**Multi-Modal Systems with the OpenAI API**

**EXERCISES**

1. **Beyond Text Generation**

**Creating a podcast transcript**

The OpenAI API **Audio** endpoint provides access to the different models, which can be used for speech-to-text transcription and translation. In this exercise, you'll create a transcript from a **[DataFramed podcast](https://www.datacamp.com/podcast" \t "_blank)** episode with OpenAI Developer, Logan Kilpatrick.

If you'd like to hear more from Logan, check out the full [**ChatGPT and the OpenAI Developer Ecosystem**](https://www.datacamp.com/podcast/chat-gpt-and-the-open-ai-developer-ecosystem) podcast episode.

**Instructions**

**100 XP**

* Open the openai-audio.mp3 file.
* Create a transcription request to the Audio endpoint with audio\_file.
* Extract and print the transcript text from the response.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Open the openai-audio.mp3 file

audio\_file = open("openai-audio.mp3", "rb")

# Create a transcript from the audio file

response = client.audio.transcriptions.create(model="whisper-1", file=audio\_file)

# Extract and print the transcript text

print(response.text)

Hi there, Logan, thank you for joining us on the show today. Thanks for having me. I'm super excited about this. Brilliant. We're going to dive right in, and I think ChatGPT is maybe the most famous AI product that you have at OpenAI, but I'd just like to get an overview of what all the other AIs that are available are. So I think two and a half years ago, OpenAI released the API that we still have available today, which is essentially our giving people access to these models. And for a lot of people, giving people access to the model that powers ChatGPT, which is our consumer-facing first-party application, which essentially just, in very simple terms, puts a nice UI on top of what was already available through our API for the last two and a half years. So it's sort of democratizing the access to this technology through our API. If you want to just play around with it, as an end user, we have ChatGPT available to the world as well.

In [1]:

**Transcribing a non-English language**

OpenAI's audio models can not only transcribe English speech but also perform well in speech in many other languages.

In this exercise, you'll create a transcript from audio.m4a, which contains speech in Portuguese.

**Instructions**

**100 XP**

* Open the audio.m4a file in **r**ead-**b**inary mode.
* Create a transcription request to the Audio endpoint.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Open the audio.m4a file

audio\_file = open("audio.m4a", "rb")

# Create a transcript from the audio file

response = client.audio.transcriptions.create(model="whisper-1", file=audio\_file)

print(response.text)

<script.py> output:

Olá, o meu nome é Eduardo, sou CTO no Datacamp. Espero que esteja a gostar deste curso que o James e eu criamos para você. Esta API permite enviar um áudio e trazer para inglês. O áudio original está em português.

In [1]:

**Translating Portuguese**

OpenAI's audio models can not only transcribe audio into its native language, but also support translation capabilities for creating English transcriptions.

In this exercise, you'll return to the Portuguese audio, but this time, you'll translate it into English!

**Instructions**

**100 XP**

* Open the audio.m4a file.
* Create a translation request to the Audio endpoint.
* Extract and print the translated text from the response.

**script.py**

**Translating Portuguese**

OpenAI's audio models can not only transcribe audio into its native language, but also support translation capabilities for creating English transcriptions.

In this exercise, you'll return to the Portuguese audio, but this time, you'll translate it into English!

**Instructions**

**100 XP**

* Open the audio.m4a file.
* Create a translation request to the Audio endpoint.
* Extract and print the translated text from the response.
* <script.py> output:
* Hello, my name is Eduardo, I am a CTO at Datacamp. I hope you are enjoying this course that James and I have created for you. This API allows you to send an audio and bring it to English. The original audio is in Portuguese.
* In [1]:

**OpenAI's text-to-speech (TTS)**

OpenAI now provide models for creating human-like speech from a text input, so-called text-to-speech or **TTS**. *OpenAI* provide several voices to choose from, and they provide the ability to stream the response to local files or downstream applications.

**Instructions**

**100 XP**

* Create the text-to-speech request for "Hi! How's your day going?", using the "ballad" voice.
* Stream the response to an .mp3 file.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Create the text-to-speech request

response = client.audio.speech.create(

  model="gpt-4o-mini-tts",

  voice="ballad",

  input="Hi! How's your day going?"

)

# Stream the response to an MP3 file

response.stream\_to\_file("output.mp3")

**TTS in other languages!**

Let's have a go at some non-English text. Try sending the following text input to the model:

Dnes je krásný slunečný den.

The text is in the *Czech* language, which is spoken primarily in the *Czech Republic*.

**Instructions**

**100 XP**

* Send the Czech text provided to the gpt-4o-mini-tts model using the "coral" voice.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Pass the non-English text to the model

response = client.audio.speech.create(

    model="gpt-4o-mini-tts",

    voice="coral",

    input="Dnes je krásný slunečný den."

)

response.stream\_to\_file("output.mp3")

**Why use text moderation models?**

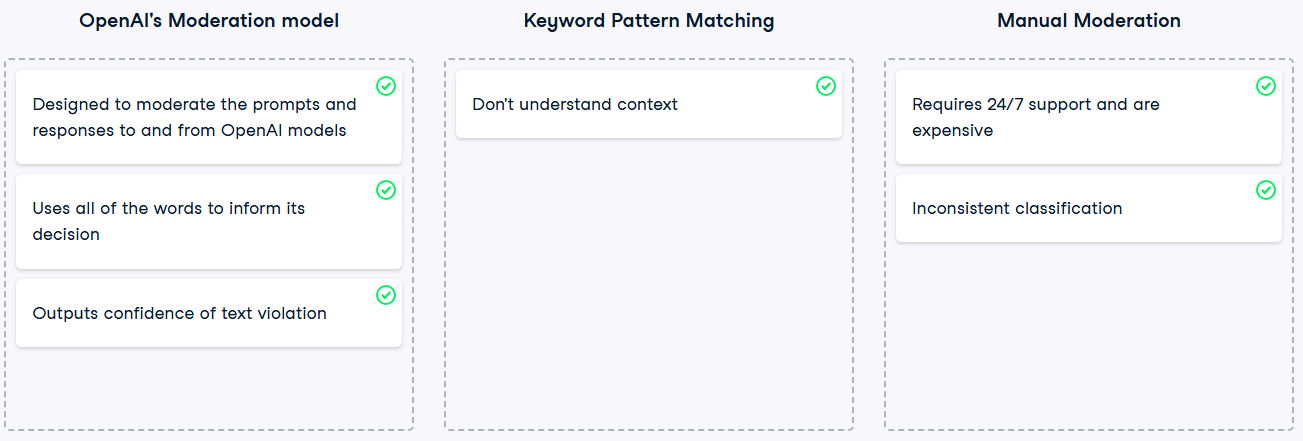
Text moderation is a vital component of most social media platforms, internet chatrooms, and many other user-facing systems. It serves the purpose of preventing the distribution and promotion of inappropriate content, such as hate speech.

In this exercise, you'll compare OpenAI's text moderation model to traditional methods of moderation: manual moderation and keyword pattern matching.

**Instructions**

**100XP**

* Classify the statements as being properties of OpenAI's moderation model, keyword pattern matching, or manual moderation.



**Requesting moderation**

Aside from text and chat completion models, OpenAI provides models with other capabilities, including text *moderation*. OpenAI's text moderation model is designed for evaluating prompts and responses to determine if they violate OpenAI's usage policies, including inciting hate speech and promoting violence.

In this exercise, you'll test out OpenAI's moderation functionality on a sentence that may have been flagged as containing violent content using traditional word detection algorithms.

**Instructions**

**100 XP**

* Check if "My favorite book is To Kill a Mockingbird." violates OpenAI’s policies using the Moderations endpoint.
* Print the category scores to see the results.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Create a request to the Moderation endpoint

response = client.moderations.create(

    model="text-moderation-latest",

    input="My favorite book is To Kill a Mockingbird."

)

# Print the category scores

print(response.results[0].category\_scores)

<script.py> output:

CategoryScores(harassment=5.243551186140394e-06, harassment\_threatening=1.1516095810293336e-06, hate=4.767837526742369e-05, hate\_threatening=3.2021056028952444e-08, illicit=None, illicit\_violent=None, self\_harm=9.466615438213921e-07, self\_harm\_instructions=5.426785065765216e-08, self\_harm\_intent=1.5536235764557205e-07, sexual=3.545879735611379e-06, sexual\_minors=1.1304399549771915e-06, violence=0.0001064608441083692, violence\_graphic=1.086988686438417e-05, self-harm=9.466615438213921e-07, sexual/minors=1.1304399549771915e-06, hate/threatening=3.2021056028952444e-08, violence/graphic=1.086988686438417e-05, self-harm/intent=1.5536235764557205e-07, self-harm/instructions=5.426785065765216e-08, harassment/threatening=1.1516095810293336e-06)

In [1]:

**Examining moderation category scores**

The same request you created in the last exercise to the Moderation endpoint has been run again, sending the sentence "My favorite book is To Kill a Mockingbird." to the model. The response from the API has been printed for you, and is available as response.

Extract the category scores to determine the correct interpretation from the following list of statements.

**Instructions**

**50 XP**

**Possible answers**

The model believes that the sentence contains violent content, as the violence category is close to 0.

The model believes that there are no violations, as all categories are close to 0.

The model believes that the sentence contains hate speech, as the hate category is close to 0.

ModerationCreateResponse(id='modr-C9894Z1vHPCm7vwOM6W8w7YOiUBHg', model='text-moderation-007', results=[Moderation(categories=Categories(harassment=False, harassment\_threatening=False, hate=False, hate\_threatening=False, illicit=None, illicit\_violent=None, self\_harm=False, self\_harm\_instructions=False, self\_harm\_intent=False, sexual=False, sexual\_minors=False, violence=False, violence\_graphic=False, self-harm=False, sexual/minors=False, hate/threatening=False, violence/graphic=False, self-harm/intent=False, self-harm/instructions=False, harassment/threatening=False), category\_applied\_input\_types=None, category\_scores=CategoryScores(harassment=5.243551186140394e-06, harassment\_threatening=1.1516095810293336e-06, hate=4.767837526742369e-05, hate\_threatening=3.2021056028952444e-08, illicit=None, illicit\_violent=None, self\_harm=9.466615438213921e-07, self\_harm\_instructions=5.426785065765216e-08, self\_harm\_intent=1.5536235764557205e-07, sexual=3.545879735611379e-06, sexual\_minors=1.1304399549771915e-06, violence=0.0001064608441083692, violence\_graphic=1.086988686438417e-05, self-harm=9.466615438213921e-07, sexual/minors=1.1304399549771915e-06, hate/threatening=3.2021056028952444e-08, violence/graphic=1.086988686438417e-05, self-harm/intent=1.5536235764557205e-07, self-harm/instructions=5.426785065765216e-08, harassment/threatening=1.1516095810293336e-06), flagged=False)])

In [1]:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Case Study: Building a Customer Support Chatbot**

**Transcribing customer calls**

You're working as an AI Engineer at *DataCamp*, which has an international learner base spanning 180+ countries. Their customer support team want to test receiving customer queries via phone call; however, they need your support to design a system to help their agents handle these requests.

To start, you will use the OpenAI API to transcribe their customer calls in the language they were recorded in.

***As this is a case study, less code will be provided for you than in the previous chapter. If you struggle, don't give up, you can do this!***

**Instructions**

**100 XP**

* Open the customer recording stored in the file customer\_call.wav.
* Create a transcription request to the Audio endpoint and extract the transcript text.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Open the customer\_call.wav file

audio\_file = open("customer\_call.wav", "rb")

# Create a transcript from the audio file

response = client.audio.transcriptions.create(

    model="whisper-1",

    file=audio\_file

)

transcript = response.text

print(transcript)

<script.py> output:

Ciao! Vorrei imparare l'AI con DataCamp, ma cosa posso usare? PyTorch, l'API di OpenAI, Lungchain o qualcos'altro?

In [1]:

**Deriving the customer's language**

Now that you have a transcript of the customer call, you need to derive the language used to eventually translate the response back to the customer in their native language. You'll use an OpenAI chat model to identify the language.

The output from the model should be standardized to a country code; for example, "en" for English, "fr" for French, and so on.

The transcript you created in the previous exercise is available as transcript.

**Instructions**

**100 XP**

* Prompt gpt-4o-mini to consider the transcript and return the country code of the language being used, e.g., "en" for English and "fr" for French.
* Extract the country code from response.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Complete a request identify the country code from the transcript

response = client.chat.completions.create(

    model="gpt-4o-mini",

    max\_completion\_tokens=5,

    messages=[{

        "role": "user",

        "content": f"""Identify the language of the following text and respond only with the country code (e.g., 'en', 'sp', 'fr'): {transcript}"""}]

)

# Extract the country code from the response

country\_code = response.choices[0].message.content

print(country\_code)

<script.py> output:

It

**Translating the transcript into English**

With the customer call transcript created, it's time to translate it into English for processing.

The transcript you created previously is available as transcript, and the country code is stored in country\_code.

**Instructions**

**100 XP**

* Translate the text in transcript into English.
* Extract the transcript text from response.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Translate the transcript into English

response = client.chat.completions.create(

    model="gpt-4o-mini",

    max\_completion\_tokens=300,

    messages=[{

        "role": "user",

        "content": f"""Translate this customer transcript from country code {country\_code} to English: {transcript}"""

    }]

)

# Extract the translated transcript text

en\_transcript = response.choices[0].message.content

print(en\_transcript)

<script.py> output:

Sure! The translation from Italian to English is:

"Hello! I would like to learn AI with DataCamp, but what can I use? PyTorch, the OpenAI API, Langchain, or something else?"

In [1]:

**Fixing translation mistakes**

As you saw, translations aren't always perfect, particularly if the transcript contains names or references outside of the model's knowledge. In this exercise, you'll use another call to the chat model to fix these incorrect words.

The English transcript you created in the previous exercise is available as en\_transcript.

**Instructions**

**0 XP**

* Create a prompt that fixes the technical names and terminology in the English transcript, en\_transcript, *leaving the rest of the transcript unchanged*.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Fix the mistake in the transcript

response = client.chat.completions.create(

    model="gpt-4o-mini",

    messages=[{"role": "user", "content": f"You are an AI assistant that corrects transcripts by fixing names and terminology, leaving the rest of the transcript unchanged. Please refine the following transcript: {en\_transcript}"}]

)

corrected\_text = response.choices[0].message.content

print(corrected\_text)

<script.py> output:

Hello! I would like to learn AI with DataCamp, but what can I use? PyTorch, the OpenAI API, LangChain, or something else?

**Moderating the customer call**

To ensure quality and safety, you'll now moderate the processed customer message. This step helps identify and flag potentially inappropriate content such as harassment.

The corrected transcript from the previous exercise is available as corrected\_text.

**Instructions**

**100 XP**

* Create a moderation request using the text-moderation-latest model with corrected\_text as input.
* Extract response's category scores and convert them to a dictionary using .model\_dump().
* Extract the harassment score from the scores dictionary and check if it is *greater than* 0.8.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Send the moderation request

response = client.moderations.create(

    model="text-moderation-latest",

    input=corrected\_text

)

# Extract scores and convert to dictionary

scores = response.results[0].category\_scores.model\_dump()

# Extract harassment score

harassment\_score = scores['harassment']

if harassment\_score > 0.8:

    print("Content flagged for harassment!")

else:

    print("Content is safe from harassment.")

<script.py> output:

Content is safe from harassment.

**Generating a tailored response**

Now that the customer inquiry passed moderation, it's time to generate a tailored response.

To help the chatbot respond accurately, you'll provide it with relevant context. Two documents are available:

* faqs: common customer questions and answers
* content\_overview: relevant tracks, including descriptions and links

The corrected transcript is still available as corrected\_text

**Instructions**

**100 XP**

* Complete the prompt that gives clear instructions and includes the faqs and content\_overview documents for context.
* Use the gpt-4o-mini model to generate a response, sending instruction\_prompt as a system message and corrected\_text as a user message.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Complete the prompt

instruction\_prompt = f"""

#### Role

You are a professional AI support assistant for DataCamp. You help with:

- Sales (pricing, plans, billing)

- Content (courses, recommendations, feedback)

- Marketing (partnerships, collaborations)

#### How to Respond

1. Use the FAQs: {faqs}

2. Use the content overview: {content\_overview}

3. Respond clearly and concisely in up to 3 paragraphs.

4. If unsure, direct the user to support@datacamp.com.

"""

# Generate response

response = client.chat.completions.create(

    model="gpt-4o-mini",

    messages=[

        {"role": "system", "content": instruction\_prompt},

        {"role": "user", "content": corrected\_text}

    ]

)

chatbot\_reply = response.choices[0].message.content

print(chatbot\_reply)

<script.py> output:

At DataCamp, you have various options to explore AI depending on your interests and focus areas. Here are some tracks that might be of interest:

1. \*\*Associate AI Engineer for Developers\*\*: This career track focuses on integrating AI into software applications using Python. You'll work with large language models (LLMs), prompt engineering, and tools like the OpenAI API, Hugging Face, LangChain, and Pinecone. You can check it out [here](https://www.datacamp.com/tracks/associate-ai-engineer-for-developers).

2. \*\*OpenAI Fundamentals\*\*: This skills track teaches you how to use the OpenAI API in Python for building AI applications such as chatbots and recommendation systems. You'll engage in hands-on coding exercises to enhance your skills. Explore this track [here](https://www.datacamp.com/tracks/openai-fundamentals).

3. \*\*Deep Learning in Python\*\*: If you're interested in deep learning specifically, this track dives into using PyTorch for building models addressing real-world tasks like language identification and cloud classification. Check it out [here](https://www.datacamp.com/tracks/deep-learning-in-python).

These tracks cover a variety of tools and technologies in AI, allowing you to choose based on what aligns best with your learning goals.

In [1]:

**Moderating the model response**

Even though the chatbot has generated a tailored response, it's important to verify that the content is safe to send to the customer. In this step, you'll moderate the response to ensure all category scores fall below the customer safety threshold.

The model's reply from the previous exercise is available as chatbot\_reply.

**Instructions**

**100 XP**

* Send a moderation request using the text-moderation-latest model with chatbot\_reply as input.
* Extract the response's category scores, convert them to a dictionary using .model\_dump(), and check if any score *exceeds* 0.7.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Send the moderation request

response = client.moderations.create(

    model="text-moderation-latest",

    input=chatbot\_reply

)

# Extract scores and convert to dictionary

scores = response.results[0].category\_scores.model\_dump()

if all(score > 0.7 for score in scores.values()):

    print("AI Response flagged for moderation!")

    chatbot\_reply = """I'm sorry, but I can't provide a response to that request. Please contact support@datacamp.com for further assistance."""

else:

    print("AI Response is safe.")

<script.py> output:

AI Response is safe.

**Response translation**

You now have a tailored response from the chatbot that passed moderation. Your next task is to translate that response into the original language of the customer request.

The model's reply and the detected country code from the previous exercises are available as chatbot\_reply and country\_code.

**Instructions**

**100 XP**

* Use the gpt-4o-mini model to translate chatbot\_reply into the language specified by country\_code.
* Extract the translated text from response.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Translate chatbot reply

response = client.chat.completions.create(

    model="gpt-4o-mini",

    messages=[

        {"role": "system", "content": f"""Translate the following text from English to country code {country\_code}. Only return the translated text!"""},

        {"role": "user", "content": chatbot\_reply}

    ],

    max\_completion\_tokens=500

)

# Extract translated text

translated\_reply = response.choices[0].message.content

print(translated\_reply)

<script.py> output:

DataCamp offre una varietà di percorsi e corsi focalizzati su diverse tecnologie relative all'IA. Per apprendere l'IA, puoi considerare le seguenti opzioni:

1. \*\*Ingegnere IA Associato per Sviluppatori\*\*: Questo percorso professionale ti aiuta a integrare l'IA nelle applicazioni software utilizzando tecnologie come Modelli Linguistici di Grandi Dimensioni (LLM), ingegneria di prompt, chatbot, motori di raccomandazione e database vettoriali. Copre strumenti come OpenAI API e LangChain. [Esplora qui](https://www.datacamp.com/tracks/associate-ai-engineer-for-developers).

2. \*\*Fondamenti di OpenAI\*\*: Questo percorso di competenze ti insegna ad utilizzare l'OpenAI API in Python per creare applicazioni come chatbot e sistemi di raccomandazione, insieme a esercizi pratici di programmazione. [Scoprilo qui](https://www.datacamp.com/tracks/openai-fundamentals).

3. \*\*Deep Learning in Python\*\*: Se sei interessato al deep learning, questo percorso si immerge nella costruzione di modelli con PyTorch per vari compiti nel mondo reale, inclusa l'identificazione della lingua e la classificazione delle immagini. [Scopri di più qui](https://www.datacamp.com/tracks/deep-learning-in-python).

Ogni percorso offre un diverso enfasi sulle tecnologie IA, quindi puoi scegliere in base ai tuoi interessi e obiettivi!

In [1]:

**Preparing the spoken response**

You've made it to the final step of the case study! Your task now is to convert the tailored, translated response into speech. Since the customer submitted their query via audio, they'll receive an audio reply in return—how cool is that?

The translated reply from the previous exercise is available as translated\_reply.

**Instructions**

**100 XP**

* Use the gpt-4o-mini-tts model with the "onyx" voice to create a speech request for translated\_reply.
* Stream the response to an .mp3 file.

**script.py**

client = OpenAI(api\_key="<OPENAI\_API\_TOKEN>")

# Create the text-to-speech request

response = client.audio.speech.create(

    model="gpt-4o-mini-tts",

    voice="onyx",

    input=translated\_reply

)

# Stream the response to an MP3 file

response.stream\_to\_file("output.mp3")