**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?**

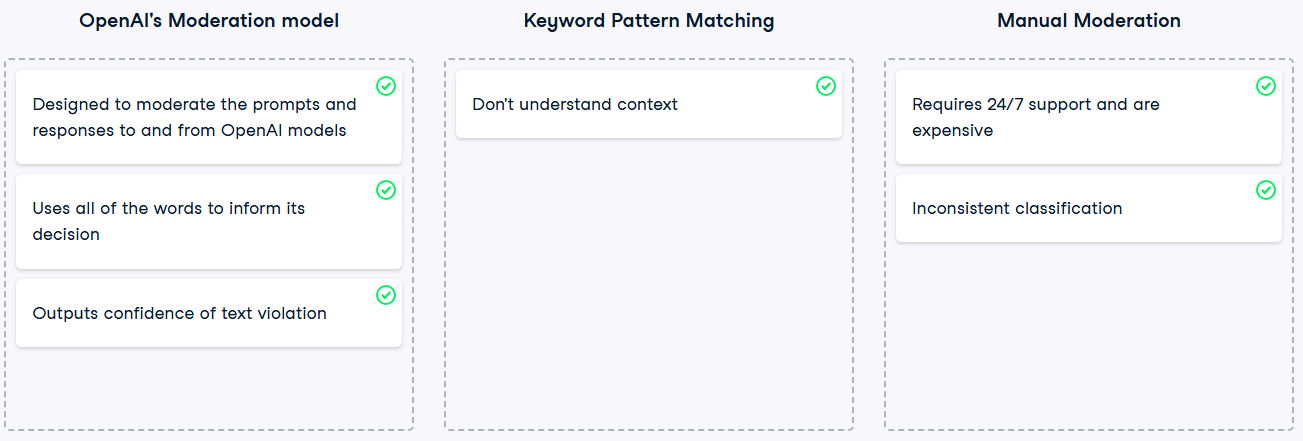
Text moderation is a vital component of most social media platforms, internet chatrooms, and many other user-facing systems. It serves the purpose of preventing the distribution and promotion of inappropriate content, such as hate speech.

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]: