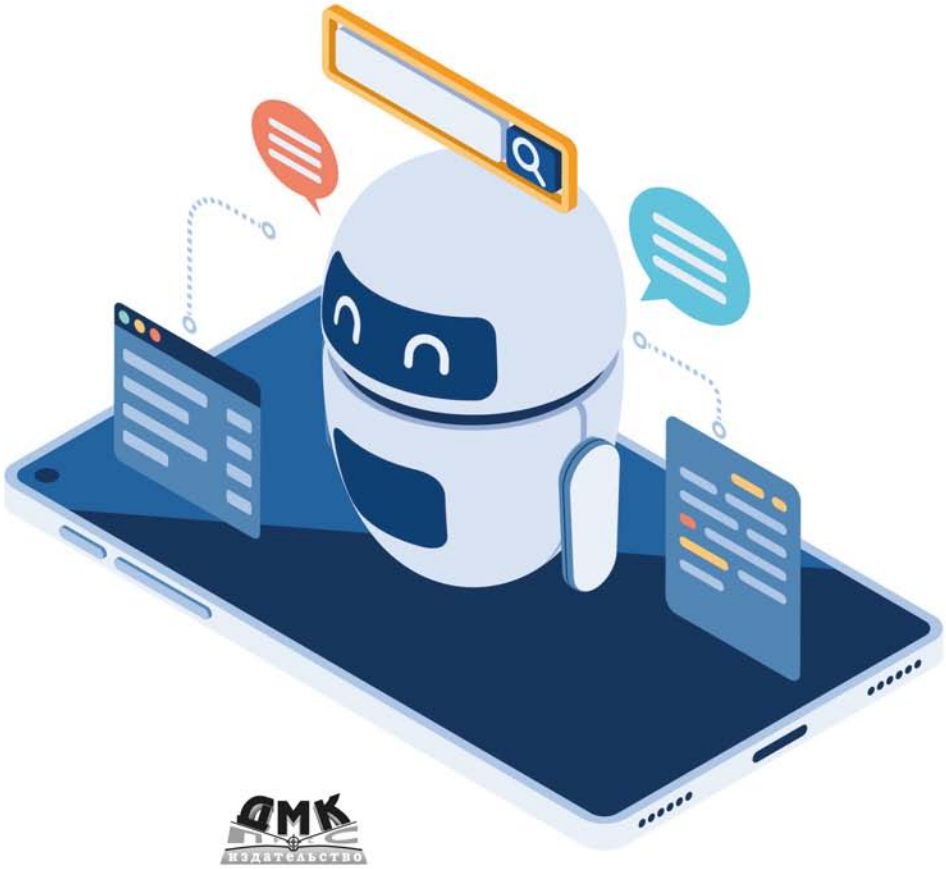
Аймен Эль Амри

GPT-4

Руководство по использованию API Open AI



ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

УДК 004.4
ББК 32.371
А62

Аймен Эль Амри
А62 GPT-4. Руководство по использованию API Open AI / пер. с англ. В. С. Яценкова. – М.: ДМК Пресс, 2024. – 274 с.: ил.

ISBN 978-5-93700-299-0

В книге рассказывается о том, как использовать генеративные текстовые модели поколений GPT-3.5 и GPT-4 для создания приложений различного назначения, в числе которых интерактивный психотерапевт, интеллектуальный голосовой помощник, система рекомендации товаров, генератор заметок в соцсетях, система распознавания речи и многие другие. Вы научитесь использовать векторные базы данных, узнаете, как управлять уровнем креативности моделей GPT, применять современные методы генерирования высококачественного текста, и даже организуете диалог между двумя чат-ботами. Примеры и практические упражнения помогут закрепить пройденный материал.

Издание предназначено для тех, кто владеет основами языка программирования Python и собирается использовать GPT в реальных сценариях для решения прикладных задач.

УДК 004.4
ББК 32.371

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-5-93700-299-0

© Aymen El Amri 2024
© Оформление, издание, перевод, ДМК Пресс, 2024

Оглавление

Предисловие от издательства 9

Предисловие 10

Об авторе 11

История OpenAI и ChatGPT 12

Об этой книге 15
Оставайтесь на связи 16

Как работает GPT? 17

Подготовка среды разработки 20

Важные примечания 20
Установка Python, pip и виртуальной среды для разработки 21
Получение ключа API OpenAI 22
Установка официальных средств интеграции Python 23
Тестирование ключей API 23

Доступные модели и выбор оптимального варианта 26

Модели OpenAI и важные соглашения 26
Какую модель лучше использовать? 28
Серии моделей OpenAI 29
Серия GPT-4 29
Серия GPT-3.5 30
Серия InstructGPT-3 31
Базовая серия GPT-3 32
Серия Codex 33
Content Filter 34
Серия DALL-E 34
Серия TTS 35
Модель Whisper 35
Модель встраивания 36
Модели и цены OpenAI 36
Что дальше? 38

Использование функции завершения 39

Вводный пример 39
Роли system, user и assistant 42
Роль system 42

Роль user 43
Роль assistant 43
Завершение чата и обучение на нескольких примерах 43
Форматирование вывода 45
Ограничение количества выходных токенов 48
Управление остановкой завершения 50
Температура и галлюцинации 55
Параметр top_p 58
Что выбрать – temperature или top_p? В чем разница? 60
Потоковая передача ответа API 61
Управление повторяемостью: presence_penalty и frequency_penalty 62
Что штрафовать – частоту или наличие? 64
Управление количеством результатов через API 65
Заключение 66

Продвинутые примеры и разработка промптов 67

Что такое разработка промптов? 67
Обучение на нескольких примерах: основной метод разработки промптов 69
Избыточная генерация и выбор лучшего варианта 74
Генерация знаний по запросу: создание песни в стиле рэп 79
Что такое Apple – фрукт или компания? 82
Динамическое управление количеством токенов 88
Создание интерактивного помощника в окне командной строки 92
Что дальше? 101

Встраивание 103

Что такое встраивание? 103
Варианты применения: от поисковых систем до беспилотных автомобилей 103
Tesla: применение встраиваний в беспилотных автомобилях 104
Kalendar AI: применение встраиваний в управлении продажами 104
Notion: расширенные возможности поиска 104
DALL-E 2: преобразование текста в изображение 104
Изучаем встраивание текста 105
Встраивания для нескольких входов 107
Пример применения: семантический поиск 107
Что такое косинусное подобие 108
Семантический поиск и встраивание текста OpenAI 112
За кулисами: как работает встраивание 123

Продвинутые примеры встраивания 125

Рекомендация подходящего сорта кофе 125
Разработка более «нечеткого» поиска 137
Прогнозирование категории новостей: классификация с помощью встраивания 141
Оценка точности классификатора 146
Точность приложений классификатора в различных сценариях 151

Тонкая настройка и передовые методы работы 153
Обучение на ограниченных примерах 153
Улучшенное обучение на ограниченных примерах 154
Практическое применение тонкой настройки 154
Полезные приемы тонкой настройки 159
Выбор модели 159
Проверка набора данных 159
Максимальное количество токенов 169
Размер набора данных 169
Тестирование и улучшение обучения (гиперпараметры) 169
Количество эпох 169
Коэффициент скорости обучения 170
Размер пакета 171
Ориентировочная оценка затрат 171
Качество набора данных 173
Экспериментируйте и учитесь 174
Используйте проверочные наборы данных 174
Тестирование модели 175
Анализ результатов 175

Продвинутый пример тонкой настройки: виртуальный консультант 177
Набор данных, используемый в примере 177
Подготовка данных 179
Проблемы использования модели в реальных приложениях 187

Контекст и память: как сделать искусственный интеллект более реалистичным 189
В чем проблема? 189
Отсутствие контекста = хаос случайности 189
История = контекст 191
Недостатки переноса контекста через историю 193
Память «последним вошел – первым вышел» (LIFO) 193
Проблема с памятью типа LIFO 196
Избирательный контекст 196

Применение векторной базы данных 204
Введение 204
Что такое векторная база данных? 204
Пример 1. Использование Weaviate для повышения контекстной зависимости модели 206
Пример 2. Семантический поиск с помощью Weaviate и OpenAI 219
Пример 3. Генеративный поиск с помощью Weaviate и OpenAI 226

Распознавание и перевод речи с Whisper 236
Что такое Whisper? 236
С чего начать? 238
Распознавание и перевод речи 239
Использование Whisper SDK в коде Python 240

Использование API OpenAI для преобразования аудиозаписи в текст 242
API распознавания 242
API перевода 243
Улучшение качества распознавания речи с Whisper 244
Очистка аудиозаписи 244
Использование подсказки 244
Постобработка полученного текста 246

Преобразование текста в речь 248

Диалог между двумя ИИ на основе OpenAI и Weaviate 251
Генерация аудиофайлов 251
Использование аватаров модели 268
Что дальше? 271

Послесловие 272

Предметный указатель 273

Как работает GPT?

GPT – это *генеративная* текстовая модель. Такая модель способна создавать новый текст, предсказывая его продолжение на основе полученных входных данных.

GPT-4 – это просто обновленная и увеличенная модель, она заметно крупнее и производительнее, чем любая другая предыдущая модель GPT, включая GPT-1/2/3/3.5.

Модель четвертого поколения обучали на большом массиве текстов, таких как книги, статьи и общедоступные веб-сайты, такие как Reddit и другие форумы. Она использует эти обучающие данные для изучения закономерностей и взаимосвязей между словами и фразами.

Ключевое отличие GPT-4 заключается в ее впечатляющем размере – с ошеломляющими 1,76 трлн параметров, – что делает ее одной из самых массивных и мощных языковых моделей, когда-либо созданных человеком. Сочетание невиданного ранее количества параметров и обширного набора данных позволяет модели генерировать правдоподобные тексты и выполнять различные задачи обработки естественного языка с впечатляющим качеством.

GPT – это тип нейронной сети, относящейся к архитектуре Transformer, которую специально разработали для задач обработки естественного языка. В повседневной жизни модели с такой архитектурой часто называют просто *трансформерами*. Архитектура Transformer основана на последовательности блоков самовнимания, которые позволяют модели параллельно обрабатывать входной текст и взвешивать важность каждого слова или токена в зависимости от контекста.

Самовнимание (self-attention) – это механизм, используемый в моделях глубокого обучения для *обработки естественного языка* (natural language processing, NLP), который позволяет модели взвешивать важность различных частей предложения или нескольких предложений при прогнозировании. Являясь частью архитектуры Transformer, он дает возможность нейронной сети достигать высокого качества работы, когда речь идет о задачах NLP. Концепция самовнимания основана на идее, что на каждое слово в предложении могут влиять другие слова в предложении и важность каждого слова можно определить на основе его контекста. Впервые механизм самовнимания был представлен в статье «Внимание – это все, что вам нужно»[1].

Представьте себе, что вы присутствуете на большом званом обеде, где все одновременно делятся историями. Вы пытаетесь следить за всеми разговорами вокруг вас, но, естественно, больше сосредоточены на историях, которые имеют к вам какое-то отношение или содержат важную информацию. Когда люди говорят, вы уделяете свое внимание рассказчику, чье повествование в данный момент наиболее интересно или актуально, иногда переключая

[1] *https://arxiv.org/abs/1706.03762*.

фокус на основе новой информации или связей, которые вы устанавливаете с другими рассказываемыми историями.

В этой аналогии вы похожи на модель трансформера в GPT. Званый ужин представляет собой входной текст, где история каждого гостя представляет собой слово или токен в последовательности. Ваша способность слушать несколько историй одновременно и решать, на какой из них сосредоточиться, аналогична механизму самовнимания. Этот механизм позволяет модели *одновременно* обрабатывать все части входных данных и определять релевантность или важность каждого слова в контексте всех остальных, подобно тому, как вы в нужный момент настраиваетесь на самую увлекательную историю.

Вот пример использования трансформеров, разработанных компанией Hugging Face для вывода GPT-2:

```
from transformers import pipeline
generator = pipeline(
    'text-generation',
    model = 'gpt2'
)
generator(
    "Привет, я языковая модель",
    max_length = 30,
    num_return_sequences=3
)
```

Так выглядит пример вывода предыдущего кода:

```
[
    {
        'generated_text': "Привет, я языковая модель. Когда я писала это, ..."
    },
    {
        'generated_text': " Привет, я языковая модель. Я пишу и поддерживаю ..."
    }
]
```

По умолчанию модель не имеет памяти, это означает, что каждый ввод обрабатывается независимо, без переноса какой-либо информации из предыдущих вводов. Когда GPT генерирует текст, у нее *нет априорных представлений* о том, что должно быть дальше, на основе предыдущих входных данных. Вместо этого она генерирует каждое слово на основе *вероятности* того, что оно будет следующим словом с учетом предыдущего ввода. Поэтому полученный текст иногда бывает неожиданным и не совсем осмысленным.

Вот еще один пример кода, использующего модель GPT для генерации текста на основе пользовательского ввода.

```
# Импорт необходимых библиотек
from transformers import GPT2Tokenizer, GPT2LMHeadModel

# Загрузка предварительно обученного токенизатора GPT-2 и модели
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')
```

```
# Использование модели в режиме оценки
model.eval()

# Определение начального запроса, за которым следует завершение
prompt = input("You: ")

# Токенизация запроса и генерация текста
input_ids = tokenizer.encode(prompt, return_tensors='pt')
output = model.generate(input_ids, max_length=50, do_sample=True)

# Декодирование сгенерированного текста и вывод в консоль
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
print("AI: " + generated_text)
```

GPT-4 целенаправленно разрабатывали как языковую модель общего назначения, поэтому ее можно использовать для различных задач обработки естественного языка, таких как языковой перевод, резюмирование текста и ответы на вопросы. Разработчики из OpenAI предварительно обучили GPT-4, но вы можете создать собственную модель с помощью процедуры тонкой настройки на собственных наборах данных. Это позволяет применять модель для более конкретных творческих задач в дополнение к задачам по умолчанию, таким как сочинение текста, стихов и рассказов. Настроенные модели также можно использовать для создания чат-ботов, которые являются экспертами в определенной области, диалоговых интерфейсов и многого другого!

В этой книге будет подробно рассказано о моделях OpenAI (GPT, встраивание и многие другие понятия) и о том, как обращаться к ним из кода на Python, включая эффективное использование API и инструментов, предоставляемых компанией и сообществом AI/ML, для создания прототипов мощных и креативных инструментов и систем. Это не только улучшит ваши навыки применения API GPT, но и расширит ваше понимание концепций и методов, используемых в обработке естественного языка и других смежных областях.

Благодаря огромному количеству параметров и впечатляющим результатам работы GPT-3 считается значительным достижением в обработке естественного языка. Однако некоторые эксперты выражают обеспокоенность по поводу того, что модель может создавать предвзятый или вредный контент. Как и в случае с любой технологией, здесь очень важно вовремя остановиться и подумать об этических последствиях ее использования. Но в этой книге мы не будем касаться этических вопросов, а сосредоточимся только на технических аспектах.

Приятного чтения!

Подготовка среды разработки

Важные примечания

В этом руководстве для создания файлов и вставки контента непосредственно из командной строки или с помощью скриптов и программ часто используется так называемый *heredoc-синтаксис*.

> Heredoc, аббревиатура от «here document» (здесь документ), представляет собой функцию сценариев оболочки, которая упрощает создание многострочного ввода (строк с переносом).

Heredoc-синтаксис начинается с символов <<, за которыми следует маркер-разделитель по вашему выбору, продолжается содержимым, которое вы хотите включить, и завершается тем же маркером-разделителем на новой строке. Этот синтаксис особенно полезен для задач сценариев и программирования, требующих создания файлов или ввода нескольких строк для команды.

Чтобы использовать heredoc-синтаксис для создания файлов или добавления содержимого, используйте следующую структуру:

```
cat  <<  EOF  >  имя_файла
Это строка  #1
Это строка  #2
..
EOF
```

В этом примере `<<EOF` означает начало раздела heredoc, а `EOF` в отдельной строке указывает на его конец. Содержимое между этими маркерами перенаправляется в файл с именем `имя_файла`. Замените `имя_файла` на имя файла, которое вам нужно, и при необходимости измените содержимое.

Основная цель использования синтаксиса heredoc в этом руководстве – дать читателям возможность точно и легко создавать файлы с помощью одного действия копирования и вставки из электронной версии книги. Такой подход гарантирует, что контент автора будет точно воспроизведен в среде читателя, что снижает вероятность ручных ошибок и делает процесс обучения более эффективным.

Я предлагаю копировать примеры heredoc из электронной версии книги прямо в ваш терминал командной строки, чтобы создать файлы и контент, как это задумано в книге[1].

Пользователи Windows могут выполнять примеры heredoc, применяя подсистему Windows для Linux (Windows Subsystem for Linux, WSL). Кроме того, вы можете вручную создать файлы и вставить их содержимое с помощью текстового редактора. Тем не менее для выполнения примеров в этой книге я рекомендую использовать Unix-подобную среду (предпочтительно систему на базе Debian).

---

[1] Читатели русского перевода могут скачать архив с готовыми примерами кода на сайте издательства. В таком случае heredoc-синтаксис не нужен. – *Прим. перев.*

Если вы работатете с другой системой, вам может потребоваться настроить некоторые незначительные детали, такие как менеджер пакетов, процесс установки пакетов и метод создания файлов и каталогов. Это не должно стать проблемой, поскольку примеры просты и могут быть легко адаптированы к любой системе.

Еще одно важное замечание, о котором стоит упомянуть, заключается в том, что это руководство построено таким образом, что каждая глава основывается на предыдущей, а темы и концепции переплетаются между собой, чтобы сформировать всестороннее понимание предмета. Важно последовательно проходить главы, заботясь о том, чтобы полностью усвоить каждый раздел, прежде чем переходить к следующему. Терпение и внимание к деталям существенно помогут вашему обучению, позволяя вам в полной мере извлечь выгоду из пошагового развития.

Код, представленный в этом руководстве, предназначен не для пассивного просмотра, а для активного выполнения. Взаимодействие с кодом – его запуск, изменение и наблюдение за результатами – важно для глубокого понимания обсуждаемых концепций. Этот практический подход поможет вам усвоить материал и развить навыки, необходимые для применения знаний в ваших собственных проектах.

Установка Python, pip и виртуальной среды для разработки

Очевидно, что вам понадобится Python. В этой книге мы будем использовать Python 3.9.5.

Мы также будем работать с установщиком пакетов pip. Вы можете использовать pip для установки пакетов из репозитория пакетов Python и других репозиториев.

Вам не нужно устанавливать Python 3.9.5 непосредственно в вашей системе, так как мы создадим виртуальную среду разработки. Какая бы у вас ни была установлена версия Python на рабочем компьютере, среда разработки будет изолирована от вашей системы и будет использовать Python 3.9.5.

Если Python не установлен, перейдите на страницу *www.python.org/downloads/*, загрузите и установите одну из версий Python 3.x. В зависимости от вашей операционной системы вам придется следовать разным инструкциям.

Для управления нашей средой разработки мы будем использовать virtualenvwrapper[1]. Инструкции по установке вы найдете в официальной документации[2].

Самый простой способ получить его – использовать pip:

```
pip install --upgrade pip
pip install virtualenv
pip install virtualenvwrapper
```

Пользователи Windows могут применять следующие команды для установки virtualenvwrapper-win[3]:

[1] *https://github.com/python-virtualenvwrapper/virtualenvwrapper*.

[2] *https://virtualenvwrapper.readthedocs.io/en/latest/install.html*.

[3] *https://github.com/davidmarble/virtualenvwrapper-win/*.

```
pip install --upgrade pip
pip install virtualenv
pip install virtualenvwrapper-win
```

Примечание: если вы привыкли к virtualenv, Poetry, Conda или любому другому диспетчеру пакетов, то можете продолжать его использовать. В этом случае нет необходимости устанавливать virtualenvwrapper.

Если pip не установлен, проще всего установить его с помощью скрипта `get-pip.py`[1]:

```
curl https://bootstrap.pypa.io/get-pip.py | python3
```

Пользователи Windows могут ввести следующие команды:

```
curl https://bootstrap.pypa.io/get-pip.py
py get-pip.py
```

В итоге в вашей системе должны быть установлены следующие пакеты:

- Python 3.9;
- pip;
- virtualenv;
- virtualenvwrapper (или любой другой менеджер пакетов, который вы предпочитаете).

Я настоятельно рекомендую пользователям Windows создать виртуальную машину с операционной системой Debian Linux, так как большинство примеров, представленных в этой книге, выполнялись и тестировались в системе Debian Linux. Хотя мнигие команды и примеры кода будут работать под Windows, в некоторых случаях могут потребоваться небольшие изменения.

Далее создадим виртуальную среду:

```
mkvirtualenv -p python3.9 chatgptforpythondevelopers
```

После создания виртуальной среды активируйте ее:

```
workon chatgptforpythondevelopers
```

Получение ключа API OpenAI

Следующим шагом является создание ключей API, которые позволят вам получить доступ к официальному API, предоставляемому OpenAI[2].

[1] *https://bootstrap.pypa.io/get-pip.py*.

[2] Сайт и сервисы OpenAI недоступны для пользователей с российским IP-адресом. Для создания учетной записи вам понадобится доступ в интернет с европейским или американским IP и временный виртуальный телефонный номер в любой европейской стране, на который вы получите СМС с кодом подтверждения регистрации. В дальнейшем вам этот телефонный номер не понадобится, но доступ к API с российских IP-адресов невозможен. Мы не будем здесь детально описывать использование виртуальных телефонных номеров, но отметим, что российским пользователям доступно множество сервисов, предоставляющих виртуальные телефонные номера для получения СМС в различных странах с оплатой российскими банковскими картами, включая «Мир», по очень доступной цене. Как показал опыт, для регистрации и входа на сайт OpenAI можно использовать российский аккаунт Google. Процедура регистрации на сайте OpenAI детально описана в различных блогах российских авторов.

Перейдите по адресу *https://platform.openai.com* и создайте учетную запись[1].

Ключ API должен принадлежать организации, поэтому вам будет предложено создать организацию. В этой книге мы будем называть ее LearningGPT.

Сохраните сгенерированный секретный ключ в надежном и доступном месте. Вы не сможете увидеть его снова через свою учетную запись OpenAI.

Установка официальных средств интеграции Python

Вы можете взаимодействовать с API, используя HTTP-запросы с любого языка через официальные средства интеграции (binding, привязка) Python, или через официальную библиотеку Node.js, или через библиотеку, поддерживаемую сообществом.

В этой книге мы будем использовать официальную библиотеку, предоставленную OpenAI. Другой альтернативой является применение Chronology[2], неофициальной библиотеки, предоставленной OthersideAI. Однако похоже, что эта библиотека больше не обновляется, поэтому не рекомендуется использовать ее для производственных целей.

Чтобы установить официальные средства интеграции Python, выполните следующую команду:

```
pip install openai==1.9.0
```

Убедитесь, что вы устанавливаете библиотеку в виртуальной среде, которую мы создали ранее.

Тестирование ключей API

Прежде всего объявим ключ API и ID организации как переменные окружения (вместо `xxx` подставьте ваши данные):

```
export API_KEY=xxx
export ORG_ID=xxx
```

Теперь выполните следующую команду для тестирования доступа к API (вводить без переноса строки):

```
curl \
https://api.openai.com/v1/models \
-H 'Authorization: Bearer '$API_KEY'' \
-H 'OpenAI-Organization: '$ORG_ID''
```

Если в вашей учетной записи OpenAI определена только одна организация, вы можете выполнить ту же команду без указания ID организации:

```
curl \
https://api.openai.com/v1/models \
-H 'Authorization: Bearer '$API_KEY''
```

Команда должна вернуть список моделей, доступных через API, таких как davinci, ada и многих других.

---

1 *https://platform.openai.com/api-keys*.

2 *https://github.com/OthersideAI/chronology*.

Далее мы выполним тест с применением Python SDK.

Создайте файл с именем `.env` и сохраните в нем ключ API и ID организации:

```
cat << EOF > .env
API_KEY=$API_KEY
ORG_ID=$ORG_ID
EOF
```

Для тестирования работы в Python SDK выполните следующий код:

```
cat << EOF > test_api.py
import os
from openai import OpenAI
from pprint import pprint

# чтение переменных из файла .env,
# а именно API_KEY и ORG_ID.
with open(".env") as env:
    for line in env:
        key, value = line.strip().split("=")
        os.environ[key] = value

# Инициализация ключа API и ID организации
client = OpenAI(
  api_key=os.environ['API_KEY'],
  organization=os.environ['ORG_ID']
)

# Печать объекта client
pprint(vars(client))
EOF
```

Запустите приведенный выше код, выполнив команду

```
python test_api.py
```

Мы будем использовать аутентификацию с помощью ключа API и ID организации в каждом примере кода на протяжении всей книги. Поэтому нам будет полезно упорядочить код.

Создайте папку с именем `src` и внутри нее создайте файл с именем `api.py`:

```
cat << EOF > src/api.py
import os
from openai import OpenAI

# Чтение переменных из файла .env,
# а именно API_KEY и ORG_ID.
with open("src/.env") as env:
    for line in env:
        key, value = line.strip().split("=")
        os.environ[key] = value

# Создание объекта client на уровне модуля
client = OpenAI(
```

```
  api_key=os.environ['API_KEY'],
  organization=os.environ['ORG_ID']
)
```

Создайте файл `.env` при помощи следующей команды:

```
cat << EOF > src/.env
API_KEY=$API_KEY
ORG_ID=$ORG_ID
EOF
```

Для тестирования доступа к API необходимо выполнить следующий код:

```
cat << EOF > src/test_api_1.py
from api import client
from pprint import pprint

# Печать объекта client
    pprint(vars(client))
EOF
```

Запустите этот код при помощи команды:

```
python src/test_api_1.py
```

Вы должны увидеть тот же вывод, что и при запуске предыдущего кода.

Доступные модели и выбор оптимального варианта

Модели OpenAI и важные соглашения

Обратившись к конечной точке API `models`, вы получите список доступных моделей.

```
cat << EOF > src/test_api_2.py
from api import client

models = client.models.list()
for model in models:
    print(vars(model))
EOF
```

Запустите этот код:

```
python src/test_api_2.py
```

Вы должны получить вывод наподобие следующего:

```
{
    'id': 'curie-search-query',
    'created': 1651172509,
    'object': 'model',
    'owned_by': 'openai-dev'
}

{
    'id': 'babbage-search-query',
    'created': 1651172509,
    'object': 'model',
    'owned_by': 'openai-dev'
}

{
    'id': 'dall-e-3',
    'created': 1698785189,
    'object': 'model',
    'owned_by': 'system'
}
... и т. д.
```

Инструмент разработки OpenAI Python SDK интуитивно понятен и прост в использовании. Нам достаточно применить метод `list()`, чтобы получить список всех доступных моделей.

Выведем на печать только ID моделей:

```
cat << EOF > src/test_api_3.py
from api import client

models = client.models.list()
for model in models:
    print(model.id)
EOF
```

Запустите код:

```
python src/test_api_3.py
```

Вывод должен содержать список всех доступных моделей; он будет выглядеть примерно так:

```
dall-e-3
dall-e-2
GPT-4
davinci-002
babbage-002
whisper-1
GPT-3.5-Turbo-16k
GPT-3.5-Turbo
GPT-3.5-Turbo-Instruct
```

Модели Babbage-002 и Davinci-002 являются частью серии GPT-3, разработанной OpenAI, и служат заменой или обновлением предыдущих моделей:

- Babbage-002. Служит заменой базовым моделям GPT-3 Ada и Babbage. Она оптимизирована для задач, требующих баланса между производительностью и экономической эффективностью, и подходит для задач средней сложности, таких как классификация текста, семантический поиск и базовое понимание языка;
- Davinci-002. Заменяет базовые модели GPT-3 Curie и Davinci. Известна своими расширенными возможностями понимания и генерации языков и идеально подходит для более сложных задач, требующих детального понимания, таких как создание контента, сложный языковой перевод и сложный логический вывод;
- DALL-E 3 и DALL-E 2. Обе являются моделями генерации изображения. DALL-E 3 представляет собой усовершенствованную версию базовой модели DALL-E 2 с более продвинутыми возможностями создания реалистичных изображений высокого разрешения на основе текстовых описаний;
- Whisper-1. Эта модель специализируется на распознавании и транскрипции речи. Она была разработана для точного преобразования устной речи в письменный текст. В будущем вы можете увидеть более продвинутые версии этой модели (например, Whisper-2, Whisper-3 и т. д.);
- GPT-3.5-Turbo и GPT-3.5-Turbo-Instruct. Обе эти модели представляют собой варианты GPT-3.5, предназначенные для более эффективного и оперативного создания текста;
- GPT-4. Это большая мультимодальная модель, которая может принимать как текст, так и изображения. Она является преемником GPT-3 и способна выполнять широкий спектр задач.

Некоторые из моделей, перечисленных в выводе, являются динамическими, поскольку они постоянно обновляются со стороны OpenAI. Например, названия GPT-3.5-Turbo, GPT-4 и GPT-4-32k указывают на последнюю версию модели. На момент написания книги последней версией модели GPT-4 была GPT-4-0613, поэтому когда ваш код вызывает конечную точку API GPT-4, вы фактически используете GPT-4-0613. В будущем, когда OpenAI выпустит новую модель GPT-4, конечная точка API будет указывать на новую модель. Вот почему рекомендуется указывать в коде конкретную версию модели, если у вас есть особые требования.

Исторически модели назывались «движками», но OpenAI отказалась от термина «движок» в пользу «модели». В некоторых публикациях эти термины используют как взаимозаменяемые, но правильное название – «модель». При использовании API запросы, использующие старые имена, по-прежнему будут работать, поскольку OpenAI обеспечила обратную совместимость, но в будущем это может измениться. Однако наименования некоторых моделей следует вручную заменить в коде на более новые. Рекомендуется использовать обновленные модели, чтобы избежать проблем в будущем.

Некоторые модели не упомянуты в перечне выше, например *модели модерации*. Эти модели используются для обнаружения нежелательного контента в тексте и его фильтрации. Применение этих моделей не является обязательным, но настоятельно рекомендуется, если вы создаете общедоступное приложение. Вы ведь не хотите, чтобы ваши пользователи получали оскорбительные или незаконные результаты?

Какую модель лучше использовать?

На этот вопрос нет однозначного ответа. Выбор зависит от вашей задачи и типа приложения, которое вы создаете. Как правило, вам следует выбирать самую последнюю доступную модель. Например, с точки зрения функциональности, если вы создаете чат-бот, вам следует использовать модель GPT-3.5-Turbo-Instruct. Если вы создаете приложение, генерирующее изображения из текста, вам следует использовать модель DALL-E 3. Если вашему приложению требуются дополнительные возможности, вам следует использовать модель GPT-4.

Второй фактор, который следует учитывать, – это цена. Некоторые модели дороже других. Например, GPT-3.5-Turbo – самая экономичная модель серии GPT-3.5.

Еще одним важным фактором является контекстное окно. У некоторых моделей контекстное окно больше, чем у других. Например, модель GPT-4-32k имеет контекстное окно из 32 768 токенов (около 50 страниц англоязычного текста), а модель GPT-4-0613 имеет контекстное окно из 8192 токенов (около 12 страниц англоязычного текста). Это означает, что модель GPT-4-32k может обрабатывать больше текста, чем модель GPT-4-0613.

В следующих разделах мы подробнее рассмотрим основные доступные модели; это поможет вам понять различия между ними и выбрать подходящую модель для вашего применения. Однако важно понимать, что некоторые из перечисленных ниже моделей устарели и, вероятно, не будут доступны в будущем. OpenAI постоянно совершенствует свои модели и отключает старые.

Серии моделей OpenAI

Серия GPT-4

GPT-4 – это новейшая серия моделей, разработанная OpenAI, которая является преемником GPT-3. Одно из основных отличий от GPT-3 заключается в том, что модели GPT-4 *мультимодальные*. Это означает, что GPT-4 может принимать как изображения, так и текстовые входные данные.

> Мультимодальное глубокое обучение обогащает модели информацией, собранной из различных модальностей данных, таких как текст, изображения, видео, аудио и данные датчиков. Это ставит его на шаг впереди традиционного машинного обучения, которое было ограничено одной модальностью, что позволяет решить проблему сложности, связанной с реальными данными, в которых часто смешаны различные источники. Цель мультимодального глубокого обучения – создать единое пространство представления, собирая разнообразную информацию из этих модальностей и применяя ее для решения таких задач, как создание подписей к изображениям, понимание речи и обработка естественного языка.

GPT-4 демонстрирует качество работы человеческого уровня по нескольким профессиональным и академическим показателям. Примечательно, что GPT-4 вошла в 10 % лучших экзаменуемых во время имитации экзамена на адвоката, что является значительным улучшением по сравнению с GPT-3.5, которая вошла в нижние 10 % по оценке за экзамен, согласно сообщению в блоге[1], опубликованному OpenAI.

GPT-4 доступна как для разработчиков, использующих API, так и для обычных пользователей, применяющих ChatGPT.

В семействе GPT-4 есть две основные модели:

- стандартная модель GPT-4 (8k) с длиной контекста 8192 токена;
- расширенная модель GPT-4 (32k) с длиной контекста 32 768 токенов (около 50 страниц текста).

Однако для конкретных случаев использования доступны и другие модели, например GPT-4-Turbo – модель, которая принимает изображения в качестве входных данных в API продолжения чата.

В следующей таблице суммированы основные различия этих моделей.

Название модели	Количество параметров, млрд	Описание	Длина контекста	Рабочий статус
GPT-4	1000*	Стандартная модель GPT-4 с контекстом длиной 8192 токена	8192	Доступна
GPT-4-32k	1000*	Расширенная модель GPT-4 с контекстом длиной 32 768 токенов	32 768	Доступна

[1] *https://openai.com/research/gpt-4*.

Окончание табл.

Название модели	Количество параметров, млрд	Описание	Длина контекста	Рабочий статус
GPT-4-Turbo	1000*	Модель, принимающая на вход изображения в API продолжения чата	128 000	Предпросмотр

* Количество параметров по неофициальной оценке Semafor[1].

Серия GPT-3.5

GPT-3.5 – это более мощная версия базовой модели GPT-3. Она способна выполнять большее количество задач и имеет больше параметров, чем базовая GPT-3.

Серия GPT-3.5 представляет три модели:

- text-davinci-002: предлагает возможности, аналогичные text-davinci-003, но использует управляемую тонкую настройку вместо обучения с подкреплением;
- text-davinci-003: блестяще справляется с языковыми задачами, обеспечивая превосходное качество, расширенные результаты и последовательное соблюдение инструкций по сравнению с моделями Curie, Babbage или Ada. Она также поддерживает встроенные дополнения в тексте;
- GPT-3.5-Turbo: самая продвинутая модель GPT-3.5, предназначенная для взаимодействия в чате. Ее использование обходится лишь в 1/10 стоимости применения text-davinci-003.

В следующей таблице приведены основные различия между этими тремя моделями.

Название модели	Количество параметров, млрд	Описание	Длина контекста	Рабочий статус
text-davinci-002	–	Имеет те же функции, что и text-davinci-003, но обучается по другому методу, ориентированному на обучение с учителем	–	Устарела, заменена на GPT-3.5-Turbo-Instruct (требуется замена вручную в коде)
text-davinci-003	–	Превосходно справляется с языковыми задачами, обеспечивая более длинные, последовательные и высококачественные результаты. Также поддерживает внутритекстовое завершение	–	Устарела, заменена на GPT-3.5-Turbo-Instruct (требуется замена вручную в коде)

[1] *https://www.semafor.com/article/03/24/2023/the-secret-history-of-elon-musk-sam-altman-and-openai.*

Окончание табл.

Название модели	Количество параметров, млрд	Описание	Длина контекста	Рабочий статус
GPT-3.5-Turbo	175	Самая совершенная модель GPT-3.5, оптимизированная для чата и доступная за небольшую часть стоимости text-davinci-003	4096	Доступна
GPT-3.5-Turbo-Instruct	175	Самая совершенная модель GPT-3.5, оптимизированная для чата и доступная за небольшую часть стоимости text-davinci-003	4096	Доступна
GPT-3.5-Turbo-16k	175	Самая совершенная модель GPT-3.5, оптимизированная для чата и доступная за небольшую часть стоимости text-davinci-003	16 384	Доступна

Серия InstructGPT-3

Серия InstructGPT-3 превосходит базовую модель GPT-3 по части следования инструкциям. Она состоит из четырех различных моделей:

- text-ada-001: созданная для выполнения базовых задач, она зачастую является самой быстрой и экономичной в линейке GPT-3;
- text-babbage-001: эффективна для простых задач, обеспечивает высокую производительность и доступность;
- text-curie: высококлассная, обеспечивает более быстрые результаты по цене ниже, чем у Davinci;
- text-davinci-001: более ранняя версия модели высшего уровня в линейке GPT-3, способная выполнять задачи, которые могут выполнять другие модели GPT-3, обычно с ограниченным контекстом.

Модели InstructGPT, как правило, менее склонны к болтовне и более ориентированы на задачи, что делает их идеальными для выполнения четких и лаконичных заданий. Это может сэкономить время и использование токенов, особенно если задача проста и контекст ясен.

В следующей таблице приведены основные различия между этими моделями.

Название модели	Количество параметров, млрд	Описание	Рабочий статус
text-ada-001	0.35	Подходит для базовых задач, обычно самая быстрая и доступная в линейке GPT-3	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление кода вручную)

Окончание табл.

Название модели	Количество параметров, млрд	Описание	Рабочий статус
text-babbage-001	175	Модельный ряд GPT-3. Эффективно решает прямые задачи с меньшими затратами	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление кода вручную)
text-curie-001	6.7	Высококлассная, но более быстрая и экономичная, чем Davinci	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление кода вручную)
text-davinci-001	175	Более ранняя версия модели высшего уровня в линейке GPT-3. Способна решать задачи, аналогичные другим моделям GPT-3, но может потребовать большего контекста	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление кода вручную)

Базовая серия GPT-3

Базовая серия GPT-3 состоит из четырех моделей (A, B, C, D), каждая из которых отличается от остальных по скорости и качеству.

- Ada (A): подходит для простых задач, обычно самая быстрая и экономичная в линейке GPT-3;
- Babbage (B): выполняет элементарные задачи быстро и с меньшими затратами;
- Curie (C): высокая эффективность, более высокое качество при более низкой цене, чем у Davinci;
- Davinci (D): вершина серии GPT-3, превосходящая по качеству результаты работы других моделей.

Базовые модели GPT-3 способны понимать человеческий язык и генерировать текст, который зачастую неотличим от созданного человеком. Они также способны выполнять большое количество задач, в том числе:

- классификацию текста;
- семантический поиск;
- обобщение текста;
- анализ настроений;
- генерацию контента;
- завершение текста;
- перевод текста;
- извлечение текста.

Однако эти модели не самые точные в серии GPT-3.

- Ada и Babbage устарели и заменены на Babbage-002. Обновления выполняются автоматически OpenAI;

- Curie устарела и заменена на Davinci-002. Обновления выполняются автоматически OpenAI;
- Text-Ada, Text-Babbage и Text-Davinci устарели и заменены на GPT-3.5-Turbo-Instnict. Обновления должны выполняться разработчиком вручную.

В следующей таблице суммированы основные различия между моделями.

Название модели	Количество параметров, млрд	Описание	Рабочий статус
ada	0.35	Подходит для базовых задач, обычно самая быстрая и доступная в линейке GPT-3	Устарела и заменена на babbage-002 (автоматическое обновление)
babbage	1.3	Эффективно и с меньшими затратами справляется с прямыми задачами	Устарела и заменена на babbage-002 (автоматическое обновление)
curie	6.7	Высококлассная, но более быстрая и экономичная, чем Davinci	Устарела и заменена на davinci-002 (автоматическое обновление)
davinci	175	Вершина моделей GPT-3, превосходно справляется с задачами, которые могут выполнить другие, часто с превосходными результатами	Устарела и заменена на davinci-002 (автоматическое обновление)
babbage-002	1.3	Служит заменой базовым моделям GPT-3 Ada и Babbage	Доступна
davinci-002	175	Заменяет базовые модели GPT-3 Curie и Davinci	Доступна

Серия Codex

Серия Codex была анонсирована в середине 2021 г. Это потомок GPT-3, который также был обучен на коде из 54 млн репозиториев GitHub и поддерживает инструмент автодополнения кода GitHub Copilot.

Серия Codex состоит из двух моделей:

- code-davinci-002;
- code-cushman-001.

Модели серии Codex объявлены устаревшими в марте 2023 г. и заменены на GPT-3.5-Turbo. Следующая таблица представлена здесь исключительно для исторических целей.

Название модели	Описание	Статус
code-davinci-002	Самая эффективная модель Codex. Отлично справляется с переводом естественного языка в код и поддерживает вставку дополнительных элементов	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление вручную)
code-cushman-001	Мощная и быстрая, подходит для многих задач генерации кода, доступнее, чем Davinci	Устарела и заменена на GPT-3.5-Turbo-Instruct (требуется обновление вручную)

Content Filter

Как следует из названия, это фильтр конфиденциального контента.

Используя этот фильтр, вы можете обнаружить конфиденциальный или небезопасный текст, сгенерированный API. Этот фильтр может классифицировать текст по трем категориям:

- безопасный (safe);
- чувствительный (sensitive);
- небезопасный (unsafe).

Если вы создаете приложение, которое будут использовать внешние пользователи, вы можете наложить на вывод модели фильтр контента, чтобы определить, возвращает ли она какой-либо неприемлемый контент.

На момент написания доступны две модели:

- text-moderation-stable;
- text-moderation-latest.

В следующей таблице приведены основные различия между ними.

Название модели	Описание	Длина контекста	Статус
text-moderation-stable	Стабильная версия фильтра контента	32 768	Доступна
text-moderation-latest	Последняя версия фильтра контента, более высокая точность	32 768	Доступна

Серия DALL-E

Это семейство моделей, которые могут генерировать изображения из текстовых описаний. Они продолжают линию ImageGPT и базируются на принципах GPT-3. В серию DALL-E входят две модели: DALL-E 2 и DALL-E 3.

В следующей таблице приведены основные различия между моделями.

Параметр	DALL-E 2	DALL-E 3
Разрешение, пикс.	512×512	1024×1024
Метод синтеза изображения	Дискретный вариационный автокодировщик	Диффузионная модель

Окончание табл.

Параметр	DALL-E 2	DALL-E 3
Интеграция с ChatGPT	Нет прямой интеграции	Интегрирована с ChatGPT
Следование сложным промптам	Иногда неправильно интерпретирует или пропускает элементы	Отлично выполняет сложные промпты
Генерация текста по изображению	Часто генерирует запутанный, нерелевантный текст	Генерирует качественный, соответствующий контексту текст
Прорисовка деталей человеческого тела	Затрудняется прорисовывать детали изображения человека	Хорошо прорисовывает такие детали, как лица и руки
Привлекательность изображений	Генерирует стандартные изображения	Производит креативные, эмоционально окрашенные изображения
Управление безопасностью	Несколько мер безопасности	Расширенные настройки безопасности для предотвращения некорректного контента
Классификатор происхождения	Эта функция отсутствует	Содержит классификатор происхождения
Контроль творчества	Может нарушать авторские права	Соблюдает права реальных художников
Доступность	В ранних версиях	Доступна в ChatGPT Plus и корпоративным клиентам через API, запланированы следующие релизы
Параметры	3.5 млрд	–

Серия TTS

В дополнение к моделям продолжения чата и генерации изображений OpenAI также предлагает серию моделей преобразования текста в речь (text-to-speech, TTS). В этой серии две модели: TTS-1 и TTS-l-HD. Они представлены в следующей таблице.

Название модели	Описание	Статус
tts-1	Модель, оптимизированная для скорости	Доступна
tts-1-hd	Модель, оптимизированная для качества	Доступна

Модель Whisper

Whisper[1] – это модель распознавания речи с открытым исходным кодом, которая может преобразовывать устную речь в письменный текст. Эта модель

[1] *https://github.com/openai/whisper.*