Image by Slang Labs

SPEECH RECOGNITION

How to build Python transcriber using Mozilla DeepSpeech

Transcriber with PyAudio and DeepSpeech in 70 lines of Python code.

Satish Chandra Gupta Follow
Jan 14 · 6 min read ★

Voice Assistants are one of the hottest techs right now. Siri, Alexa, Google Assistant, all aim to help you talk to computers and not just touch and type. Automated Speech

Recognition (ASR) and Natural Language Understanding (NLU/NLP) are the key technologies enabling it. If you are just-a-programmer like me, you might be itching to get a piece of the action and hack something. You are at the right place; read on.

Though these technologies are hard and the learning curve is steep, but are becoming increasingly accessible. Last month, Mozilla released DeepSpeech along with models for US English. It has smaller and faster models than ever before, and even has a TensorFlow Lite model that runs faster than real time on a single core of a Raspberry Pi 4. There are several interesting aspects, but right now I am going to focus on its refreshingly simple batch and stream APIs in C, .NET, Java, JavaScript, and Python for converting speech to text. By the end of this blog post, you will build a voice transcriber. No kidding :-)

Let's get started

You need a computer with Python 3.6.5+ installed, good internet connection, and elementary Python programming skills. Even if you do not know Python, read along, it is not so hard. If you don't want to install anything, you can try out DeepSpeech APIs in the browser using this code lab.

Let's do the needed setup:

```
# Create virtual environment named ds082
$ python3 -m venv ./some/pyenv/dir/path/ds082

# Switch to virtual environment
$ source ./some/pyenv/dir/path/ds082/bin/activate

# Install DeepSpeech
$ pip3 install deepspeech==0.8.2

# Download and unzip en-US models, this will take a while
$ mkdir -p ./some/workspace/path/ds082
$ cd ./some/workspace/path/ds082

$ mkdir deepspeech-0.8.2-models
$ wget
https://github.com/mozilla/DeepSpeech/releases/download/v0.8.1/deepsp
eech-0.8.1-models.pbmm
$ wget
https://github.com/mozilla/DeepSpeech/releases/download/v0.8.1/deepsp
eech-0.8.1-models.scorer
$ mv deepspeech-0.8.1-models.pbmm deepspeech-0.8.1-models.scorer
```

```
deepspeech-0.8.2-models/
$ ls -l ./deepspeech-0.8.2-models/

# Download and unzip some audio samples to test setup
$ curl -LO
https://github.com/mozilla/DeepSpeech/releases/download/v0.8.2/audio-
0.8.2.tar.gz
$ tar -xvzf audio-0.8.2.tar.gz
x audio/
x audio/2830-3980-0043.wav
x audio/Attribution.txt
x audio/4507-16021-0012.wav
x audio/8455-210777-0068.wav
x audio/License.txt

$ ls -l ./audio/

# Test deepspeech
$ deepspeech --model deepspeech-0.8.2-models/deepspeech-0.8.1-
models.pbmm --scorer deepspeech-0.8.2-models/deepspeech-0.8.1-
models.scorer --audio ./audio/2830-3980-0043.wav

$ deepspeech --model deepspeech-0.8.2-models/deepspeech-0.8.1-
models.pbmm --scorer deepspeech-0.8.2-models/deepspeech-0.8.1-
models.scorer --audio ./audio/4507-16021-0012.wav

$ deepspeech --model deepspeech-0.8.2-models/deepspeech-0.8.1-
models.pbmm --scorer deepspeech-0.8.2-models/deepspeech-0.8.1-
models.scorer --audio ./audio/8455-210777-0068.wav
```

Examine the output of the last three commands, and you will see results *“experience proof less”*, *“why should one halt on the way”*, and *“your power is sufficient i said”* respectively. You are all set.

DeepSpeech API

The API is quite simple. You first need to create a model object using the model files you downloaded:

```
$ python3
>>> import deepspeech
>>> model_file_path = 'deepspeech-0.8.2-models/deepspeech-0.8.1-
models.pbmm'
>>> model = deepspeech.Model(model_file_path)
```

You should add a language model for better accuracy:

```
>>> scorer_file_path = 'deepspeech-0.8.2-models/deepspeech-0.8.1-
models.scorer'
>>> model.enableExternalScorer(scorer_file_path)

>>> lm_alpha = 0.75
>>> lm_beta = 1.85
>>> model.setScorerAlphaBeta(lm_alpha, lm_beta)

>>> beam_width = 500
>>> model.setBeamWidth(beam_width)
```

Once you have the model object, you can use either batch or streaming speech-to-text API.

Batch API

To use the batch API, the first step is to read the audio file:

```
>>> import wave
>>> filename = 'audio/8455-210777-0068.wav'
>>> w = wave.open(filename, 'r')
>>> rate = w.getframerate()
>>> frames = w.getnframes()
>>> buffer = w.readframes(frames)
>>> print(rate)
16000
>>> print(model.sampleRate())
16000
>>> type(buffer)
<class 'bytes'>
```

As you can see that the speech sample rate of the wav file is 16000hz, same as the model's sample rate. But the buffer is a byte array, whereas the DeepSpeech model expects 16-bit int array. Let's convert it:

```
>>> import numpy as np
>>> data16 = np.frombuffer(buffer, dtype=np.int16)
>>> type(data16)
<class 'numpy.ndarray'>
```

Run speech-to-text in batch mode to get the text:

```
>>> text = model.stt(data16)
>>> print(text)
your power is sufficient i said
```

Streaming API

Now let's accomplish the same using streaming API. It consists of 3 steps: open session, feed data, close session.

Open a streaming session:

```
>>> ds_stream = model.createStream()
```

Repeatedly feed chunks of speech buffer, and get interim results if desired:

```
>>> buffer_len = len(buffer)
>>> offset = 0
>>> batch_size = 16384
>>> text = ''
>>> while offset < buffer_len:
...     end_offset = offset + batch_size
...     chunk = buffer[offset:end_offset]
...     data16 = np.frombuffer(chunk, dtype=np.int16)
...     ds_stream.feedAudioContent(data16)
...     text = ds_stream.intermediateDecode()
...     print(text)
...     offset = end_offset
...
your power i
your power is suffi
your power is sufficient i said
your power is sufficient i said
```

Close stream and get the final result:

```
>>> text = ds_stream.finishStream()
>>> print(text)
```

```
your power is sufficient i said
```

Transcriber

A transcriber consists of two parts: a producer that captures voice from microphone, and a consumer that converts this speech stream to text. These two execute in parallel. The audio recorder keeps producing chunks of the speech stream. The speech recognizer listens to this stream, consumes these chunks upon arrival and updates the transcribed text.

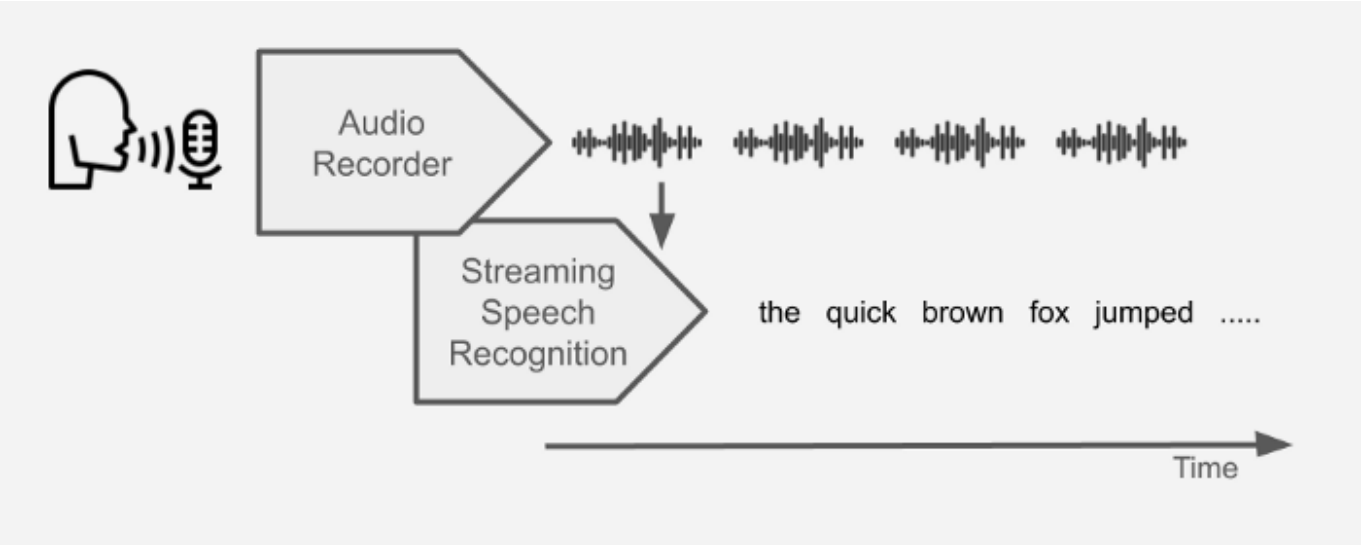

Image by author

To capture audio, we will use PortAudio, a free, cross-platform, open-source, audio I/O library. You have to download and install it. On macOS, you can install it using brew:

```
$ brew install portaudio
```

PyAudio is Python bindings for PortAudio, and you can install it with pip:

```
$ pip3 install pyaudio
```

PyAudio has two modes: blocking, where data has to read (pulled) from the stream; and non-blocking, where a callback function is passed to PyAudio for feeding (pushing) the

audio data stream. The non-blocking mechanism suits transcriber. The data buffer processing code using DeepSpeech streaming API has to be wrapped in a call back:

```
text_so_far = ''
def process_audio(in_data, frame_count, time_info, status):
    global text_so_far
    data16 = np.frombuffer(in_data, dtype=np.int16)
    ds_stream.feedAudioContent(data16)
    text = ds_stream.intermediateDecode()
    if text != text_so_far:
        print('Interim text = {}'.format(text))
        text_so_far = text
    return (in_data, pyaudio.paContinue)
```

Now you have to create a PyAudio input stream with this callback:

```
audio = pyaudio.PyAudio()
stream = audio.open(
    format=pyaudio.paInt16,
    channels=1,
    rate=16000,
    input=True,
    frames_per_buffer=1024,
    stream_callback=process_audio
)

print('Please start speaking, when done press Ctrl-C ...')
stream.start_stream()
```

Finally, you need to print the final result and clean up when a user ends recording by pressing Ctrl-C:

```
try:
    while stream.is_active():
        time.sleep(0.1)
except KeyboardInterrupt:
    # PyAudio
    stream.stop_stream()
    stream.close()
    audio.terminate()
    print('Finished recording.')
    # DeepSpeech
```

```
text = ds_stream.finishStream()
print('Final text = {}'.format(text))
```

That's all it takes, just 70 lines of Python code to put it all together: ds-transcriber.py.

Recap

In this article, you had a quick introduction to batch and stream APIs of DeepSpeech 0.8.2, and learned how to marry it with PyAudio to create a speech transcriber. The ASR model used here is for US English speakers, accuracy will vary for other accents. By replacing the model for other languages or accents, the same code will work for that language/accent.

Did you enjoy building it? Any feedback, improvements, suggestions? What other voice application would you like to build? Do let me know in the comments. Thanks for reading.

If you want to learn about other options for speech recognition, please check out:

> Speech Recognition with Python: Comparing 9 most prominent alternatives
>
> Learn which of the 9 most prominent automatic speech recognition engines is best for your needs, and how to use it in...
>
> medium.com

If you enjoyed this, please:

Originally published at Slang Labs blog.

UPDATE: 24 Aug 2020 — Updated to use Mozilla 0.8.2

Python Machine Learning TensorFlow Voice Assistant Artificial Intelligence