Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 3 | Тема 4 | Урок 3

Работа с API GPT: создание, настройка, подготовка данных

Уважаемый коллега, обратите внимание!

Учёт присутствия ребёнка на уроке ведётся только по цифровому следу. Каждый ученик из группы на уроке должен передать цифровой след на нашей платформе.

Вы отвечаете за передачу следа учеником на нашей платформе во время урока.

Передавать цифровой след нужно в личном кабинете ученика.

Цифровой след засчитывается по итогу выполнения учеником трёх задач на платформе в ходе урока.

Обращаем внимание — в задачах с отправкой ответа в виде файла наша платформа поддерживает следующие форматы: `.pdf, .doc, .png, .jpeg, .xlsx, .mp3, .py, .ipynb, .txt, .csv, .json, .xml, .sb3, .ino, .hex`.

Контролируйте время урока, чтобы не пропустить момент, когда ученики должны залогиниться в личном кабинете и перейти к решению задач на платформе.

Проследите, чтобы каждый ученик залогинился в личном кабинете и выполнил 3 задачи на платформе. В противном случае, Вам не будет засчитано, что вы провели урок, занятие не будет оплачено, а ученику не будет выставлена отметка о присутствии на уроке.

Просим вас в ходе 45 минут урока проконтролировать, чтобы:

1. Каждый из учеников залогинился в личном кабинете.
2. Каждый ученик в разделе **«Решить задания»** отправил решения по 3 задачам — «Практическое задание 1», «Практическое задание 2» и «Практическое задание 3».

О занятии

Краткое содержание занятия:

1. Узнаете, что такое API и API-ключ, и как их использовать.
2. Научитесь настраивать окружение Python и устанавливать необходимые библиотеки.
3. Изучите формат JSON-запроса и параметры генерации.
4. Напишете код для отправки сообщения к GPT.
5. Получите и обработаете ответ от модели.
6. Создадите простое консольное приложение для общения с моделью.

Цель:

Научиться подключаться к API GPT, управлять параметрами генерации и правильно подготавливать данные.

Задачи:

1. Получить и безопасно сохранить API-ключ.
2. Настроить окружение Python: установить пакеты requests, dotenv.
3. Изучить структуру JSON-запроса
4. Написать функцию отправки сообщения и получения ответа.
5. Создать консольное приложение с циклом диалога.

Термины

API (Application Programming Interface)

Интерфейс программирования приложений — набор правил и инструментов для взаимодействия программ друг с другом.

О занятии

API-ключ (токен)

Секретный токен (пароль) для доступа к сервису API. Необходимо хранить в безопасности и не публиковать.

Endpoint

URL-адрес, к которому отправляется запрос для взаимодействия с API.

JSON (JavaScript Object Notation)

Текстовый формат обмена данными, похожий на словарь Python. Используется для передачи запросов и ответов.

Необходимые материалы:

1. Python 3.10
2. Редактор кода (VS Code, PyCharm или другой)
3. Доступ к интернету и API-ключ

Темы и время

ЭТАП	ВРЕМЯ
Приветствие	5 мин.
Теория	25 мин.
Итоги занятия	5 мин.
Получение фидбека. Рефлексия. Проверка знаний. Домашнее задание	5 мин.
Вопрос-ответ	5 мин.
Итого	45 мин.

Ход занятия

Номер слайда	Пояснение к слайду
1	Приветствие учеников. Добрый день, ребята! Сегодня мы продолжаем изучение работы с нейросетями и переходим к практической части — работе с API GPT. Помните, как мы изучали устройство трансформера на прошлом уроке? Сегодня мы научимся не просто понимать, как работает модель, а реально общаться с ней через код. Вы научитесь подключаться к API, отправлять запросы и получать умные ответы. Это как дать команду голосом, только через программу. Не переживайте, если код покажется сложным, мы разберём всё по шагам.
2	На прошлом занятии вы познакомились с внутренней структурой GPT: изучили архитектуру трансформера, механизмы внимания, позиционное кодирование, этапы обучения модели. Также провели практический эксперимент с BertViz. Давайте вспомним: что такое механизм внимания и зачем он нужен? Как модель понимает порядок слов? Эти знания помогут нам лучше понимать, что происходит «под капотом», когда мы отправляем запрос к API. Попросите двух-трёх учеников ответить устно, чтобы активировать знания перед практикой.
3	Сегодня на занятии вы узнаете, что такое API и API-ключ, научитесь настраивать окружение Python и устанавливать библиотеки, изучите формат JSON-запроса и параметры генерации, напишете код для отправки сообщения к GPT, получите и обработаете ответ, создадите простое консольное приложение для общения с моделью. Всё это — реальные навыки разработчика, которые вы сможете использовать в своих проектах. Обратите внимание, что мы не будем сегодня дообучать модель, мы будем использовать уже готовую, подключаясь к ней как пользователи.

Ход занятия

4	Цель и задачи урока. Цель: научиться подключаться к API GPT, формировать корректные запросы и обрабатывать ответы. Задачи: 1) получить и безопасно сохранить API-ключ, 2) настроить окружение Python: установить пакеты requests, dotenv, 3) изучить структуру JSON-запроса: поля name, gpt_version, request, 4) написать функцию отправки сообщения и получения ответа, 5) создать консольное приложение с циклом диалога. Каждая задача — шаг к работающему приложению. Запишите цели в тетрадь, чтобы понимать вектор движения.
5	Что такое API и API-ключ. API — это способ отправить команду к модели напрямую, как отправить сообщение другу. API-ключ работает как пароль: его нельзя показывать другим, нельзя выкладывать в интернет. В нашем коде ключ хранится в переменной окружения, чтобы его не было видно в файле. Пример: HEADERS = {'Authorization': f'Bearer {API_TOKEN}'}. Объясните, что Bearer — это стандартный способ передачи токена, а сам токен — длинная секретная строка, которую выдаёт сервер. Никогда не загружайте файлы с ключами на GitHub! Чтобы получить API ключ необходимо обратиться к провайдеру API или получить его в личном кабинете провайдера после создания проекта
6	Настройка окружения. Для работы потребуется: Python, библиотека requests для отправки запросов, библиотека dotenv для загрузки ключей из файла .env. Установка: pip install requests python-dotenv. Создайте файл .env в папке проекта и запишите туда: API_BASE_URL и API_TOKEN. После этого в коде load_dotenv() автоматически подгрузит эти значения. Проверьте соединение, запустив простой запрос — если нет ошибок, всё настроено верно. Покажите на экране, как выглядит файл .env и как он скрыт от посторонних глаз. env файлы необходим для безопасного хранения данных, скрытия чувствительных данных(Токены, эндпоинты)

Ход занятия

7	Как не превысить лимит и обработать ошибки. Важно знать про таймауты и статусы. В коде мы используем timeout, чтобы программа не зависла навсегда, если сервер молчит. Также нужно проверять статус ответа: если код 200 или 201 — всё хорошо, если 401 — неверный ключ, если 500 — ошибка сервера. Контролируйте количество запросов, чтобы не перегрузить сервер. Объясните, что обработка ошибок — это признак профессионального кода, программа не должна падать просто так.
8	Структура запроса к API. Наш API работает через эндпоинты. Данные отправляются в формате JSON, в котором передаются токен и запроса, который мы написали, указывая необходимую роль для ответа
9	Обработка ответа от API. Сервер возвращает данные в формате JSON. Важно правильно извлечь ответ]. Всегда проверяйте, нет ли в ответе поля errors — если есть, значит, произошла ошибка (неверный токен, недоступная модель и т.д.). Покажите, как вывести результат пользователю красиво, например: print(f"🤖 Модель: {answer}"). Это делает общение более приятным. Объясните, что JSON — это просто текст в фигурных скобках, который легко превратить в словарь Python.
10	Что делает наша программа. Алгоритм работы: читает вопрос из поля ввода по ID userInput, проверяет, не пустой ли ввод, показывает «Загрузка...» и блокирует поле, создаёт чат если ещё не создан, отправляет сообщение через sendMessage, получает ответ и показывает его, в блоке finally разблокирует интерфейс. Важно: обработка ошибок через try/catch позволяет показать понятное сообщение вместо падения страницы. В нашем консольном приложении это цикл while True, который ждёт ввода пользователя пока он не напишет /exit.

Ход занятия

11	Практика: создаём консольное приложение. Задание: 1) Создайте файл app.py и скопируйте шаблон кода. 2) Настройте файл .env с вашим API_BASE_URL и API_TOKEN. (используйте приложение) 3) Запустите приложение: python app.py. 4) Напишите несколько вопросов модели: «Что такое нейросеть?», «Реши задачу: 2+2*2». 5) Проверьте обработку команды /exit. 6) Посмотрите, как меняется chat_id и branch_id при перезапуске. Дайте ученикам 10–15 минут на выполнение, помогайте с ошибками подключения.
12	Итоги занятия. Сегодня вы научились подключаться к API GPT, безопасно хранить API-ключ через .env, формировать JSON-запросы с нужными параметрами, отправлять сообщения через requests.post, обрабатывать JSON-ответы и извлекать текст ответа, создавать консольное приложение с циклом диалога. Вопрос для обсуждения: Как можно улучшить это приложение? Может, добавить историю сообщений или сохранить диалог в файл?
13	Анонс следующего занятия. На следующем уроке вы узнаете о REST и WebSocket, что такое Webhook, создадите Telegram-бота, взаимодействующего с GPT, настроите Webhook для обработки сообщений и проведёте практическую часть: подключите бота к API и протестируете ответы.

Практика

Вопросы для самопроверки

Где можно получить API-ключ?
Почему нельзя публиковать свой ключ?
Зачем использовать .env-файл или переменные окружения?

Задание 1

Заполните шаблон Python-скрипта для работы с GPT.

Шаблон кода:

```
import requests
from dotenv import load_dotenv
load_dotenv()
API_BASE_URL = os.getenv('API_BASE_URL')
API_TOKEN = os.getenv('API_TOKEN')
Отправляем запрос к API
....

def main():
    app = AIConsole()
    app.run()
if __name__ == '__main__':
    main()
```

Примеры выполнения заданий

Проверь себя по следующим критериям:

- Обратиться к провайдеру API или в личном кабинете у провайдера API
- Это приведёт к несанкционированному использованию, утечке данных, финансовым потерям и блокировке аккаунта.
- Чтобы хранить секреты (типа API-ключей) отдельно от кода и не коммитить их в репозиторий.
  Чёткая инструкция/вопрос и ожидаемый ответ (или завершение), соответствующие целевой задаче (формат зависит от модели). Модель корректно решает целевую задачу на тестовых данных, метрики (точность, loss) улучшились, поведение соответствует ожиданиям.
- JSONL: каждая строка — отдельный JSON-объект (гибкий, подходит для сложных структур). CSV: табличный формат с разделителями (простой, но плохо подходит для вложенных данных).

Задание 1

Как выполнять задание:

```
import requests

import os

from dotenv import load_dotenv

from telegram import Update

from telegram.ext import Application, MessageHandler, filters, ContextTypes

load_dotenv()

API_KEY = os.getenv("OPENROUTER_API_KEY")

MODEL = os.getenv("OPENROUTER_MODEL",
"qwen/qwen3-vl-30b-a3b-thinking")

TG_TOKEN = os.getenv("TELEGRAM_BOT_TOKEN")
```

Примеры выполнения заданий

```
if not API_KEY or not TG_TOKEN:

    print("❌ Укажите OPENROUTER_API_KEY и TELEGRAM_BOT_TOKEN в
.env")

    exit(1)

print(f"✓ Бот запущен (модель: {MODEL})")

messages = {}  # Хранение истории по user_id

async def handle_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):

    user_id = update.effective_user.id

    text = update.message.text

    if user_id not in messages:

        messages[user_id] = []

    messages[user_id].append({"role": "user", "content": "отвечай на русском"
+ text})

    await update.message.reply_text("🤖 Печатает...")

    response = requests.post(

        "https://openrouter.ai/api/v1/chat/completions",

        headers={

            "Authorization": f"Bearer {API_KEY}",

            "Content-Type": "application/json"

        },
```

Примеры выполнения заданий

```
        json={
            "model": MODEL,
            "messages": messages[user_id]
        },
        timeout=120
    )
    assistant_message = response.json()["choices"][0]["message"]["content"]
    messages[user_id].append({"role": "assistant", "content":
assistant_message})
    await update.message.reply_text(assistant_message)

app = Application.builder().token(TG_TOKEN).build()

app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

app.run_polling()
```

Сегодня на занятии вы:

- Научились подключаться к API GPT.
- Безопасно хранили API-ключ через .env.
- Формировали JSON-запросы с нужными параметрами.
- Отправляли сообщения через requests.post.
- Обрабатывали JSON-ответы и извлекали текст.
- Создали консольное приложение с циклом диалога.

На следующем занятии вы:

- Поймёте принципы REST и WebSocket для передачи данных.
- Узнаете, что такое Webhook и как его настроить.
- Создадите Telegram-бота, взаимодействующего с GPT.
- Настроите Webhook для обработки сообщений.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока.

Ответы к интерактивным задачам

Практическое задание 1

Как называется секретный токен для доступа к сервису API?

API-ключ, API-key

Практическое задание 2

Какой формат используется для отправки данных через API?

- CSV
- XML
- **JSON**
- TXT

Практическое задание 3

Что необходимо для успешной отправки запроса к модели?

- **API-ключ**
- **Использование метода POST**
- Обязательное использование метода GET
- **JSON файл**

Ответы к интерактивным задачам

Вопрос-ответ

Если во время подготовки или проведения урока возникают вопросы, их необходимо адресовать в методический чат.