**FastAPI-сервер для работы с языковой моделью**

**Обзор**

Этот блок кода представляет собой FastAPI-сервер, который взаимодействует с языковой моделью GPT4All. Сервер предоставляет API для генерации текстовых ответов на основе входных сообщений и системного промпта. Поддерживается как синхронная генерация ответов, так и потоковая передача токенов через WebSocket.

**Технологии**

1. **FastAPI**:
   * Используется для создания RESTful API и WebSocket-сервера.
   * Позволяет быстро разрабатывать высокопроизводительные API с автоматической генерацией документации (Swagger/OpenAPI).
2. **GPT4All**:
   * Локальная языковая модель, которая работает на CPU.
   * Используется для генерации текстовых ответов на основе входных данных.
3. **Pydantic**:
   * Используется для валидации входных данных (например, сообщений и системного промпта).
4. **asyncio.Queue**:
   * Используется для организации потоковой передачи токенов через WebSocket.
5. **Uvicorn**:
   * ASGI-сервер для запуска FastAPI-приложения.

**Организация кода**

1. **FastAPI-приложение**:
   * Инициализируется с помощью app = FastAPI().
   * Определены маршруты для обработки HTTP-запросов и WebSocket-соединений.
2. **Модель GPT4All**:
   * Загружается при старте сервера (startup event).
   * Выгружается при завершении работы сервера (shutdown event).
3. **Классы для валидации данных**:
   * Message: Модель для представления сообщения (роль и содержание).
   * ChatRequest: Модель для запроса на генерацию ответа (список сообщений, системный промпт, флаг стриминга).
4. **Генерация ответа**:
   * Функция generate\_response:
     + Формирует историю сообщений для модели.
     + Генерирует ответ с использованием GPT4All.
     + Поддерживает потоковую передачу токенов через Queue.
5. **Маршруты**:
   * **POST /chat**:
     + Принимает запрос на генерацию ответа.
     + Возвращает ответ в синхронном режиме.
     + Если запрошен стриминг, возвращает ошибку (стриминг поддерживается только через WebSocket).
6. **WebSocket (не реализован на данном этапе)**:
   * Планируется для потоковой передачи токенов в реальном времени.

**Как скачать и настроить модель GPT4All**

GPT4All — это локальная языковая модель, которая работает на CPU и не требует мощного GPU. Она идеально подходит для использования на персональных компьютерах. В этом разделе я расскажу, как скачать и настроить модель GPT4All для использования в вашем проекте.

**Шаг 1: Установка GPT4All**

1. **Установите Python**:
   * Убедитесь, что у вас установлен Python версии 3.8 или выше. Скачать его можно с [официального сайта](https://www.python.org/downloads/).
2. **Установите библиотеку GPT4All**:
   * Выполните команду:

|  |
| --- |
| pip install gpt4all |

**Шаг 2: Скачивание модели**

1. **Выберите модель**:
   * GPT4All поддерживает несколько моделей. Вы можете выбрать подходящую модель из списка на официальном сайте gpt4all Например, модель nous-hermes-2-solar-10.7b.Q4\_0.gguf (используется в моей реализации).
2. **Скачайте модель**:
   * Перейдите на страницу модели и скачайте файл .gguf.
   * Например, для модели nous-hermes-2-solar-10.7b.Q4\_0.gguf
3. **Сохраните модель**:
   * Поместите скачанный файл модели в удобную директорию. Например:

|  |
| --- |
| D:/models/nous-hermes-2-solar-10.7b.Q4\_0.gguf |

**Шаг 3: Настройка FastAPI-сервера**

1. **Укажите путь к модели**:
   * В FastAPI-коде укажите путь к скачанной модели:

|  |
| --- |
| model = GPT4All("D:/models/nous-hermes-2-solar-10.7b.Q4\_0.gguf", device="cpu") |