Speech Recognition with Python

By Maciej Morawski

- in/mpfmorawski (LinkedIn)
- @mpfmorawski (GitHub)

whoami

mpfmorawski/README.md

Welcome! 👋

- My name is Maciej Morawski! 🙋
- I'm a Python Backend Developer Intern @ Schneider Electric, living in Warsaw, Poland 🏢
- I have graduated from the MSE in Robotics and Automatic Control @ WUT but...
- That's not the end of my university education! I'm still a Computer Science Student @ WUT !
- I'm interested in Robotics, AI and... all those amazing things you can do using Python 🐍 (that's why I'm here with you today 😉)
- Today is my first time speaking at such a large conference, but I hope to speak many more times!

Agenda

1. Speech-to-text intro
2. SpeechRecognition
3. AssemblyAI
4. OpenAI’s Whisper
5. Transformers
6. Summary

Slides

Slides can be found here

https://bit.ly/pyconpl-asr-slides

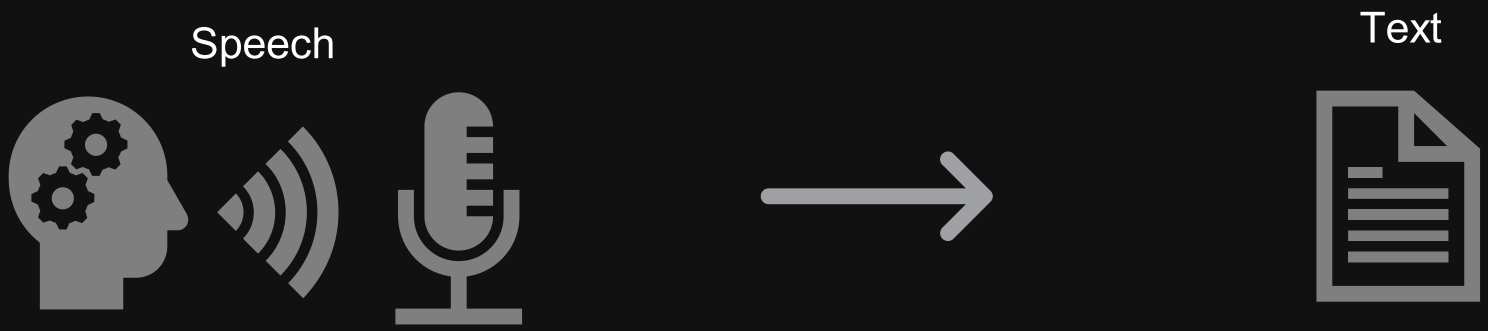
1

Speech-to-text intro

Speech-to-text intro

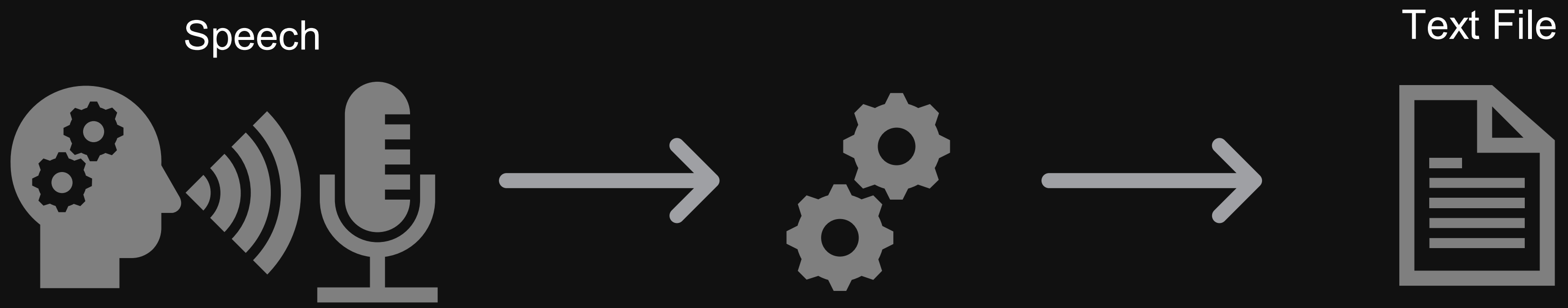
What is... speech recognition?

Speech-to-text intro



Speech Recognition

Speech-to-text intro



Automatic Speech Recognition (ASR)

Speech-To-Text (STT)

Speech-to-text intro

Today we will not focus on theory.
We will answer the question…

What solutions can we use?

Speech-to-text intro

If your first thought was to make your own speech recognition engine

putting it simply...

🛑 **DON’T DO THAT!** 🛑

Speech-to-text intro

Data Requirements

Expertise

Why it's bad idea to

build your own

speech recognition engine?

Time and Cost

Accuracy and Performance

Maintenance and Updates

Speech-to-text intro

And because…

there are plenty of solutions available on the market!

Speech-to-text intro

Speech recognition engines/APIs can be divided by **their availability** into two main groups:

- tools that can only be used online
- tools that can be used offline

Speech-to-text intro

But… speech recognition engines/APIs can also be divided by **service providers** into two main groups:

- external solutions
- internal solutions

Hint 💡: Which is much more important from the company's point of view 😉

Speech-to-text intro

External solutions

Advantages (+):

- Easy to use
- Scalability
- No need for infrastructure and maintenance
- Cost (when processing a small amount of audio data)

Disadvantages (-):

- Privacy concerns
- Dependence on internet connection
- Cost (when processing a large amount of audio data)

Speech-to-text intro

Internal solutions

Advantages (+):

- Data security
- Cost (when processing a large amount of audio data)

Disadvantages (-):

- Limited scalability
- Maintenance
- Cost (when processing a small amount of audio data)

Speech-to-text intro

What options are available to us as Python Developers 🐍?

Speech-to-text intro

Let's go straight to practice together!

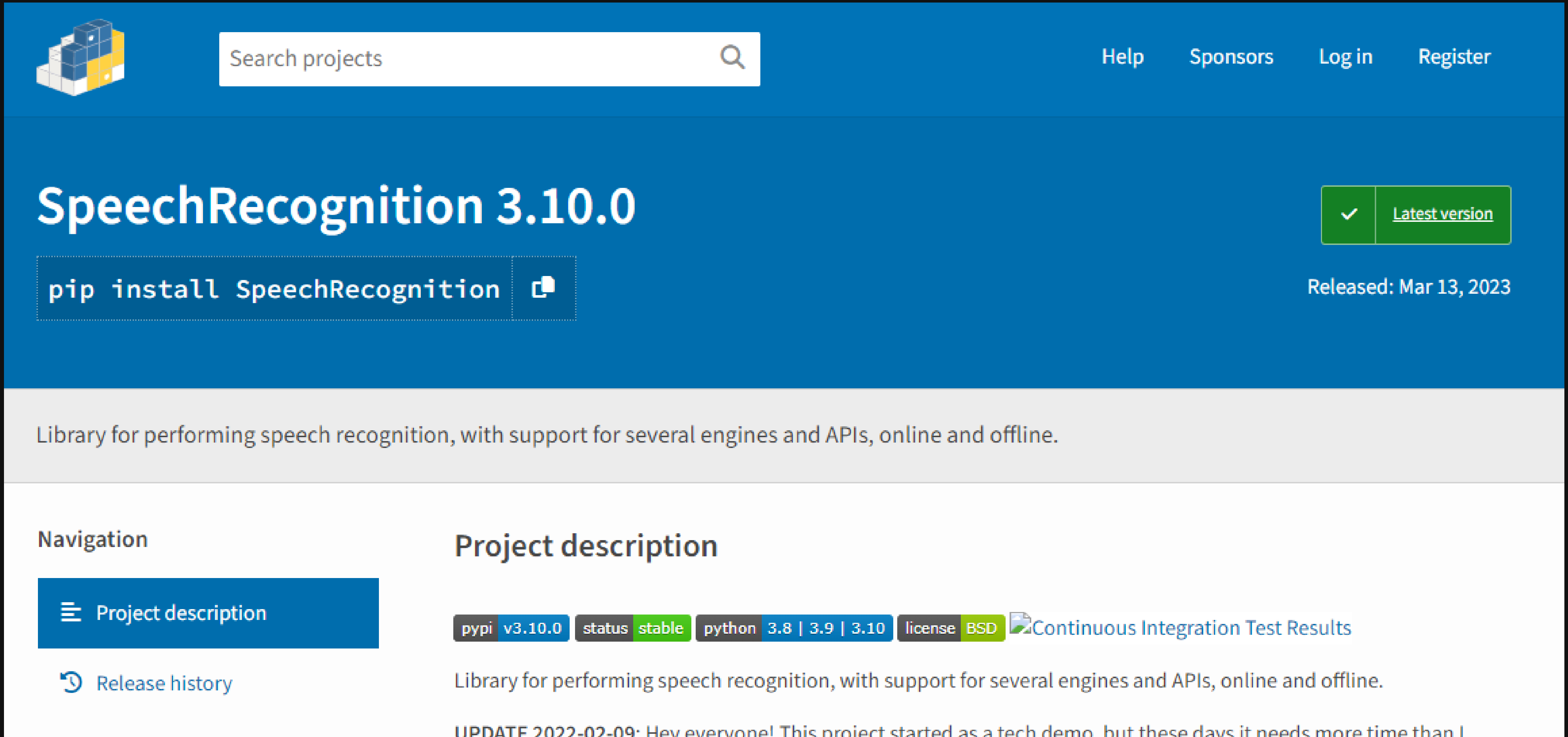
https://bit.ly/pyconpl-asr-notebook

2

SpeechRecognition

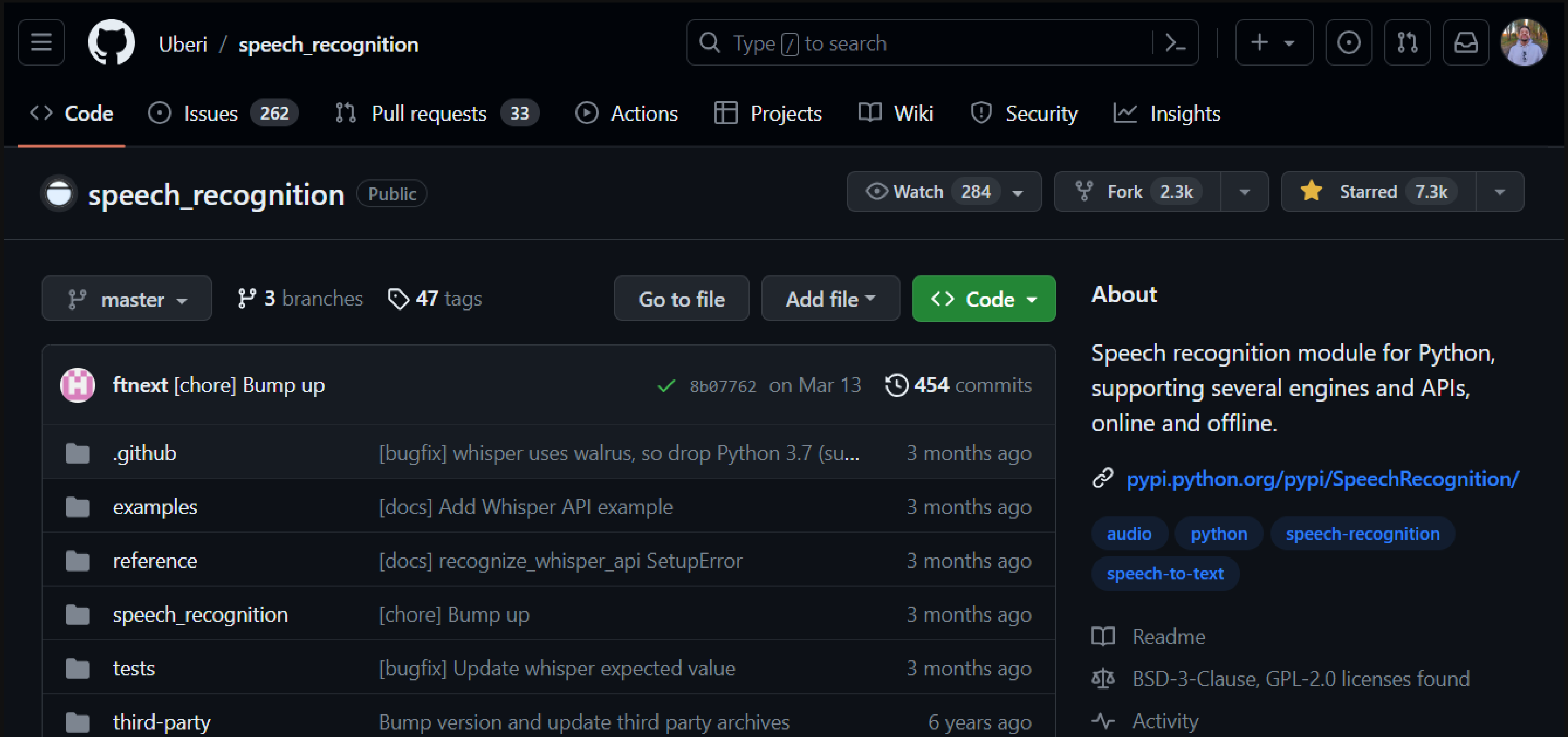
SpeechRecognition

Source: https://pypi.org/project/SpeechRecognition/



SpeechRecognition

Source: https://github.com/Uberi/speech_recognition



SpeechRecognition

Speech recognition engine/API support:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Azure Speech
- Houndify API
- IBM Speech to Text
- Snowboy Hotword Detection (works offline)
- Tensorflow
- Vosk API (works offline)
- OpenAI whisper (works offline)
- Whisper API

✨ added in the last release in March ✨

SpeechRecognition

```
pip install SpeechRecognition
```

SpeechRecognition

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)

r.recognize_google(audio)
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

Google Speech Recognition (using the default API key)

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)

try:
  print( r.recognize_google(audio) )
except sr.UnknownValueError:
  print( "Could not understand audio" )
except sr.RequestError as e:
  print( f"Could not request results service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

Google Cloud Speech

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)


GOOGLE_CLOUD_SPEECH_CREDENTIALS = "CREDENTIALS"

try:
  print( r.recognize_google_cloud(audio, credentials_json=GOOGLE_CLOUD_SPEECH_CREDENTIALS) )
except sr.UnknownValueError:
  print( "Could not understand audio" )

except sr.RequestError as e:
  print( f"Could not request results from service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

Wit.ai

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)


WIT_AI_KEY = „KEY"

try:
  print( r.recognize_wit(audio, key=WIT_AI_KEY) )
except sr.UnknownValueError:
  print( "Could not understand audio" )
except sr.RequestError as e:
  print( f"Could not request results from service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

Houndify

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)


HOUNDIFY_CLIENT_ID = "ID"
HOUNDIFY_CLIENT_KEY = "KEY"

try:
  print( r.recognize_houndify(audio,
                              client_id=HOUNDIFY_CLIENT_ID,
                              client_key=HOUNDIFY_CLIENT_KEY)
except sr.UnknownValueError:
  print( "Could not understand audio" )
except sr.RequestError as e:
  print( f"Could not request results from service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

IBM Speech to Text

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)



IBM_USERNAME = „USERNAME"
IBM_PASSWORD = „PASSWORD"

try:
  print( r.recognize_ibm(audio,
                         username=IBM_USERNAME,
                         password=IBM_PASSWORD) )
except sr.UnknownValueError:
  print( "Could not understand audio" )
except sr.RequestError as e:
  print( f"Could not request results from service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

SpeechRecognition

Whisper API ← ✨ added in the last release in March ✨

```
import speech_recognition as sr

r = sr.Recognizer()
  with sr.AudioFile(audio_path) as source:
    audio = r.record(source)


OPENAI_API_KEY = „KEY"

try:
  print( r.recognize_whisper_api(audio, api_key=OPENAI_API_KEY) )
except sr.UnknownValueError:
  print( "Could not understand audio" )
except sr.RequestError as e:
  print( f"Could not request results from service; {e}" )
```

Source: https://github.com/Uberi/speech_recognition/blob/master/examples/audio_transcribe.py

Speech-to-text intro

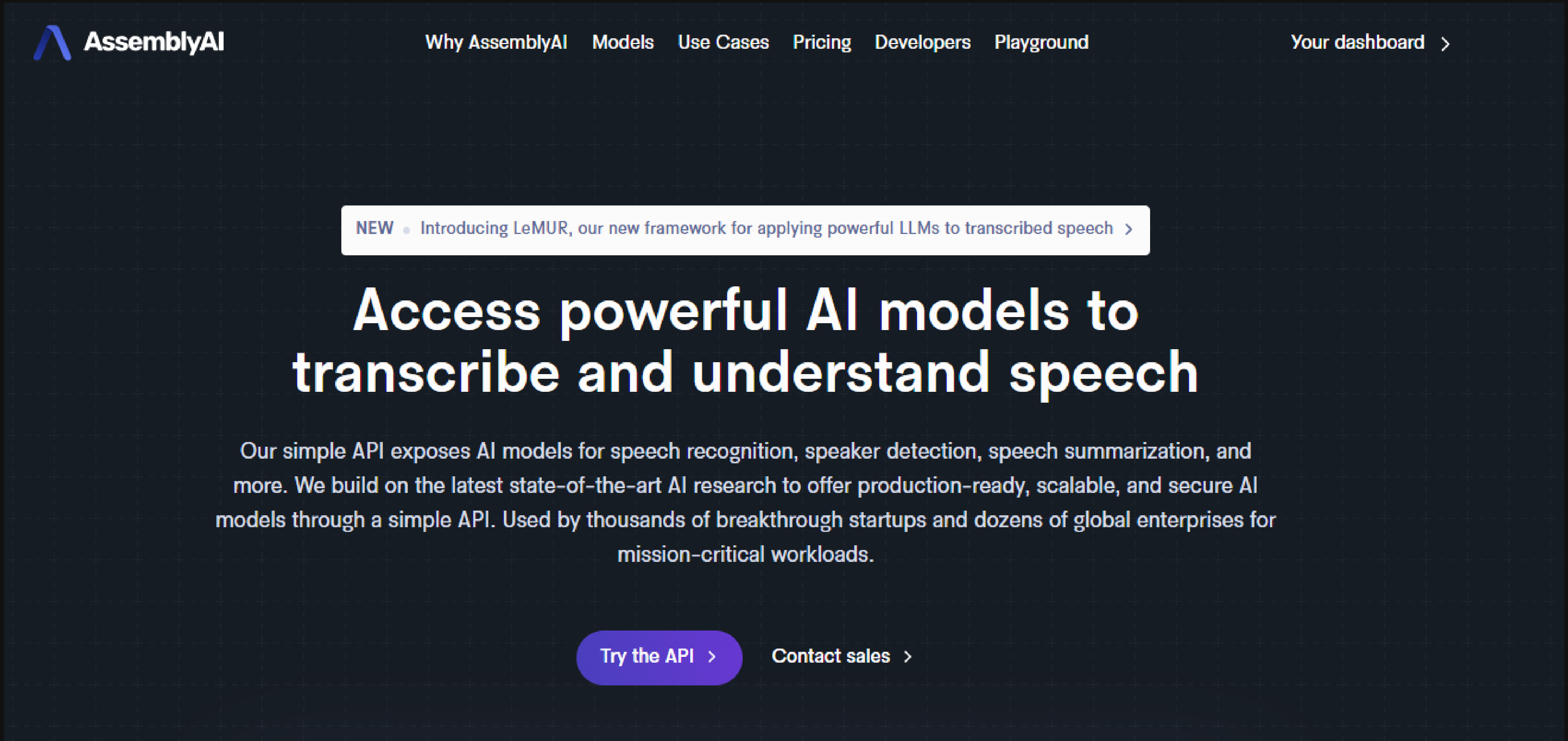
Let's get back to Google Colab!

3

AssemblyAI

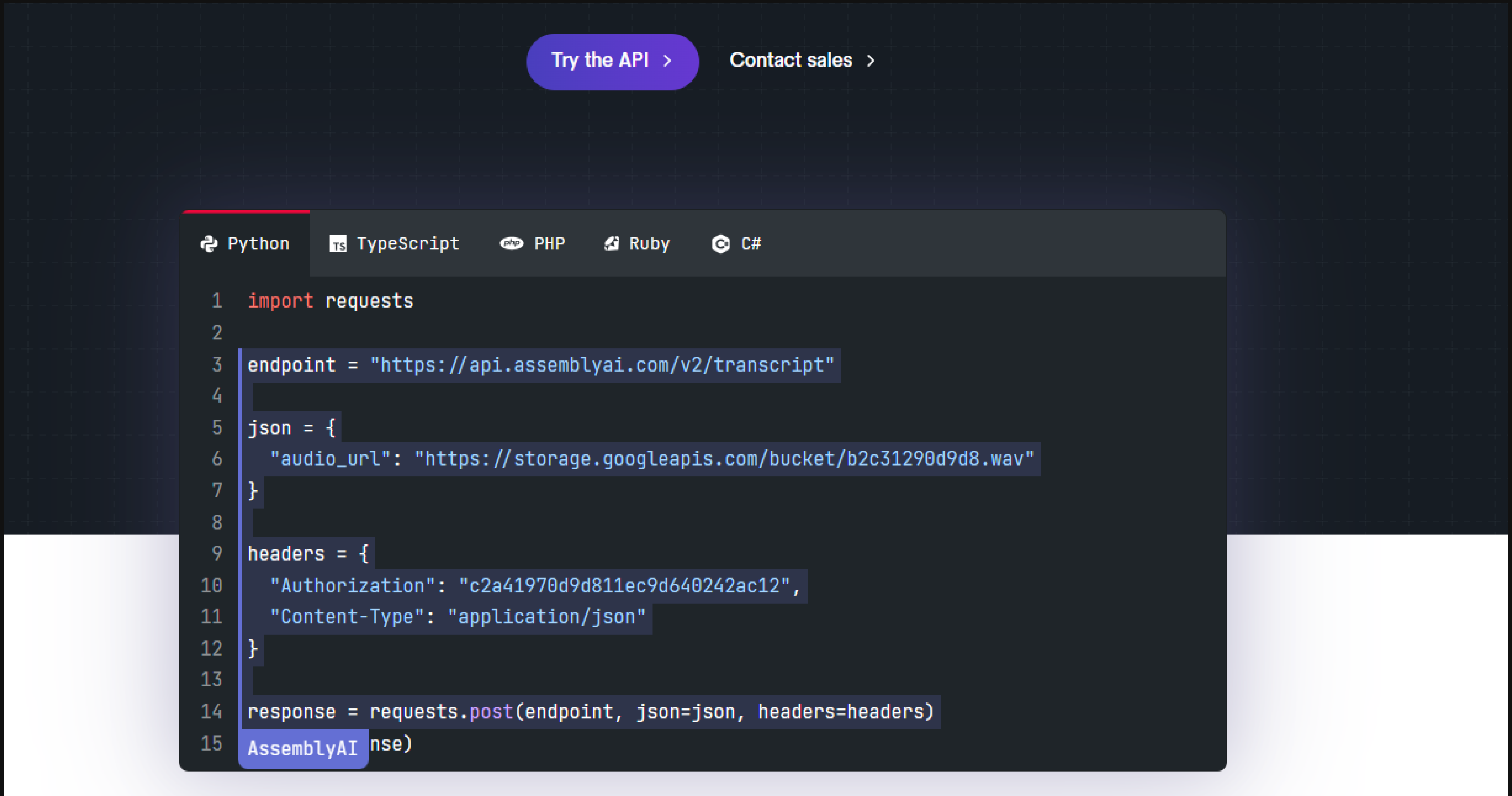
AssemblyAI API

Source: https://www.assemblyai.com/



AssemblyAI API

Source: https://www.assemblyai.com/



AssemblyAI API

```python
import requests
```

AssemblyAI API

```
endpoint = "https://api.assemblyai.com/v2/transcript"

json = {
  "audio_url": "https://storage.googleapis.com/bucket/b2c31290d9d8.wav"
}

headers = {
  "Authorization": "c2a41970d9d811ec9d640242ac12",
  "Content-Type": "application/json"
}

response = requests.post(endpoint, json=json, headers=headers)
parse(response)
```

Source: https://www.assemblyai.com/

AssemblyAI API

Output

```
NameError                                 Traceback (most recent call last)
<ipython-input-2-3072985cb8ce> in <cell line: 13>()
     11
     12 response = requests.post(endpoint, json=json, headers=headers)
---> 13 parse(response)

NameError: name 'parse' is not defined
```

AssemblyAI API

1

```
endpoint = "https://api.assemblyai.com/v2/transcript"

json = {
  "audio_url": "https://storage.googleapis.com/bucket/b2c31290d9d8.wav"
}

headers = {
  "Authorization": "c2a41970d9d811ec9d640242ac12",
  "Content-Type": "application/json"
}

response = requests.post(endpoint, json=json, headers=headers)
parse(response)
```

2

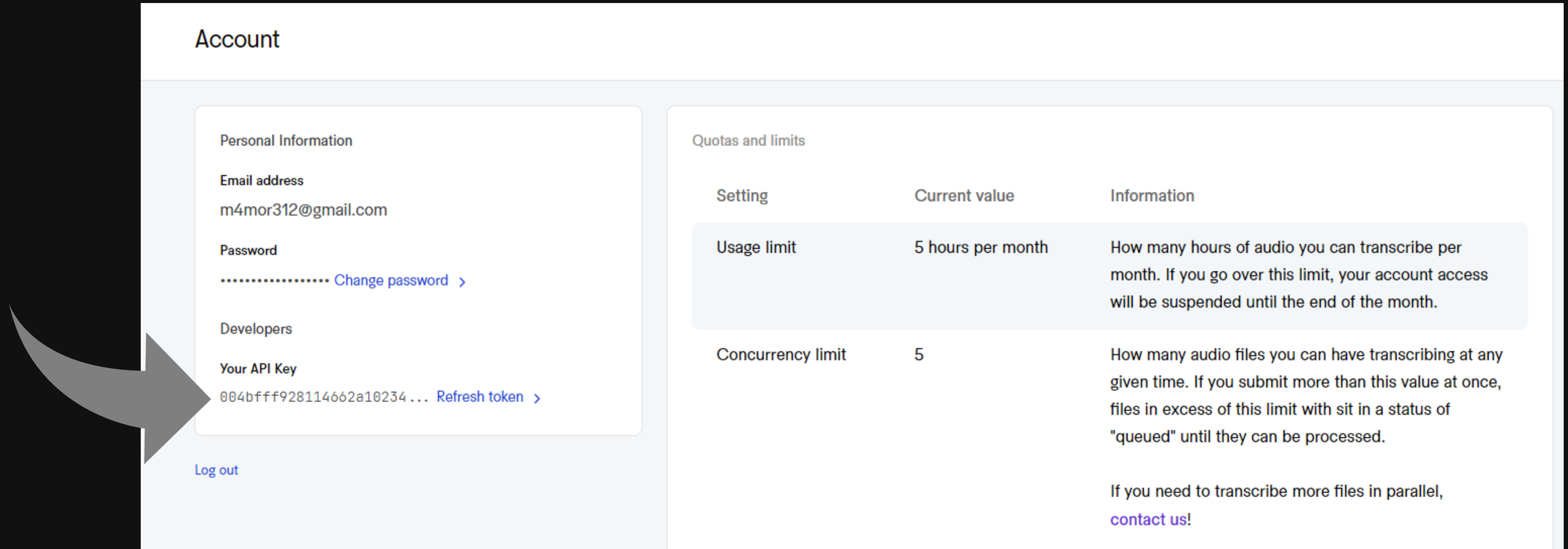
3

Source: https://www.assemblyai.com/

AssemblyAI API

Authorization

1. Go to: https://www.assemblyai.com/dashboard/signup
2. Sign up
3. Go to: https://www.assemblyai.com/app/account
4. Copy your API Key



AssemblyAI API

```
ASSEMBLY_AI_API_KEY = "your_api_key"
```

AssemblyAI API

Uploading audio

```
UPLOAD_ENDPOINT = "https://api.assemblyai.com/v2/upload"

headers = {"authorization": ASSEMBLY_AI_API_KEY}
with open(filename , "rb") as f:
    response = requests.post(UPLOAD_ENDPOINT,
                             headers=headers,
                             data=f)

print(response.json())
```

Source: https://www.assemblyai.com/docs/Guides/transcribing_an_audio_file#uploading-local-files-for-transcription

AssemblyAI API

Output

```
{'upload_url': 'https://cdn.assemblyai.com/upload/random-letters-and-numbers'}
```

AssemblyAI API

Making request for transcription

```
TRANSCRIPT_ENDPOINT = "https://api.assemblyai.com/v2/transcript"

json = {
  "audio_url": response.json()["upload_url"]
}

headers = {
  "Authorization": "c2a41970d9d811ec9d640242ac12",
  "Content-Type": "application/json"
}

response = requests.post(TRANSCRIPT_ENDPOINT,
                         json=json,
                         headers=headers)

print(response.json())
```

Source: https://www.assemblyai.com/docs/Guides/transcribing_an_audio_file#uploading-local-files-for-transcription

AssemblyAI API

Output

```
json full of data!
```

But if we take a closer look…

```
response.json()['text']=None
response.json()['status']='queued’
```

AssemblyAI API

Polling

```
polling_endpoint = f"{TRANSCRIPT_ENDPOINT}/{response.json()['id']}"

while True:
  response = requests.get(polling_endpoint, headers=headers).json()
  if response['status'] == 'completed':
    break
  elif response['status'] == 'error':
    raise RuntimeError(f"Transcription failed: {response['error']}")
  else:
    time.sleep(3)

print(response["text"])
```

Source: https://www.assemblyai.com/docs/Guides/transcribing_an_audio_file#uploading-local-files-for-transcription

AssemblyAI API

Output

```
Our transcription!
```

AssemblyAI API

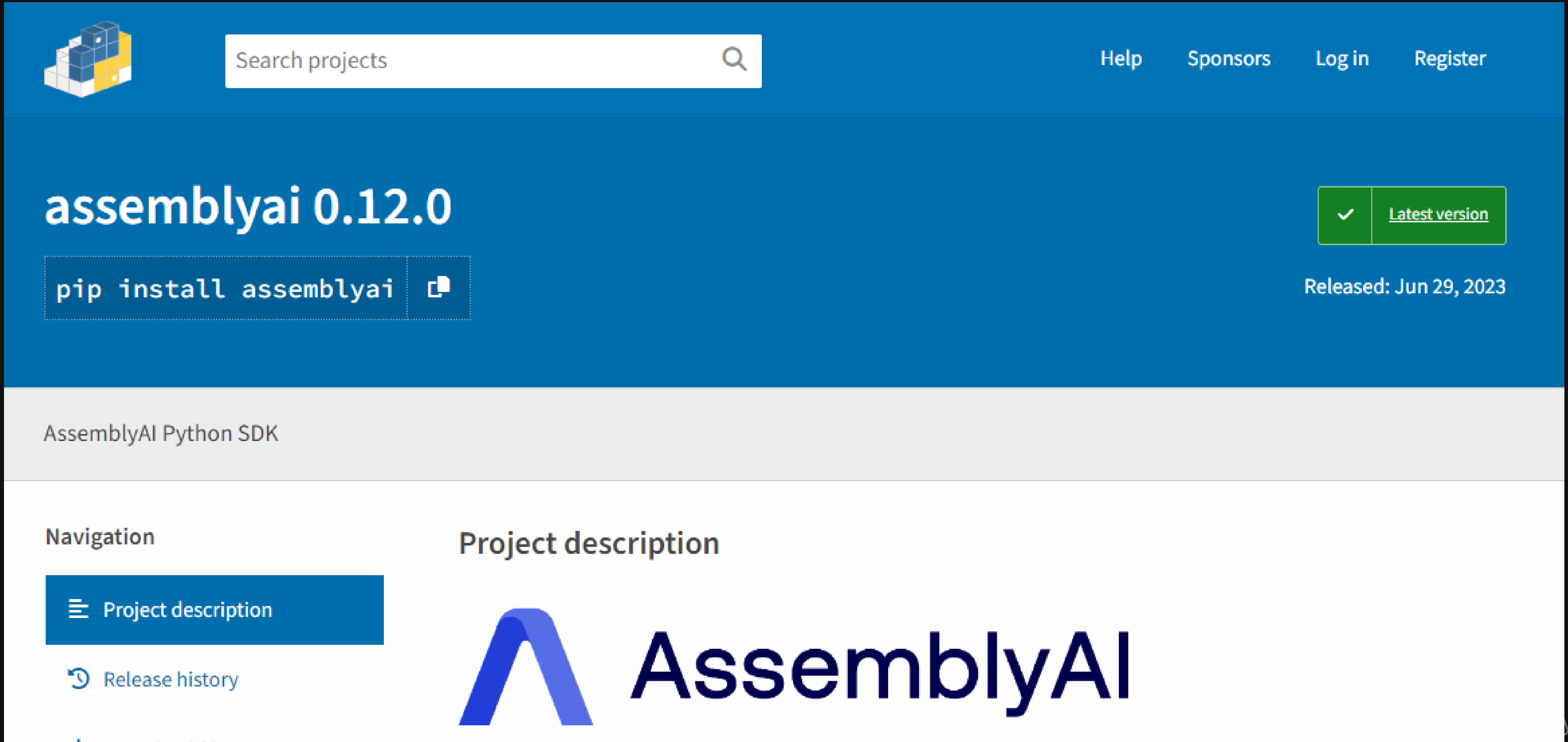
Let's get back to Google Colab!

AssemblyAI API

On the other hand, we can use…

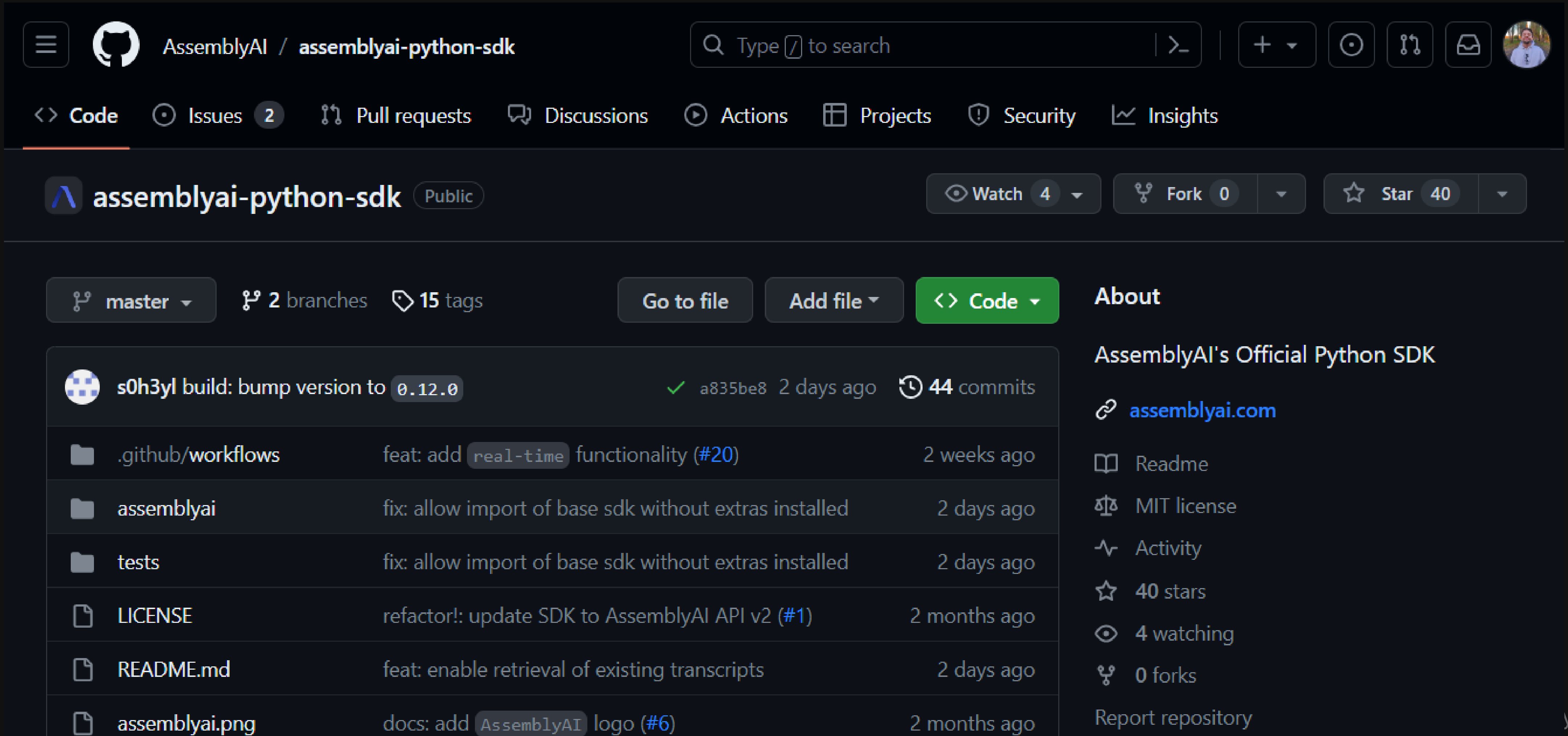
AssemblyAI API

Source: https://pypi.org/project/assemblyai/



AssemblyAI API

Source: https://github.com/AssemblyAI/assemblyai-python-sdk



AssemblyAI API

```
pip install assemblyai
```

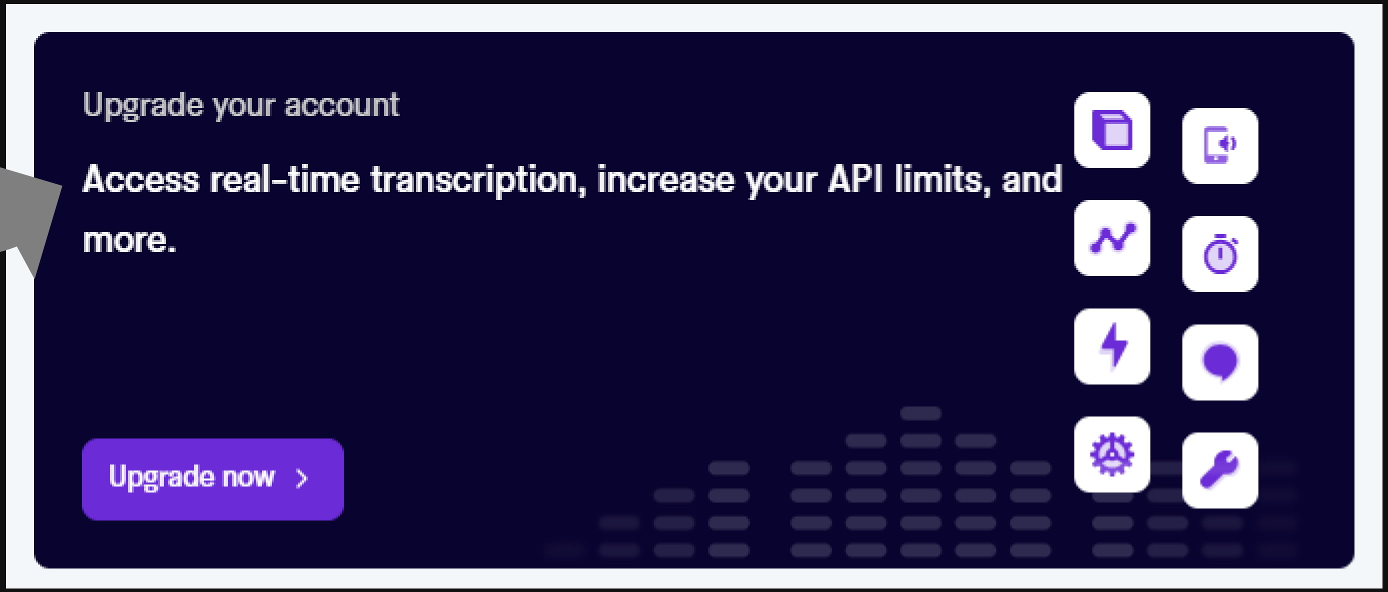
AssemblyAI API

```
import assemblyai as aai

aai.settings.api_key = "YOUR API KEY"

transcriber = aai.Transcriber()
transcript = transcriber.transcribe(audio_path)

print( transcript.text )
```

Source: https://www.assemblyai.com/docs/Guides/transcribing_an_audio_file#uploading-local-files-for-transcription

AssemblyAI API

Source: https://www.assemblyai.com/app

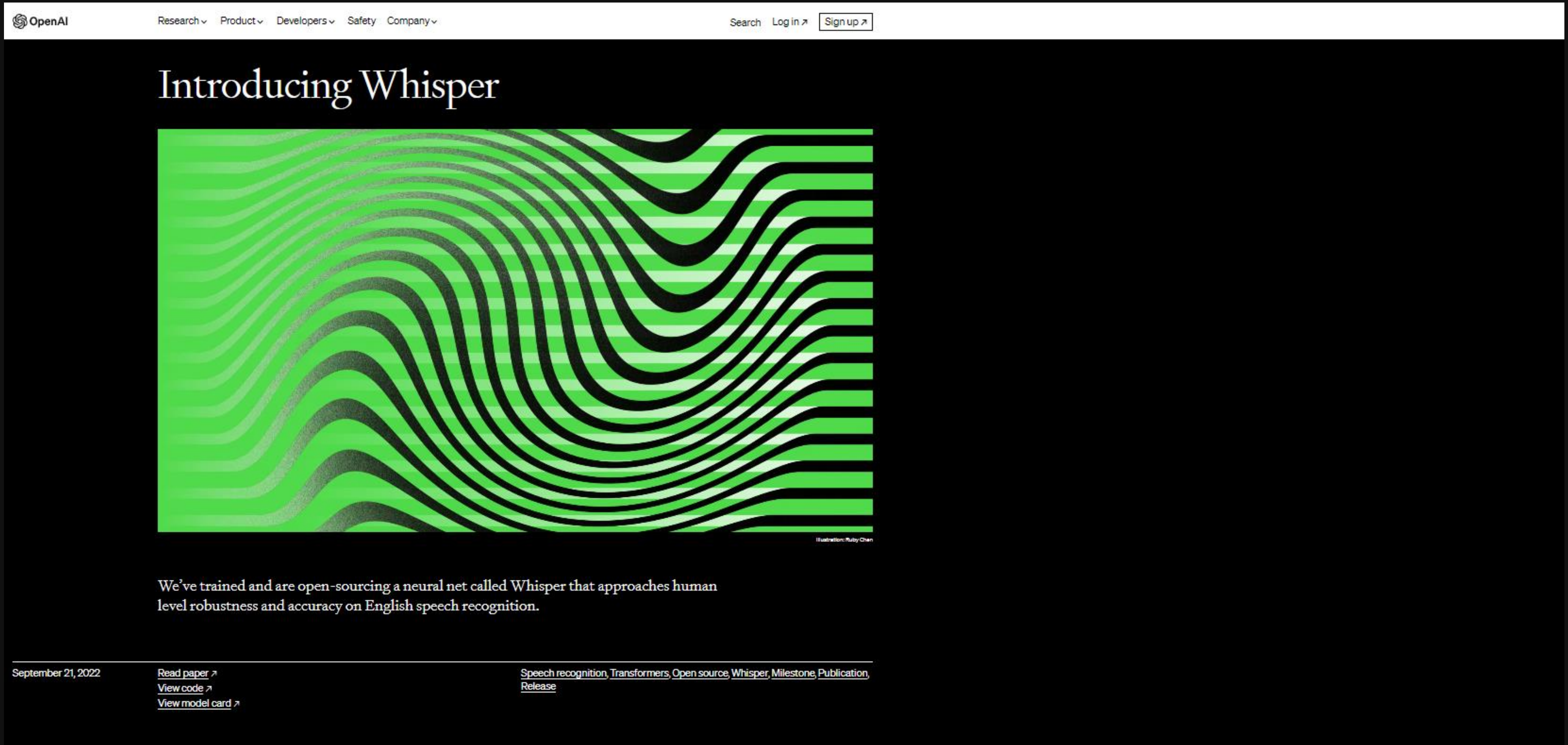


4

OpenAI's Whisper

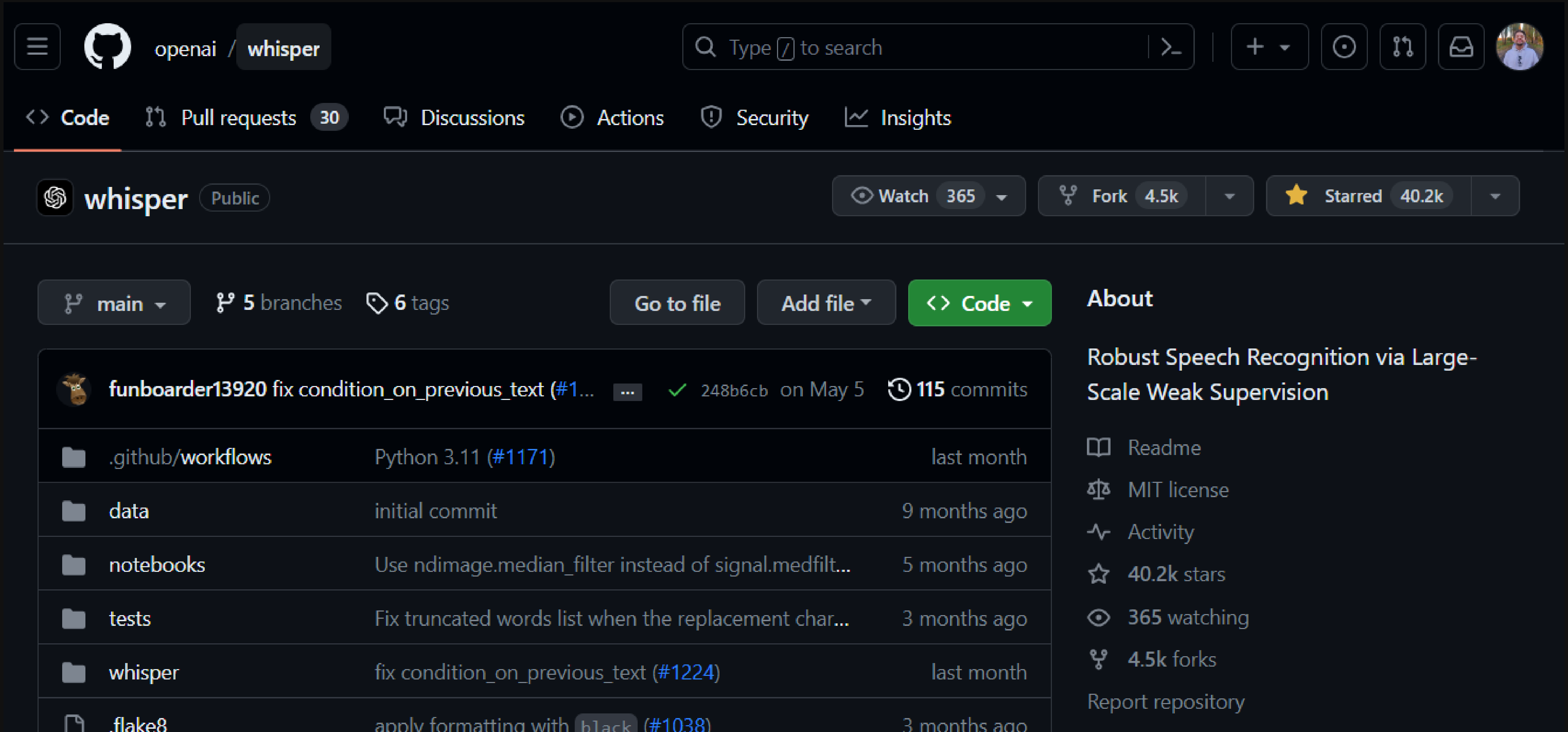
OpenAI's Whisper

Source: https://openai.com/research/whisper



OpenAI's Whisper

Source: https://openai.com/research/whisper



OpenAI's Whisper

```
pip install -U openai-whisper
```

OpenAI's Whisper

```
import whisper

model = whisper.load_model(model_name)
transcription = model.transcribe(audio_path)
print(transcription["text"])
```

Source: https://github.com/openai/whisper/blob/main/notebooks/Multilingual_ASR.ipynb

OpenAI's Whisper

Source: https://github.com/openai/whisper

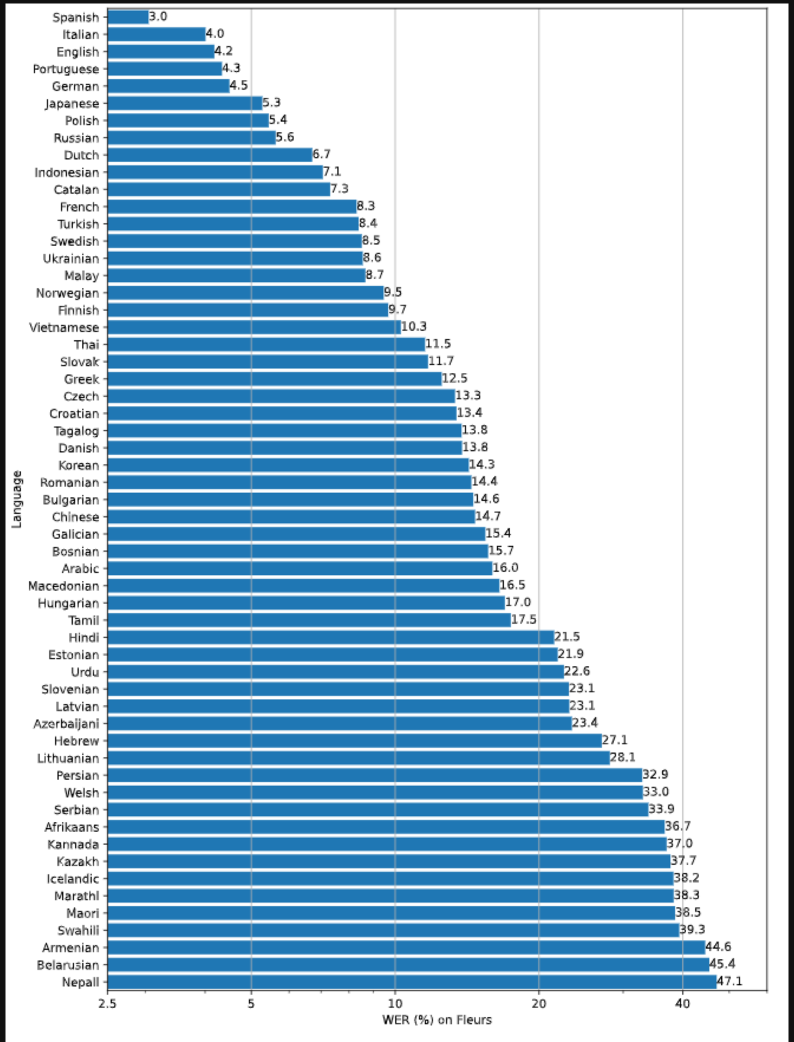
Available models and languages

There are five model sizes, four with English-only versions, offering speed and accuracy tradeoffs. Below are the names of the available models and their approximate memory requirements and relative speed.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~32x
base	74 M	`base.en`	`base`	~1 GB	~16x
small	244 M	`small.en`	`small`	~2 GB	~6x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x

OpenAI's Whisper

Source: https://github.com/openai/whisper



OpenAI's Whisper

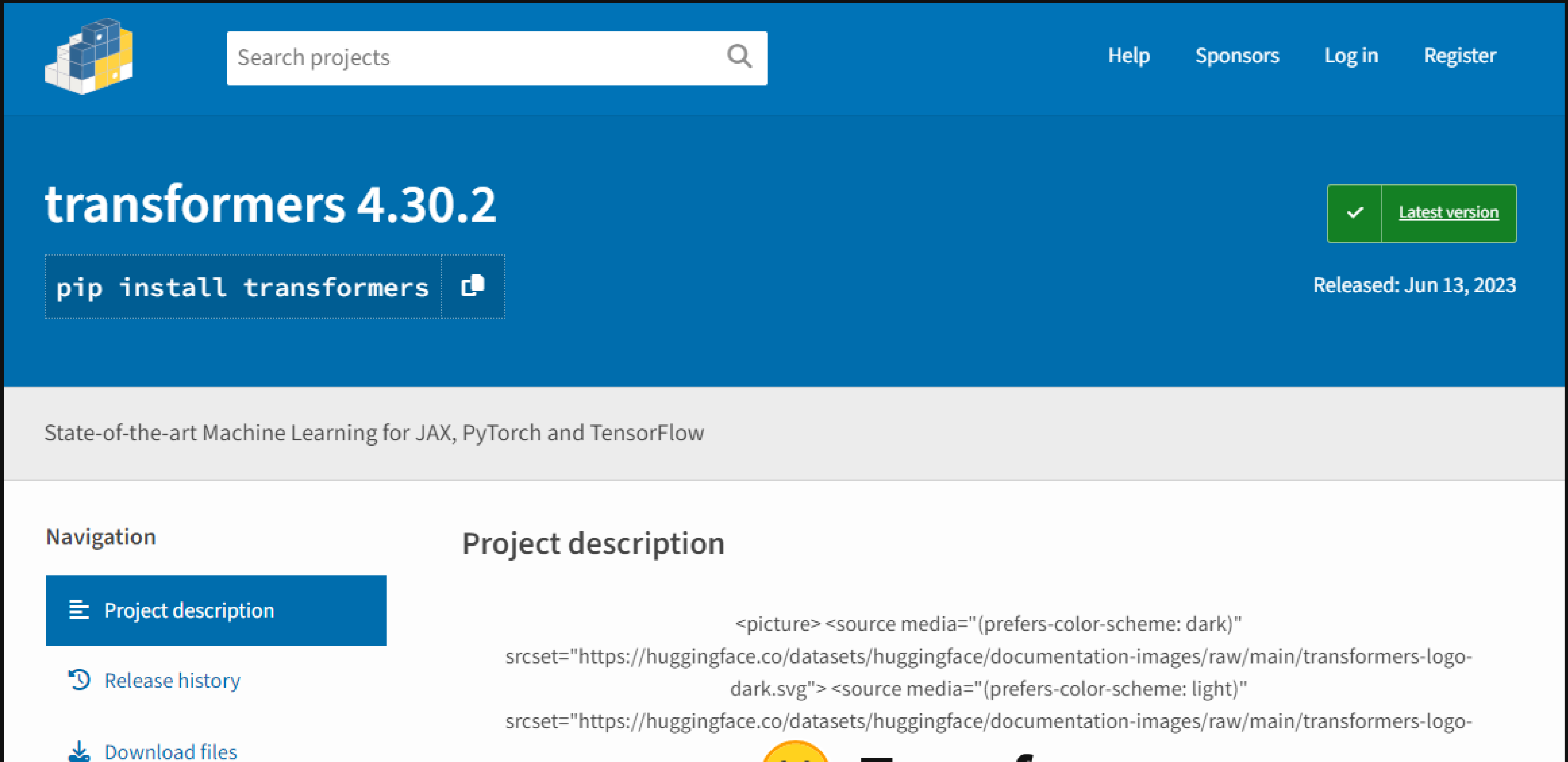
Let's get back to Google Colab!

5

Transformers

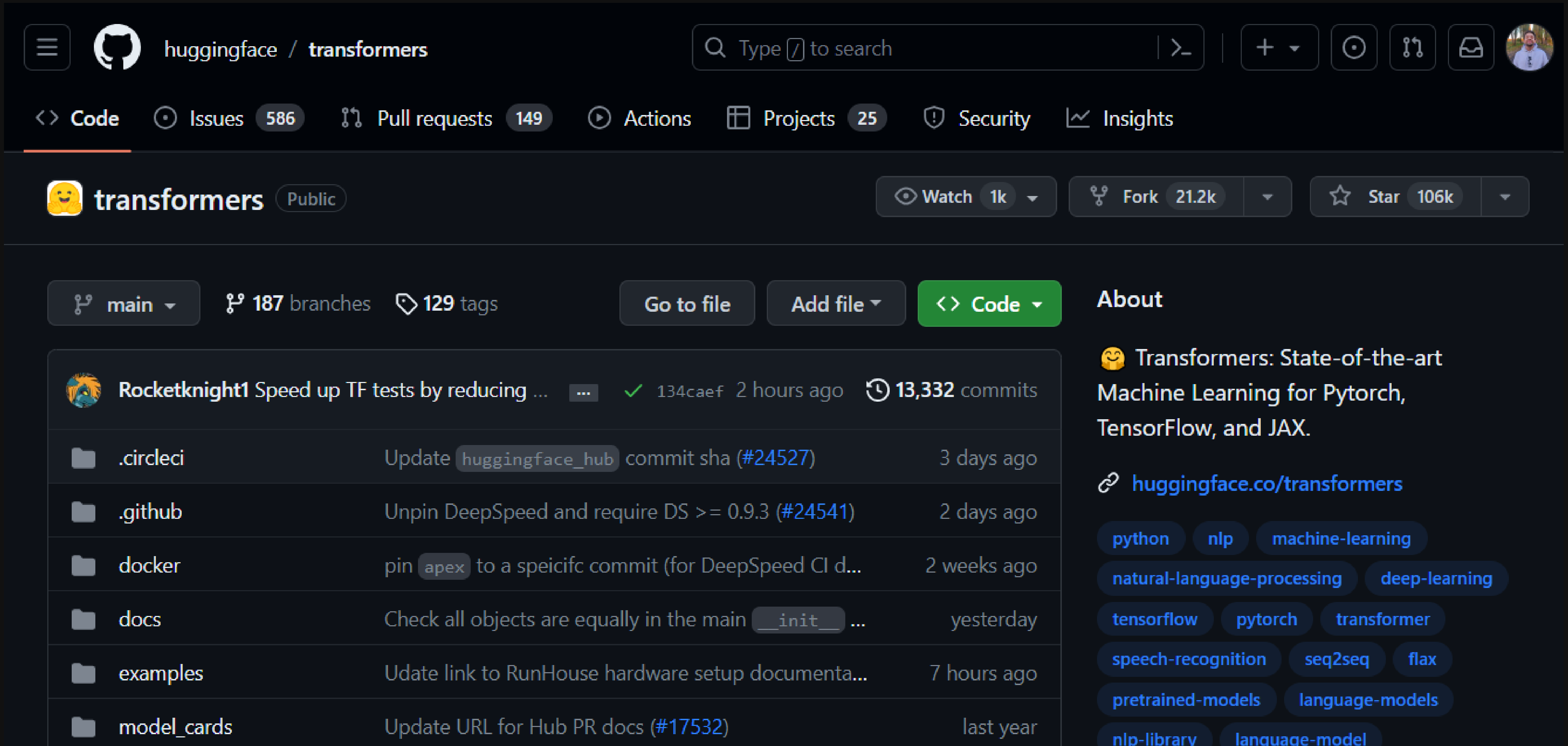
Transformers

Source: https://pypi.org/project/transformers/



Transformers

Source: https://github.com/huggingface/transformers



Transformers

```
pip install transformers
```

Transformers

```
from transformers import pipeline

transcriber = pipeline("automatic-speech-recognition",
                       model="facebook/wav2vec2-base-960h")
transcription = transcriber(audio_path)
print( transcription["text"] )
```

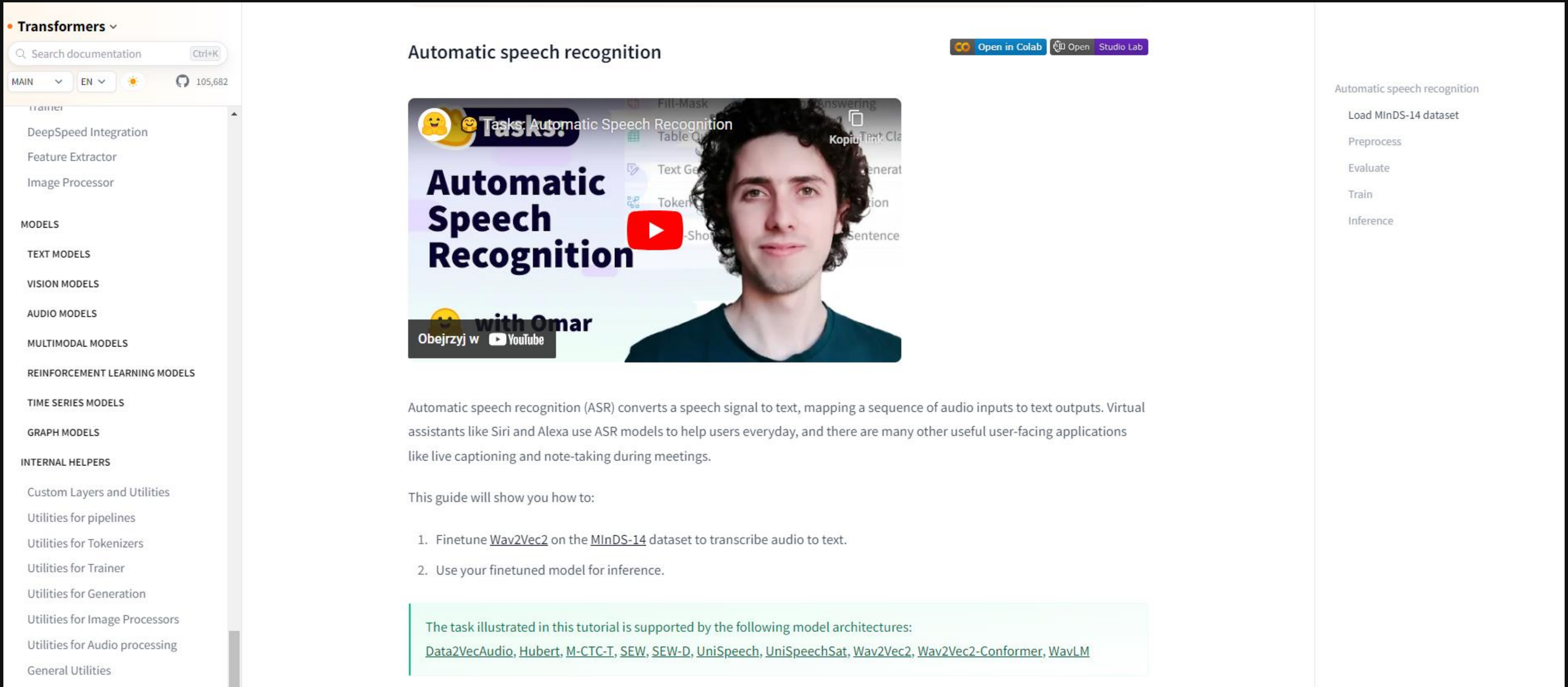
Source: https://huggingface.co/docs/transformers/main/tasks/asr

Transformers

Let's get back to Google Colab!

Transformers

Source: https://huggingface.co/docs/transformers/main/tasks/asr



Transformers

Search documentation Ctrl+K

MAIN EN 105,682

DeepSpeed Integration

Feature Extractor

Image Processor

MODELS

TEXT MODELS

VISION MODELS

AUDIO MODELS

MULTIMODAL MODELS

REINFORCEMENT LEARNING MODELS

TIME SERIES MODELS

GRAPH MODELS

INTERNAL HELPERS

Custom Layers and Utilities

Utilities for pipelines

Utilities for Tokenizers

Utilities for Trainer

Utilities for Generation

Utilities for Image Processors

Utilities for Audio processing

General Utilities

Automatic speech recognition

Open in Colab Open Studio Lab

Automatic speech recognition (ASR) converts a speech signal to text, mapping a sequence of audio inputs to text outputs. Virtual assistants like Siri and Alexa use ASR models to help users everyday, and there are many other useful user-facing applications like live captioning and note-taking during meetings.

This guide will show you how to:

1. Finetune Wav2Vec2 on the MInDS-14 dataset to transcribe audio to text.
2. Use your finetuned model for inference.

The task illustrated in this tutorial is supported by the following model architectures:
Data2VecAudio, Hubert, M-CTC-T, SEW, SEW-D, UniSpeech, UniSpeechSat, Wav2Vec2, Wav2Vec2-Conformer, WavLM

Automatic speech recognition

Load MInDS-14 dataset

Preprocess

Evaluate

Train

Inference

Transformers

Source: https://huggingface.co/docs/transformers/main/tasks/asr

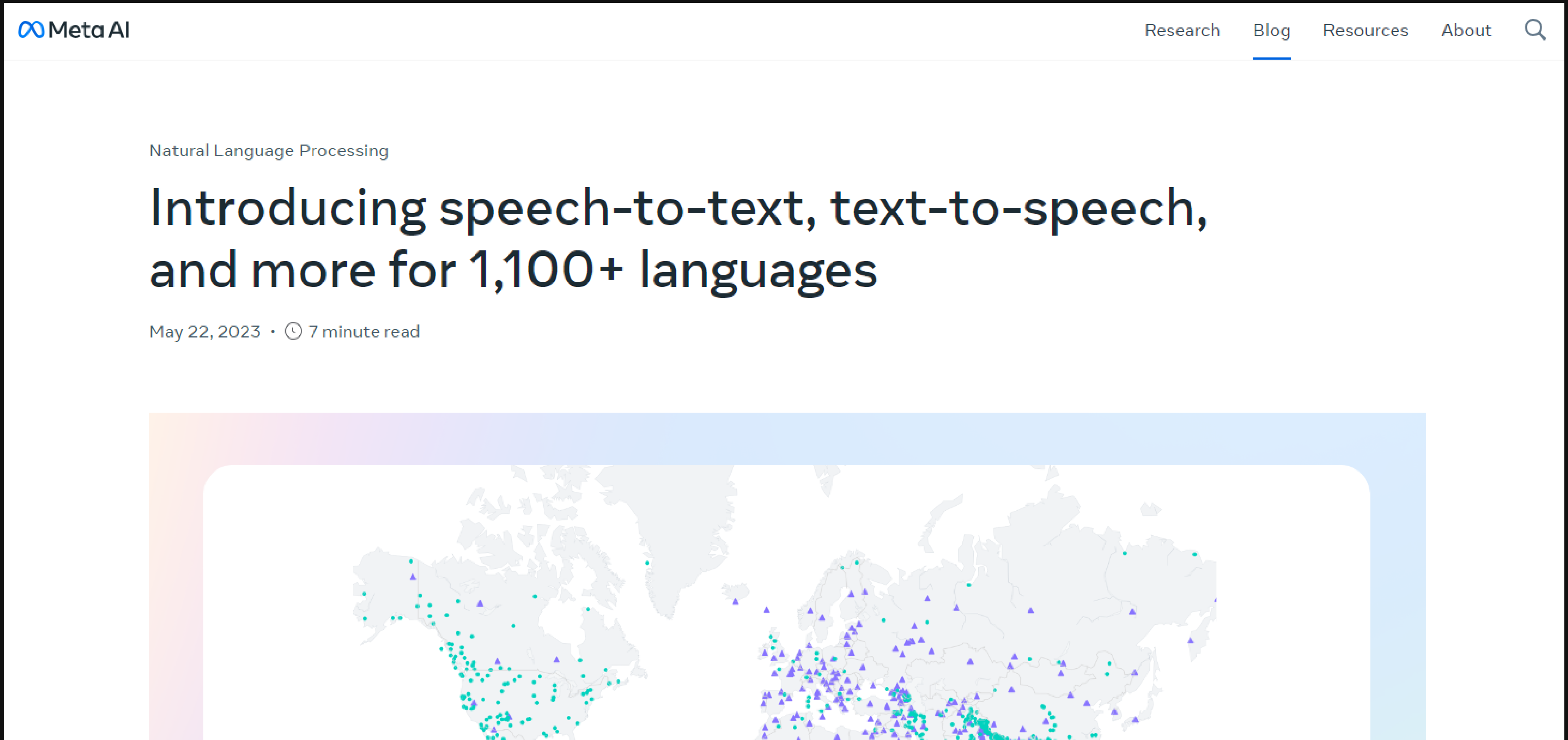
The task illustrated in this tutorial is supported by the following model architectures:

Data2VecAudio, Hubert, M-CTC-T, SEW, SEW-D, UniSpeech, UniSpeechSat, Wav2Vec2, Wav2Vec2-Conformer, WavLM

The latest announcement

The latest announcement

Source: https://ai.facebook.com/blog/multilingual-model-speech-recognition/



The latest announcement

Source: https://ai.facebook.com/blog/multilingual-model-speech-recognition/

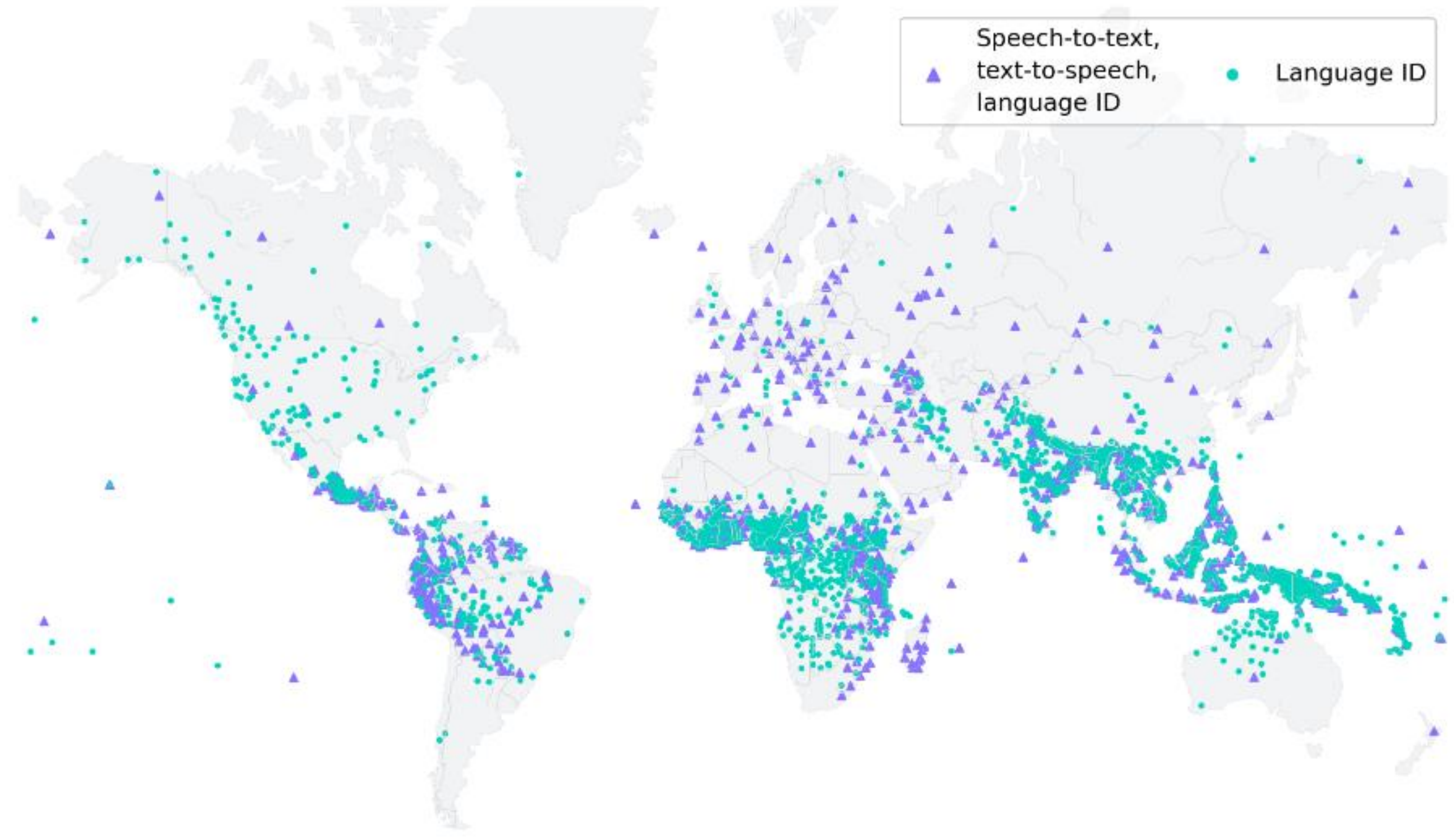


Illustration of the languages the Massively Multilingual Speech (MMS) recognition model supports. MMS supports speech-to-text and text-to-speech for 1,107 languages and language identification for over 4,000 languages.

The latest announcement

Source: https://ai.facebook.com/blog/multilingual-model-speech-recognition/

We trained multilingual speech recognition models on over 1,100 languages using a 1B parameter wav2vec 2.0 model. As the number of languages increases, performance does decrease, but only very slightly: Moving from 61 to 1,107 languages increases the character error rate by only about 0.4 percent but increases the language coverage by over 18 times.

The latest announcement

Source: https://github.com/facebookresearch/fairseq/tree/main/examples/mms

README.md

MMS: Scaling Speech Technology to 1000+ languages

The Massively Multilingual Speech (MMS) project expands speech technology from about 100 languages to over 1,000 by building a single multilingual speech recognition model supporting over 1,100 languages (more than 10 times as many as before), language identification models able to identify over 4,000 languages (40 times more than before), pretrained models supporting over 1,400 languages, and text-to-speech models for over 1,100 languages. Our goal is to make it easier for people to access information and to use devices in their preferred language.

You can find details in the paper Scaling Speech Technology to 1000+ languages and the blog post.

An overview of the languages covered by MMS can be found here.

🤗 Transformers

MMS has been added to Transformers. For more information, please refer to Transformers' MMS docs.

Click here to find all MMS checkpoints on the Hub.

Checkout the demo here Open in HF Spaces

The latest announcement

Source: https://huggingface.co/docs/transformers/main/en/model_doc/mms

Automatic Speech Recognition (ASR)

The ASR model checkpoints can be found here : mms-1b-fl102, mms-1b-l1107, mms-1b-all. For best accuracy, use the `mms-1b-all` model.

Tips:

- All ASR models accept a float array corresponding to the raw waveform of the speech signal. The raw waveform should be pre-processed with Wav2Vec2FeatureExtractor.
- The models were trained using connectionist temporal classification (CTC) so the model output has to be decoded using Wav2Vec2CTCTokenizer.
- You can load different language adapter weights for different languages via load_adapter(). Language adapters only consists of roughly 2 million parameters and can therefore be efficiently loaded on the fly when needed.

6

Summary

Summary

1. Check out the different tools and choose the one that best suits your needs!

2. After this lecture, you can start using speech recognition in your projects right away - just use the examples presented today.

Thank you!

Feel free to ask questions **now**
or contact me **later**!

How to contact me?

You can contact me via LinkedIn:
https://www.linkedin.com/in/mpfmorawski/

Or check out more on my GitHub:
https://github.com/mpfmorawski

Morawski Maciej

@mpfmorawski

Hope to see you soon!