AI代码评审组件

✓ 背景

随着软件开发的规模和复杂性的不断增长，代码质量变得愈发重要。代码评审（Code Review）作为确保代码质量的关键环节，能够帮助团队识别潜在问题、提高代码的可维护性和可读性。然而，传统的人工代码评审通常需要耗费大量的时间和人力资源，尤其是在大规模项目中，人工评审的效率和覆盖面往往难以满足需求。

近年来，随着人工智能（AI）技术的迅猛发展，AI 已被广泛应用于自动化任务处理领域。特别是在自然语言处理（NLP）和机器学习（ML）领域的进步，使得 AI 能够理解和分析代码的语义、结构，从而能够自动化地执行一些代码评审任务。结合 AI 的代码评审工具，能够自动化地分析代码、识别常见错误、提供最佳实践建议，从而显著提高代码评审的效率与准确性。

目标

本项目旨在开发一个基于人工智能的代码评审组件。该组件能够自动分析代码的质量、结构和潜在问题，并为开发者提供改进建议。

✓ 思路

1.提取git的本次记录和最近记录的差异 交由ai分析, 以模板形式提问和整理结果

2.将分析结果形成日志保存在代码仓库以及通过微信公众号等消息平台通知开发者

3.将上述过程通过如github action 工作流的方式, 当代码提交达到自动审阅的效果

✓ 问题

1.如何对接多种大模型?对接多种消息通知平台?

可以尝试使用策略模式

2.github 还是 gitlab ?

GitHub是开源社区的热门选择，许多知名的开源项目都托管在GitHub上。通过GitHub，你可以与其他开发者协作，提交问题报告、合并请求和讨论代码变更。GitHub提供了易于使用的界面和功能，方便团队协作和代码审查。

通过GitLab，你可以在自己的服务器上搭建托管平台，并具备更大的控制和定制选项。GitLab提供了许多额外的功能，如持续集成、部署管道和自动化测试，使团队能够更高效地协作和交付项目。对于一些组织或项目，安全性可能是最重要的考虑因素之一。在这种情况下，自托管的GitLab实例可以提供更好的控制和保护代码的能力。你可以自行决定代码的访问权限、配置安全设置，并确保代码在你自己的服务器上得到保护。

✓ 扩展

考虑到对接其他大模型, 会存在代码泄露问题, 可以尝试私有化部署大模型, 实现本机或内网进行代码评审

对于大多数新手而言，Ollama是最适合上手的部署框架，而且前段时间也推出了Windows版本的安装包，使用门槛大幅降低。在官网下载并安装完成后就会自动弹出命令提示符，因为Ollama官方就有很多开源大模型的镜像文件，所以只需要输入对应的指令，它就会自动下载对应的大模型，完成后就能进入推理界面开始对话了，整个过程非常直观简单。

Ollama官方地址: **https://ollama.com/blog**

常用命令:

1. ollama list：显示模型列表。
2. ollama show：显示模型的信息
3. ollama pull：拉取模型
4. ollama push：推送模型
5. ollama ps：查看当前模型运行情况
6. ollama cp：拷贝一个模型
7. ollama rm：删除一个模型
8. ollama run：运行一个模型

拉取和运行智普的GLM大大模型命令：ollama run glm4

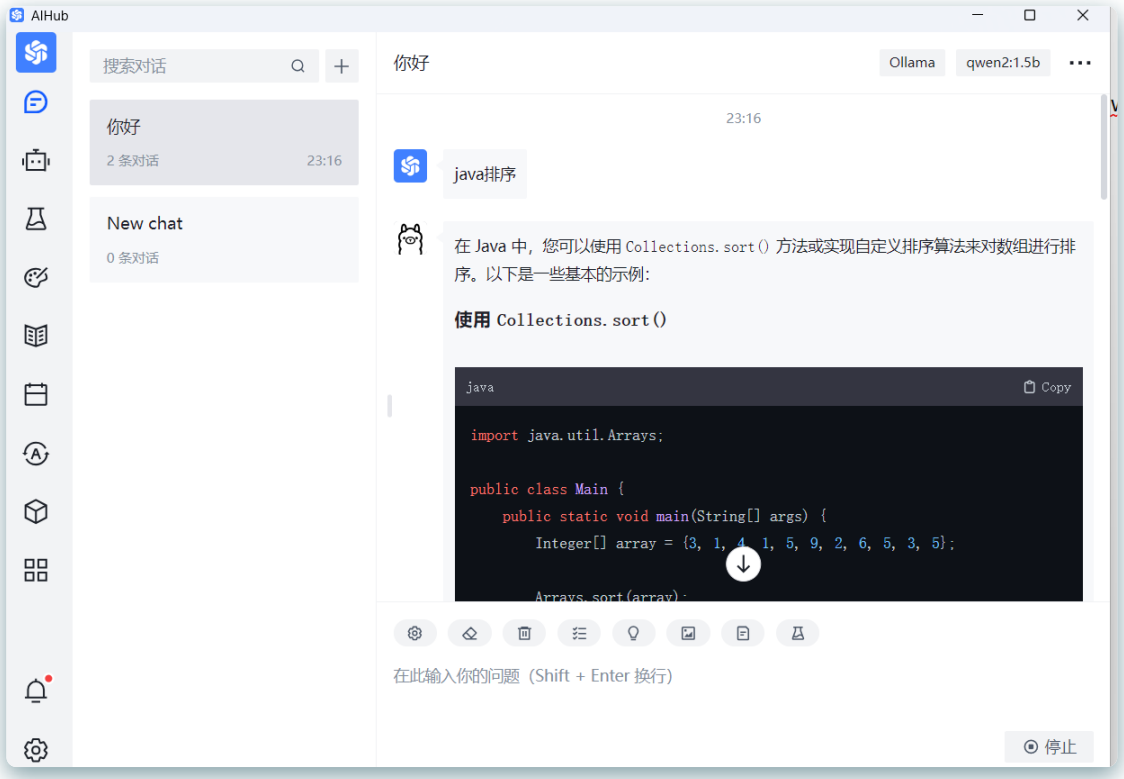
拉取和运行谷歌的开源大模型：ollama run gemma2或ollama run gemma2:9b

拉取和运行阿里开源的通义千问大模型：ollama run qwen2、ollama run qwen2:1.5b、ollama run qwen2:7b

我运行 ollama run qwen2:7b 挺卡的 于是我把它删了 命令 ollama rm qwen2

下载的 qwen2:1.5b

效果图:



客户端推荐: AI Hub

地址: https://github.com/classfang/AIHub/releases/tag/v1.8.9

✓ github action

1. GitHub Actions 是什么?

GitHub Actions 是 GitHub 提供的一种持续集成（CI）和持续部署（CD）服务，它允许开发者自动化他们的软件开发工作流程。通过定义一系列任务（也称为“作业”或“actions”）在特定事件（如代码推送、创建 Pull Request 等）发生时自动触发执行，开发者可以自动测试、构建、打包和部署他们的应用程序。

2. GitHub Actions 有什么用?

GitHub Actions 的主要用途包括：

- **持续集成（CI）**：自动化代码的测试和构建流程，以确保每次提交的代码都能正常编译并通过测试。

- **持续部署（CD）**：在代码通过测试后自动部署应用程序，支持各种部署目标，如服务器、容器、云服务等。
- **自动化开发工作流**: 例如，自动标记版本、发布构建产物、生成文档等。
- **代码质量检查**: 自动执行代码分析工具，检查代码风格、检测潜在的安全漏洞等。
- **跨平台测试**: 通过在不同操作系统（如 Windows、Linux、macOS）上自动测试代码，以确保跨平台兼容性。

3. 如何使用 GitHub Actions?

使用 GitHub Actions 主要分为以下几个步骤：

1. **创建工作流文件**

工作流文件是 GitHub Actions 的核心配置，存放在项目的 `.github/workflows/` 目录下，文件名通常以 `.yml` 或 `.yaml` 结尾。每个工作流文件定义了特定的触发条件、作业和步骤。

2. **编写工作流配置**

以下是一个简单的工作流示例，它会在每次推送代码到主分支时自动运行单元测试：

```yaml
name: CI

# 触发条件：每次推送到 main 分支时
on:
  push:
    branches:
      - main

# 定义作业
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      # 检出仓库代码
      - uses: actions/checkout@v2

      # 设置 Java 版本（如果使用 Java 项目）
      - name: Set up JDK 11
        uses: actions/setup-java@v1
        with:
          java-version: '11'

      # 构建并运行测试
      - name: Build with Gradle
        run: ./gradlew build
```

3. **推送工作流文件到仓库**

将 `.yml` 文件推送到仓库后，GitHub Actions 会根据工作流文件中的配置自动运行相关任务。你可以在 GitHub 仓库的 “Actions” 选项卡中查看工作流的运行状态和日志。

4. **监控与调试**

每次工作流运行后，你可以通过 GitHub 提供的 UI 查看执行结果、日志、以及遇到的错误。如果工作流失败，通常可以在日志中找到详细的错误信息，以帮助调试。

5. **使用现有 Actions**

GitHub Actions 提供了大量的现成 Actions，你可以在工作流中直接使用它们。例如，使用 `actions/checkout` 来检出代码，使用 `actions/setup-node` 来设置 Node.js 环境。

示例：自动化发布流程

以下是一个发布流程的示例，它在创建标签时自动打包并发布项目：

```yaml
name: Release

on:
  push:
    tags:
      - 'v*.*.*'

jobs:
  release:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: Set up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14'

      - run: npm install
      - run: npm run build

      - name: Create Release
        uses: actions/create-release@v1
        with:
          tag_name: ${{ github.ref }}
          release_name: Release ${{ github.ref }}
          body: Release notes here
          draft: false
          prerelease: false
          token: ${{ secrets.GITHUB_TOKEN }}

      - name: Upload Release Asset
        uses: actions/upload-release-asset@v1
        with:
          upload_url: ${{ steps.create_release.outputs.upload_url }}
          asset_path: ./dist/my-app.zip
          asset_name: my-app.zip
          asset_content_type: application/zip
```

资源

- **GitHub Actions 官方文档**: GitHub Actions Documentation

- **市场中的 Actions**: GitHub Marketplace

4.GitLab CI和github action区别?

GitLab CI 更适合深度依赖 GitLab 生态的团队，提供了全面的 CI/CD 功能，适合企业级用户和复杂的 CI/CD 流水线。

GitHub Actions 则以灵活性和易用性著称，特别适合已经在 GitHub 上开发的项目，社区支持强大，适合快速集成各种自动化任务。

#