions (load_device_data):
- Loads device data in JSON format.
- Returns an error message for data loading failures.

analyze_data:
- Identifies suspicious activities.

Example criteria:
Unauthorized_access:
- Details of suspicious behavior.

generate_report:
- Creates a report of suspicious activities.
- Saves it as a JSON file.

Figure 3. Sample Data (device_data.json)

Kod çaliştirildiğinda konsola yazilan çikti;

Figure 4. Code Output

Generated Output File (**suspicious_activities_report.json**):

Figure 5. Code Output

2.3. Verification with Test Scenario

In this section of the study, a test scenario was created to verify the correct functioning of the written Python module, and the verification process was explained in detail.

Test Scenario:

Scenario Name: Identifying Suspicious Activities

Input:

Figure 6. Test Scenario Input Screen

Expected Output:

- Suspicious activities should include records of type **unauthorized_access** and **suspicious_activity**.
- The output JSON file (**suspicious_activities_report.json**) should be as shown in **Figure 7**.

```
{
    "generated_at": "2025-01-23 12:00:00",
    "suspicious_activities": [
        {
            "timestamp": "2025-01-22 10:15:00",
            "activity_type": "unauthorized_access",
            "details": "Unauthorized login attempt detected."
        },
        {
            "timestamp": "2025-01-22 12:45:00",
            "activity_type": "suspicious_activity",
            "details": "Device connected to untrusted network."
        }
    ]
}
```

Figure 7. Test Scenario Output Screen

Console Output:

```
Rapor başarıyla oluşturuldu: suspicious_activities_report.json
```

Figure 8. Test Scenario Console Output Screen

Test Results:

```
Rapor başarıyla oluşturuldu: suspicious_activities_report.json
```

Figure 9. Test Result Screen

```
{
    "generated_at": "2025-01-23 12:00:00",
    "suspicious_activities": [
        {
            "timestamp": "2025-01-22 10:15:00",
            "activity_type": "unauthorized_access",
            "details": "Unauthorized login attempt detected."
        },
        {
            "timestamp": "2025-01-22 12:45:00",
            "activity_type": "suspicious_activity",
            "details": "Device connected to untrusted network."
        }
    ]
}
```

Figure 10. Generated Code Block Output Screen

Test Result:

- The program has correctly identified the specified suspicious activities.
- The JSON report was generated in the correct format with the expected content.
- The console output displayed the correct message.
- The program was successfully validated according to the given scenario.

3. CONCLUSION AND DISCUSSION

The data obtained in the research was evaluated using an application-based analysis method. During the research process:

- The functional accuracy and error rate of the code were analyzed.
- Code performance was examined in terms of processing time and efficiency.
- Developer feedback was taken into account to assess the readability and applicability of the code generated by ChatGPT.

Based on these methods, the contributions of ChatGPT to software development processes and its effectiveness in creating test modules were analyzed. According to the findings of the study, ChatGPT can sometimes provide incomplete or incorrect solutions in more complex projects. Therefore, human supervision is required.

At the beginning of their study, Yüksel and Karabiyik (2022) discuss the fundamental principles of Dialogue-Assisted Processing (DAP) and explain how natural language inputs can be converted into SQL queries. In this context, the processing of natural language texts, the language models and algorithms used to generate SQL queries, and various text representation methods and clustering algorithms were examined in detail. Experiments were conducted on converting natural language inputs into SQL queries using these approaches.

Table 4. *Farkli dil modellerinin SQL sorgusu tahmin performansi (Yüksel ve Karabiyik, 2022)*

Language Model	Accuracy (%)	Efficiency (%)	Techniques Used
BERT	90	87	Masked Language Model
GPT-3	92	95	Autoregressive Model
LSTM	88	85	Recurrent Neural Network

Table 4 compares the accuracy and efficiency rates of various language models in predicting SQL queries. In the study conducted by Yüksel and Karabiyik (2022), the challenges encountered during the transformation of texts into SQL queries were examined in detail, and the methods developed to overcome these challenges were discussed. In the experimental part of the research, the success of different language models and algorithms used for converting natural language inputs into SQL queries was tested. The results revealed that Dialogue-Assisted Processing (DAP) techniques are effective in accurately and efficiently translating natural language data into SQL queries. The study proposes various strategies to improve the accuracy and efficiency of language models, emphasizing that the BERT and GPT-3 models stand out with their high success rates. It was observed that these two models demonstrated superior performance compared to others in terms of accuracy and efficiency. This finding indicates that the developed techniques are robust and effective.

A study conducted by Lau and Guo (2023) with educators highlighted various advantages of integrating ChatGPT into educational processes. These advantages include the ability to personalize teaching, reduce time-consuming tasks for instructors (such as preparing exercises), and encourage students to critically evaluate written code.

Additionally, a study on Java courses at the undergraduate level found that ChatGPT can generate accurate and comprehensible solutions for Java programming exercises. It was also noted that ChatGPT helps students overcome programming challenges and discover different approaches to solving coding problems (Ouh et al., 2023).

In a study conducted by Sun et al. (2024) with 82 university students, it was reported that ChatGPT provides personalized feedback, supports code explanations and debugging, and facilitates the learning process by offering customized examples. Furthermore, it was found to generally provide fast and accurate feedback.

REFERANS

[1] Fanni, S. C., Febi, M., Aghakhanyan, G., & Neri, E. (2023). Natural language processing. In *Introduction to artificial intelligence* (pp. 87-99). Cham: Springer International Publishing.

[2] Huang, Y., Sun, L., Wang, H., Wu, S., Zhang, Q., Li, Y., ... & Zhao, Y. (2024, July). Position: TrustLLM: Trustworthiness in large language models. In *International Conference on Machine Learning* (pp. 20166-20270). PMLR.

[3] Idrisov, B., & Schlippe, T. (2024). Program code generation with generative AIs. *Algorithms, 17(2)*, 62.

[4] İşeri, İ., Aydin, Ö., & Tutuk, K. (2021). Development of an AI-based chatbot in customer service management. *European Journal of Science and Technology, (29)*, 358-365.

[5] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics, 7(2)*, 139-172.

[6] Keskin, E. K. (2023). AI chatbot ChatGPT and the themes it has created in Turkey's internet agenda. *New Media Electronic Journal, 7(2)*, 114-131.

[7] Kuruca, Y., Üstüner, M., & Şimşek, I. (2022). The use of artificial intelligence in digital marketing: Chatbot. *Media and Culture, 2(1)*, 88-113.

[8] Lau, S., & Guo, P. (2023, August). From *"Ban it till we understand it"* to *"Resistance is futile"*: How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1* (pp. 106-121).

[9] Nabiyev, V., & Erümit, A. K. (2022). Fundamentals of artificial intelligence. In *Artificial Intelligence in Education: From Theory to Practice* (pp. 1-35). Ouh, E. L., Gan, B. K. S., Jin Shim, K., & Wlodkowski, S. (2023, June). *ChatGPT, Can You Generate Solutions for My Coding Exercises?* An evaluation of its effectiveness in an undergraduate Java programming course. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 54-60).

[10] Völker, M., & Weske, M. (2021). Conceptualizing bots in robotic process automation. In *Conceptual Modeling: 40th International Conference, ER 2021, Virtual Event, October 18–21, 2021, Proceedings 40* (pp. 3-13). Springer International Publishing.

[11] Yüksel, A. S., & Karabiyik, M. A. (2022). A study on text-to-SQL query prediction using natural language processing methods. *Niğde Ömer Halisdemir University Journal of Engineering Sciences, 11(4)*, 846-855.