BRINGING NLP TO JAVA CODING - CHATGPT CODE DEVELOPER

Alexandru TĂBUȘCĂ[1]

Andrei LUCHICI[2]

Mihai Alexandru BOTEZATU[3]

Abstract

The integration of ChatGPT, a state-of-the-art NLP (Natural Language Processing) model developed by the company OpenAI, with the Java programming language provides significant advancements in coding generation and software development. ChatGPT has the ability to generate useful and correct, human-like, text responses, a fact that offers developers a robust tool for automating tasks such as code generation, documentation, error diagnosis, and test case creation. Java, renowned for its extensive ecosystem and backend capabilities, is an ideal choice for leveraging ChatGPT's potential. This paper explores methodologies and best practices for integrating ChatGPT with Java, focusing on API interaction, error handling, and performance optimization. Developers can employ HTTP libraries such as OkHttp and frameworks like Spring Boot to create intelligent and scalable applications. By dynamically generating prompts, integrating with databases, and caching frequent responses, developers can enhance the efficiency of their applications. However, limitations remain, such as the need for manual intervention in resolving library version conflicts and addressing complex tasks beyond AI's current capabilities. Real-life use cases discussed include generating Java code, translating programming languages, enforcing code styles, and supporting real-time applications. Despite occasional inaccuracies in AI-generated outputs, ChatGPT's speed and versatility make it a valuable assistant for both novice and, mostly, experienced developers. The paper concludes by emphasizing the need to experiment with AI-driven tools to maximize productivity and stay abreast of technological advancements in software engineering. As there is no way back to a time without AI support, developers should embrace the new paradigm of assisted programming languages code generation and make the best out of the new environment, while remaining focused on any new relevant developments in this field.

Keywords: ChatGPT integration, java development tools, ai-assisted programming, code automation, NLP in coding

JEL Classification: C8, O39

[1] PhD, Associate Professor, Romanian-American University, Romania, alex.tabusca@rau.ro
[2] PhD, Lecturer, Romanian-American University, Romania, andrei.luchici@rau.ro
[3] PhD, Habil Professor, Romanian-American University, Romania, mihai.botezatu@rau.ro

1. Introduction

The current time is probably one of the most important and mind-startling occurrences in the field of software development since its inception. Today, the programmer is not alone anymore, or together with his team-mates, he can have a permanent assistant, very smart and educated, in the form of a modern AI prompt-conversation solution. Surely the best known and widely recognized NLP model (Natural Language Processing) application and also one of the most versatile tools of its kind is the ubiquitous ChatGPT, developed by the company called OpenAI. The ability of the NLP to generate human-like text answers transforms it into a very attractive and useful option for software developers looking to integrate such AI-driven conversational agents into their software applications. Java, as one of the most widely used and powerful programming languages today, offers a lot of robust capabilities for backend development, making it a very suitable choice for integration with the ChatGPT tool. This paper present various methodologies and best practices for using Java to interact with ChatGPT, enabling the creation of intelligent, scalable applications.

With this article, I want actually to encourage the use of this technology in our everyday work or at least give it a try in order to make sure that you understand its possibilities, and its shortcomings.

ChatGPT can significantly improve the performance of a developer. It can generate different prototypes and it can help save our time for more complex things than writing basic boilerplate code.

There is quite a hype right now around the ChatGPT NLP technology, but this is still hype mixed with reality. In order to demystify the AI we need to train ourselves and even force ourselves to start implementing AI support in our daily workloads. This is a fact indeed easier said than done, mostly because we are in fact not used to relying on the support of the AI assistance… or not yet anyway. We need to experiment first in order to find the most useful and practical use cases, the cases and scenarios when we can apply it to practice with the most success.

2. Understanding ChatGPT API

To effectively utilize the ChatGPT API, developers should have a foundational understanding of RESTful principles and how they apply to modern API design. This includes knowledge of HTTP methods like GET, POST, and DELETE, as well as understanding the importance of structured data formats like JSON for sending and receiving data. Familiarity with tools like Postman or cURL can also prove invaluable for testing API endpoints during the development process. By mastering these basics,

programmers can ensure seamless integration and debugging when working with the ChatGPT API.

Another critical aspect to consider is the proper handling of API responses. Developers must account for the various response codes the API may return, such as 200 for success, 400 for bad requests, and 429 for exceeding rate limits. Implementing robust error handling ensures the application remains functional even during unexpected scenarios. Furthermore, understanding the structure of the response payload is essential for extracting useful data, such as the generated text or metadata, and transforming it into actionable insights for the application.

Lastly, managing API security is paramount to protect sensitive data and prevent unauthorized access. Developers should adopt best practices like storing API keys securely in environment variables or using secret management solutions. Regularly rotating API keys and monitoring their usage can also mitigate the risk of exploitation. Additionally, adhering to OpenAI's usage policies and implementing user authentication mechanisms when exposing ChatGPT-powered features helps maintain compliance and trustworthiness in production environments.

In this section we will briefly list the basic knowledge a programmer must have before enrolling his AI assistant for writing support code in Java.

The ChatGPT API, offered by OpenAI, actually provides a RESTful interface for developers to interact with the model. By sending HTTP requests with appropriate parameters, developers can obtain responses from ChatGPT to build features like chatbots, text summarization tools, or content generators. Key aspects of the API include:

1. Endpoints: The API typically offers endpoints for generating text, retrieving models, and managing sessions.
2. Authentication: Access to the API requires an API key, which serves as a credential for authorized usage.
3. Rate Limiting: OpenAI enforces rate limits to ensure fair usage, which developers need to account for in their applications.

3. Setting Up a Java Environment

Before integrating the ChatGPT tool into our development environment, it is essential that the we set up the Java development environment correctly. Among the necessary steps for these end we can include:

1. Installation of Java - the developer should ensure that the Java Development Kit (JDK), version 8 or higher is installed.
2. Building Tools - the developer should use recognized tools, such as Maven or Gradle, to manage the dependencies and build the project.
3. HTTP Libraries – the developer should include libraries such as Apache HttpClient, OkHttp or Spring RestTemplate in order to handle HTTP requests.

Below, we can see a source-code example for a Maven dependency with OkHttp:

```
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.10.0</version>
</dependency>
```

3.1. Making API Calls

The Java programming language provides the developers with lots of libraries for making HTTP requests. Below we can see an example of such a case, by the use of the same OkHttp library:

```
import okhttp3.*;
import java.io.IOException;
public class ChatGPTIntegration {
    private static final String API-URL = "https://api.openai.com/v1/completions";
    private static final String API-KEY = "ChatGPT_api_key";
    public static void main(String[] args) throws IOException {
        OkHttpClient client = new OkHttpClient();
        String jsonRequest = "{" +
            "\"model\": \"gpt-4\"," +
            "\"prompt\": \"Hello, what can I do for you today?\"," +
            "\"max_tokens\": 150" +
        "}";
        RequestBody body = RequestBody.create(
```

```
        jsonRequest, MediaType.parse("application/json")
    );
    Request request = new Request.Builder()
        .url(API-URL)
        .addHeader("Authorization", "Bearer "+API-KEY)
        .post(body)
        .build();
    Response response = client.newCall(request).execute();
    if (response.isSuccessful()) {
        System.out.println(response.body().string());
    } else {
        System.err.println("Request failure: " + response.code());
    }
  }
}
```

3.2. Parsing Responses

Responses received from ChatGPT are generally returned in the ubiquitous JSON format. Libraries like Jackson or Gson can be used to parse these responses and transform them into Java objects, usable further for out applications.

Below we can see an example using Jackson:

```
import com.fasterxml.jackson.databind.ObjectMapper;

String responseBody = response.body().string();

ObjectMapper mapper = new ObjectMapper();

ChatGPTResponse chatResponse = mapper.readValue(responseBody,
ChatGPTResponse.class);

class ChatGPTResponse {
  public String id;
  public String object;
  public List<Choice> choices;
```

```
    static class Choice {
        public String text;
    }
}
```

3.3. Handling Errors and Rate Limits

When working with ChatGPT, robust error handling ensures that your application can gracefully recover from different issues. Key scenarios to handle include:

1. Timeouts: Implement retry mechanisms with exponential backoff.
2. Invalid Responses: Validate JSON responses to prevent processing invalid data.
3. Rate Limits: Monitor API usage and implement logic to pause or limit requests when nearing rate limits.

Example error handling code:

```
try {
    Response response = client.newCall(request).execute();
    if (response.isSuccessful()) {
        System.out.println(response.body().string());
    } else {
        System.err.println("API error: " + response.code());
    }
} catch (IOException e) {
    System.err.println("Network error: " + e.getMessage());
```

3.4. Advanced Use Cases

1. Dynamic Prompt Generation: Java applications can dynamically construct prompts based on user input, enhancing personalization.
2. Integration with Databases: Store and retrieve conversational histories in relational databases like MySQL or NoSQL databases like MongoDB.
3. Real-Time Applications: Use frameworks like Spring Boot to create real-time chat applications powered by ChatGPT.

Below we can see an example of integrating with Spring Boot:

```
@RestController
@RequestMapping("/chat")
public class ChatController {
    @PostMapping("/ask")
    public ResponseEntity<String> askChatGPT(@RequestBody String userInput) {
        // Use OkHttp or RestTemplate to make API call
        // Return the ChatGPT response
        return ResponseEntity.ok("ChatGPT Response");
    }
}
```

3.5. Optimizing Performance

Effective performance optimization requires careful planning and implementation of concurrency strategies. When handling multiple users, leveraging thread pools can help efficiently manage system resources and prevent overloading. Asynchronous request handling further enhances performance by allowing the system to process multiple requests simultaneously without blocking threads. For example, the use of Java CompletableFuture or of the reactive frameworks like Project Reactor or RxJava can quite significantly improve throughput and reduce response times in high-traffic applications.

In addition to concurrency, caching plays a vital role in reducing the load on APIs and improving user experience. Implementing a robust caching mechanism, such as using libraries like Ehcache or Redis, allows applications to store frequent responses locally or in-memory. This reduces the number of redundant API calls, thus saving bandwidth and decreasing latency. However, developers must also consider cache invalidation strategies to ensure that the application serves accurate and up-to-date information. Balancing these techniques with thorough load testing using tools like Apache JMeter or Gatling helps simulate real-world usage patterns, identify bottlenecks, and optimize the application's scalability under heavy traffic conditions.

Performance optimization is crucial when integrating with APIs. Summarizing, the main lines regarding the possibilities to optime performance go along the below topics:

1. Concurrent Requests: Use thread pools or asynchronous requests for handling multiple users.
2. Caching Responses: Cache frequent responses in order to reduce the number of API calls and to improve response time.
3. Load Testing: Use tools like JMeter or Gatling to simulate and test application load.

3.6. Security Considerations

The most relevant and widely used topics in regards to security considerations are listed below:

1. API Key Management: Never hard-code API keys. Use secure storage solutions like AWS Secrets Manager or environment variables.
2. Data Privacy: Ensure compliance with data protection regulations by anonymizing sensitive user data.
3. Secure Communication: Use HTTPS for all API interactions.

4. Real-life use cases for ChatGPT in daily Java development

4.1. Code generation

ChatGPT was originally created as a language model. Due to this fact, it is basically meant for reading text and producing text. Programming language code, Java in our case, is a subtype of the text, so ChatGPT can understand issues related to it and generate this type of text also [1].

At this time the entire world is changing, in more than one way and at many levels, and we should not actually resist those changes but in fact embrace them and make the best of it [2]. We can use the newly opened possibilities to improve our productivity. The current situation is not actually something completely new for mankind – even though it had different faces throughout history – scientific progress was always entangled with the fear of losing jobs. Nevertheless, humankind was always able to adapt and this time it will be no different – there are even more types of jobs coming to implement all the AI capabilities that we are being offered in our current environment [3].

ChatGPT can generate Java programming language code based on the natural language input (prompt). As an example, one software developer can just enter/prompt a quite short description of the code he needs, and ChatGPT will try to generate the corresponding code,

written in Java, to solve the request. This scenario can save developer time and reduce the chance of (some) errors in the code.

For example, we can use the below prompt:

Prompt: Generate a JPA model for Student, University and School.

For this request, ChatGPT understands that it should define a Many-To-One relationship between University, School and Student. Also, it defines properties, for example, university name, school name, student's First_Name and Last_Name - elements we did not purposely ask for, but these elements were in reality included within the response. This fact shows that ChatGPT can deliver the missing parts of its task by understanding a somewhat complex context.

The result of the previously mentioned prompt session with ChatGPT is available in the below Figure 1.

```java
import javax.persistance.*;
@Entity
@Table(name = "student")
public class Student{
    @ID
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @ManyToOne
    @JoinColumn(name = "school_id")
    private School school;

    //get and set functionality
}

@Entity
@Table(name = "school")
public class School {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name")
    private String name;

    @ManyToOne
    @JoinColumn(name = "university_id")
    private University university;

    @OneToMany(mappedBy = "school")
    private List<Student> students;

    //get and set functions
}

@Entity
@Table(name = "university")
public class University {
    @Id
    @GenerationValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name")
    private String name;

    @OneToMany(mappedBy = "university")
    private List<School> schools;

    //get and set functions
}
```

Figure 1. This is the illustration of the dialog with Chat GPT

4.2. Documentation generation

The ChatGPT tool is able to also generate documentation for our Java code. This is done based on the Java code itself (perhaps provided by ChatGPT also) and on any other comments or annotations relevant for this purpose. This approach can not only help developers in order to create more complete and accurate documentation, without the need to write it by themselves, but it is also incomparably faster.

For example, we can use the below prompt:

Prompt: For the above generated code please generate documentation

4.3. Test case generation

ChatGPT can generate test cases for Java code based on natural language input. For example, a developer can describe a desired test case, and the model will generate the corresponding Java code. This can save developer's time and ensure that the test cases are comprehensive and accurately reflect the desired behavior of the code.

For example, we can use the below prompt:

Prompt: For the above generated code please generate Unit / BDD tests

4.4. Error diagnosis

The ChatGPT tool can also help Java developers to diagnose potential errors in their Java code. For example, one developer could provide a model-code and a description of the encountered error, waiting for the model to propose different possible solutions to mitigate the issues it finds.

For example, we can use the below prompt:

Prompt: For the above generated code please find the errors in the code

4.5. Error diagnosis

ChatGPT can help Java developers diagnose errors in their code. For example, a developer can provide a model with a description of the error, and the model will suggest possible solutions. Of course, the ChatGPT proposals are not 100% sure and for any complex task the bulletproof solution is actually based on experienced programmers/developers.

For example, we can use the below prompt:

Prompt: For the above generated code please find the errors in the code

4.6. Language translation

The ChatGPT tool can be used as a "translator" between two different programming languages, transforming itself into a valuable help for reusing previously coded algorithms available into a different coding language.

For example, we can use the below prompt:

Prompt: For the above generated code please translate it into the language C# language

4.7. Code style enforcement

The ChatGPT tool is also able to be used to enforce code style standards by suggesting changes to the Java code in order to make it more consistent with a certain style guide that we want to strictly adhere to.

For example, we can use the below prompt:

Prompt: For the above generated code please validate that all constants are upper-cased

5. And… is not exactly working! Why?

For the time being, at least, developers must verify everything that is generated with the ChatGPT tool very carefully [4]. The code generated with ChatGPT's help is often non-functional or is based on several wrong assumptions and it works but in unexpected ways [5]. As a consequence, in real-life, the code generative tool is not, at least yet, a complete replacement for an experienced (senior or at mid-tier) developer. Specialists consider that the quality of the generated code is actually similar to a junior developer's one generally, while the code generation speed is obviously much higher than any human could possibly do.

There are two most frequent reasons for this failure to work out of the box – or out of the generative prompt to be more precise.

1. GPT is in fact based on previously existing programming language code available on the internet. This code might quite often be using different versions of the same libraries. For example, a section of code might uses library L version X, and another section of code might be using the same general library L but version Y. These different variants can be incompatible, at least to a certain degree, but the ChatGPT tool does not has the understanding and it produces a mix of different versions implemented in the same proposed code. The developer can ask the tool to use only

a certain version of a certain library but this is not a guarantee; in some cases the results will be better, in other cases there is no difference. If the more specific request does not help the developer has to actually check for incompatibilities manually and make all the possible necessary changes.

2. In some scenarios, the task send to the ChatGPT tool can be too complex for it. In such a case it would in fact be possible to split the task into several subtasks, but this operation is quite often too difficult for the current implementations of NLP. Such a complex task is currently beyond the features and capabilities of the publicly available AI, but this might change in future versions quite fast.

6. Conclusions

Integrating ChatGPT with Java opens a plethora of possibilities for building intelligent and efficient applications. By following best practices and leveraging Java's rich ecosystem, developers can harness the power of AI to deliver innovative solutions. This combination not only enhances user experience but also provides a scalable approach to addressing complex challenges in software development.

The integration of ChatGPT with Java represents a paradigm shift in software development, combining artificial intelligence and robust backend capabilities to revolutionize coding workflows. By leveraging ChatGPT's API, developers can streamline tasks such as code generation, error diagnosis, and documentation creation, allowing them to focus on complex problem-solving. The methodologies outlined, including API interaction, error handling, and performance optimization, empower developers to build intelligent, scalable, and efficient applications.

Despite its remarkable capabilities, ChatGPT is not without limitations. Challenges such as handling library version conflicts and managing complex tasks highlight the need for human oversight. Nevertheless, its ability to accelerate prototyping, enhance productivity, and generate creative solutions far outweighs its current shortcomings. By experimenting with use cases like dynamic prompt generation, test case creation, and language translation, developers can unlock AI's potential to augment their day-to-day operations. Furthermore, large scale usage of these tools will not only benefit their user but also their developers – supporting the accelerated development and enhancement of even better tools.

Looking ahead, as AI and NLP technologies continue to evolve further and further, support tools like ChatGPT are meant to play an increasingly more important and integral role in the field of modern software development. To fully realize these benefits, developers must embrace a mindset of continuous learning and innovation. By combining the power of AI with human ingenuity, we can navigate the challenges and opportunities of this new

technological frontier, shaping a real future in which the software development and code-generation are concomitantly faster, smarter, more accessible and permanently improving.

Acknowledgment

The paper is based on research carried out in part within the Center for Computational Science and Machine Intelligence (CSMI) of the Romanian-American University's School of Computer Science for Business Management.

References

[1] https://www.aegissofttech.com/insights/chatgpt-for-java/ - ChatGPT for Java Developers: Top 12+ Use Cases. Thomas E, 18.12.2024

[2] https://www.infoworld.com/article/2338520/build-a-java-application-to-talk-to-chatgpt.html - Build a Java application to talk to ChatGPT. Tyson M, 18.12.2024

[3] M. Guo - Java Web Programming with ChatGPT – "2024 5th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM)", Nanjing, China, 2024, pp. 834-838, doi: 10.1109/ICMTIM62047.2024.10629560.

[4] https://digma.ai/java-developer-vs-chatgpt-part-i-writing-a-spring-boot-microservice/ - Java Developer Vs ChatGPT: Writing a Spring Boot Microservice - Ron Dover, 18.12.2024

[5] Eng Lieh Ouh, Benjamin Kok Siew Gan, Kyong Jin Shim, Swavek Wlodkowski - ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course – ITiCSE 2023: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, 2023

Bibliography

Bao Lingfeng, Xing Zhenchang, Xia Xin, Lo David - 2018. VT-Revolution: Interactive programming video tutorial authoring and watching system. IEEE Transactions on Software Engineering, Vol. 45, 8 (2018), 823--838.

Buck Alan - Practical Java Programming with ChatGPT: Develop, Prototype and Validate Java Applications by integrating OpenAI API and leveraging Generative AI and LLMs – ISBN 978-8119416790, Orange Education, 2023

Guo M. - Java Web Programming with ChatGPT – "2024 5th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM)", Nanjing, China, 2024, pp. 834-838, doi: 10.1109/ICMTIM62047.2024.10629560.

Haque Mubin Ul, Dharmadasa Isuru, Sworna Zarrin Tasnim, Rajapakse Roshan Namal, Ahmad Hussain. 2022. "I think this is the most disruptive technology": Exploring Sentiments of ChatGPT Early Adopters using Twitter Data. arXiv preprint arXiv:2212.05856 (2022).

Hopkins B. - ChatGPT for Java – ISBN 979-8-8688-0115-0, Apress Berkeley, CA, 2024

Ouh Eng Lieh, Gan Benjamin Kok Siew, Shim Kyong Jin, Wlodkowski Swavek - ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course – ITiCSE 2023: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, 2023

https://www.aegissofttech.com/insights/chatgpt-for-java/ - ChatGPT for Java Developers: Top 12+ Use Cases. Thomas E, 18.12.2024

https://www.comsol.com/support/learning-center/article/86731 - Modeling with ChatGPT. 18.12.2024

https://digma.ai/java-developer-vs-chatgpt-part-i-writing-a-spring-boot-microservice/ - Java Developer Vs ChatGPT: Writing a Spring Boot Microservice - Ron Dover, 18.12.2024

https://www.infoworld.com/article/2338520/build-a-java-application-to-talk-to-chatgpt.html - Build a Java application to talk to ChatGPT. Tyson M, 18.12.2024