Chamando a API GPT3.5 em Java

Introdução

A integração de APIs é uma habilidade essencial no desenvolvimento de software, especialmente ao trabalhar com serviços de inteligência artificial como o **GPT-3.5-turbo**.

Este guia oferece uma explicação didática, passo a passo, de como configurar um projeto Java para se comunicar com a API GPT-3.5 da OpenAI usando as bibliotecas **OkHttp** para requisições HTTP e **Gson** para manipulação de dados JSON.

Ao final deste tutorial, você terá uma aplicação Java capaz de enviar prompts para a API GPT e receber respostas de maneira eficiente, além de processar os dados JSON retornados.

Pré-requisitos

Antes de começar, você precisará de:

- **Chave de API da OpenAI**: Para acessar o GPT-3.5, você precisará de uma chave de API, que pode ser obtida ao se registrar na plataforma da OpenAI.
- **Ambiente Java configurado**: Java 8 ou superior.
- **Dependências externas**: Bibliotecas OkHttp e Gson.

Estrutura do Projeto

O primeiro passo é configurar as dependências do projeto. Vamos utilizar o **Maven** para gerenciar as bibliotecas necessárias.No arquivo pom.xml, adicione as seguintes dependências para **OkHttp** e **Gson**:

xml

```xml
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.9.1</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.8</version>
</dependency>
```

Após configurar o Maven ou outro gerenciador de dependências, estamos prontos para implementar o código Java

Implementação Passo a Passo

1. Armazenamento Seguro de Credenciais

Para evitar expor suas credenciais no código-fonte, vamos utilizar variáveis de ambiente para armazenar a **chave da API** e a **URL da API**.

Essa abordagem melhora a segurança e facilita a manutenção.

java
```
private static final String API_KEY = System.getenv("CHATGPT_API_KEY");
private static final String API_URL = System.getenv("CHATGPT_API_URL");
```

Aqui, utilizamos o método System.getenv() para acessar as variáveis de ambiente.

Antes de executar o projeto, você deverá configurar essas variáveis no seu sistema:

- **No Linux/macOS**: Use o comando export CHATGPT_API_KEY=suachave no terminal.
- **No Windows**: Vá até as variáveis de ambiente do sistema e adicione CHATGPT_API_KEY com o valor da sua chave da OpenAI.

2. Criando o Payload da Requisição

A comunicação com a API do GPT-3.5 envolve o envio de um corpo de requisição JSON contendo o prompt e outras informações, como o modelo.

Para isso, criaremos uma classe interna que represente esse payload:

java
```
static class RequestBodyPayload {
    String model = "gpt-3.5-turbo";
    Message[] messages;

    RequestBodyPayload(String prompt) {
        this.messages = new Message[]{new Message("user", prompt)};
    }

    static class Message {
        String role;
        String content;

        Message(String role, String content) {
            this.role = role;
            this.content = content;
        }
    }
}
```

Esta classe encapsula o modelo utilizado (no caso, gpt-3.5-turbo) e o prompt enviado pelo usuário. A classe Message define o papel do remetente ("user") e o conteúdo da mensagem.

3. Enviando Requisições com OkHttp

Agora que temos o payload da requisição definido, podemos implementar a lógica para enviar a requisição HTTP utilizando o **OkHttp**:

java

```
public static String getChatGPTResponse(String prompt) throws IOException {
    OkHttpClient client = new OkHttpClient();

    // Criando o payload em JSON
    String json = new Gson().toJson(new RequestBodyPayload(prompt));

    // Definindo o corpo da requisição
    RequestBody body = RequestBody.create(
        MediaType.parse("application/json; charset=utf-8"), json
    );

    // Construindo a requisição HTTP
    Request request = new Request.Builder()
        .url(API_URL)
        .post(body)
        .addHeader("Authorization", "Bearer " + API_KEY)
        .addHeader("Content-Type", "application/json")
        .build();

    // Executando a requisição e retornando a resposta
    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Erro na requisição: " + response);
        }
        return response.body().string();
    }
}
```

1. **Criação do cliente HTTP**: O objeto OkHttpClient é responsável por realizar a requisição.
2. **Definição do corpo da requisição (RequestBody)**: Aqui, usamos a biblioteca Gson para converter o objeto Java em JSON e, em seguida, passamos esse JSON no corpo da requisição.
3. **Configuração da requisição (Request.Builder)**: O método post define que estamos enviando uma requisição POST com o corpo da mensagem. Adicionamos os cabeçalhos Authorization com o token da API e o Content-Type como application/json.
4. **Execução da requisição**: A função newCall().execute() realiza a requisição de forma síncrona e retorna a resposta da API.

4. Processando a Resposta da API

Agora que recebemos a resposta em formato JSON, precisamos extrair o conteúdo relevante. Para isso, utilizamos a biblioteca Gson para navegar pela estrutura JSON e capturar a resposta gerada pelo modelo GPT-3.5:

java
```
public static String extractContentFromResponse(String jsonResponse) {
    JsonObject jsonObject = JsonParser.parseString(jsonResponse).getAsJsonObject();
    JsonObject choiceObject = jsonObject.getAsJsonArray("choices").get(0).getAsJsonObject();
    JsonObject messageObject = choiceObject.getAsJsonObject("message");
    return messageObject.get("content").getAsString();
}
```

1. **JsonParser.parseString()**: Converte a string JSON em um objeto JsonObject manipulável.
2. **Navegação pelo JSON**: A resposta da API contém uma lista de opções (choices), e cada opção contém um objeto message com o conteúdo gerado. Navegamos por essa estrutura até extrair o campo "content" que contém a resposta textual do GPT-3.5.

5. Exemplo Completo e Output

O código completo do exemplo seria o seguinte:

java
```
public class ChatGPTIntegration {

    private static final String API_KEY = System.getenv("CHATGPT_API_KEY");
    private static final String API_URL = System.getenv("CHATGPT_API_URL");

    public static void main(String[] args) {
        try {
            String prompt = "O que é Java?";
            String response = getChatGPTResponse(prompt);
            String content = extractContentFromResponse(response);
            System.out.println("Resposta GPT-3.5: " + content);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static String getChatGPTResponse(String prompt) throws IOException {
        OkHttpClient client = new OkHttpClient();
        String json = new Gson().toJson(new RequestBodyPayload(prompt));
        RequestBody body = RequestBody.create(
            MediaType.parse("application/json; charset=utf-8"), json
        );
        Request request = new Request.Builder()
            .url(API_URL)
            .post(body)
            .addHeader("Authorization", "Bearer " + API_KEY)
            .addHeader("Content-Type", "application/json")
```

```
            .build();
        try (Response response = client.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new IOException("Erro na requisição: " + response);
            }
            return response.body().string();
        }
    }

    public static String extractContentFromResponse(String jsonResponse) {
        JsonObject jsonObject = JsonParser.parseString(jsonResponse).getAsJsonObject();
        JsonObject choiceObject =
jsonObject.getAsJsonArray("choices").get(0).getAsJsonObject();
        JsonObject messageObject = choiceObject.getAsJsonObject("message");
        return messageObject.get("content").getAsString();
    }

    static class RequestBodyPayload {
        String model = "gpt-3.5-turbo";
        Message[] messages;

        RequestBodyPayload(String prompt) {
            this.messages = new Message[]{new Message("user", prompt)};
        }

        static class Message {
            String role;
            String content;

            Message(String role, String content) {
                this.role = role;
                this.content = content;
            }
        }
    }
}
```

Exemplo de Output

Ao executar este código com o prompt "O que é Java?", você pode receber uma resposta como a seguinte:

****** Java Netbeans gpt-3.5-turbo ******
Resposta GPT-3.5: Java é uma linguagem de programação de alto nível, orientada a objetos, desenvolvida pela Sun Microsystems em 1995. Ela é amplamente utilizada para o desenvolvimento de aplicações de desktop, web e móveis, além de sistemas distribuídos e em tempo real.

Conclusão

Este guia detalhado mostrou como integrar o GPT-3.5 da OpenAI com Java, utilizando as bibliotecas OkHttp e Gson para gerenciar as requisições HTTP e a manipulação de dados JSON.

Você pode adaptar este exemplo para diferentes prompts e expandir o código para tratar respostas mais complexas, sempre com foco em boas práticas de segurança e organização do código.

EducaCiência FastCode para a comunidade