Creating transcription helper functions

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke

Machine Learning Engineer/YouTube Creator

datacamp

Exploring audio files

```
# Import os module
import os

# Check the folder of audio files
os.listdir("acme_audio_files")
```

```
(['call_1.mp3',
  'call_2.mp3',
  'call_3.mp3',
  'call_4.mp3'])
```

Preparing for the proof of concept

```
import speech_recognition as sr
from pydub import AudioSegment
# Import call 1 and convert to .wav
call_1 = AudioSegment.from_file("acme_audio_files/call_1.mp3")
call_1.export("acme_audio_files/call_1.wav", format="wav")
# Transcribe call 1
recognizer = sr.Recognizer()
call_1_file = sr.AudioFile("acme_audio_files/call_1.wav")
with call_1_file as source:
    call_1_audio = recognizer.record(call_1_file)
recognizer.recognize_google(call_1_audio)
```

Functions we'll create

- `convert_to_wav()` converts non-`.wav` files to `.wav` files.
- `show_pydub_stats()` shows the audio attributes of a `.wav` file.
- `transcribe_audio()` uses `recognize_google()` to transcribe a `.wav` file.

Creating a file format conversion function

```
# Create function to convert audio file to wav
def convert_to_wav(filename):
  "Takes an audio file of non .wav format and converts to .wav"
  # Import audio file
  audio = AudioSegment.from_file(filename)
  # Create new filename
  new_filename = filename.split(".")[0] + ".wav"
  # Export file as .wav
  audio.export(new_filename, format="wav")
  print(f"Converting {filename} to {new_filename}...")
```

Using the file format conversion function

```
convert_to_wav("acme_studios_audio/call_1.mp3")
```

```
Converting acme_audio_files/call_1.mp3 to acme_audio_files/call_1.wav...
```

Creating an attribute showing function

```
def show_pydub_stats(filename):
  "Returns different audio attributes related to an audio file."
  # Create AudioSegment instance
  audio_segment = AudioSegment.from_file(filename)
  # Print attributes
  print(f"Channels: {audio_segment.channels}")
  print(f"Sample width: {audio_segment.sample_width}")
  print(f"Frame rate (sample rate): {audio_segment.frame_rate}")
  print(f"Frame width: {audio_segment.frame_width}")
  print(f"Length (ms): {len(audio_segment)}")
  print(f"Frame count: {audio_segment.frame_count()}")
```

Using the attribute showing function

```
show_pydub_stats("acme_audio_files/call_1.wav")
```

```
Channels: 2
Sample width: 2
Frame rate (sample rate): 32000
Frame width: 4
Length (ms): 54888
Frame count: 1756416.0
```

Creating a transcribe function

```
# Create a function to transcribe audio
def transcribe_audio(filename):
  "Takes a .wav format audio file and transcribes it to text."
  # Setup a recognizer instance
  recognizer = sr.Recognizer()

  # Import the audio file and convert to audio data
  audio_file = sr.AudioFile(filename)
  with audio_file as source:
    audio_data = recognizer.record(audio_file)

  # Return the transcribed text
  return recognizer.recognize_google(audio_data)
```

Using the transcribe function

```
transcribe_audio("acme_audio_files/call_1.wav")
```

```
"hello welcome to Acme studio support line my name is Daniel how can I best help
you hey Daniel this is John I've recently bought a smart from you guys and I know
that's not good to hear John let's let's get your cell number and then we
can we can set up a way to fix it for you one number for 1757 varies how long do
you reckon this is going to take about an hour now while John we're going to try
our best hour I will we get the sealing member will start up this support case
I'm just really really really really I've been trying to contact 34 been put on
hold more than an hour and half so I'm not really happy I kind of wanna get this
issue 6 is fossil"
```

Let's practice!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Sentiment analysis on spoken language text

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke
Machine Learning Engineer/YouTube Creator

datacamp

Installing sentiment analysis libraries

```
$ pip install nltk
```

```
# Download required NLTK packages
import nltk
nltk.download("punkt")
nltk.download("vader_lexicon")
```

Sentiment analysis with VADER

```
# Import sentiment analysis class
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# Create sentiment analysis instance
sid = SentimentIntensityAnalyzer()
# Test sentiment analysis on negative text
print(sid.polarity_scores("This customer service is terrible."))
```

```
{'neg': 0.437, 'neu': 0.563, 'pos': 0.0, 'compound': -0.4767}
```

Sentiment analysis on transcribed text

```
# Transcribe customer channel of call_3
call_3_channel_2_text = transcribe_audio("call_3_channel_2.wav")
print(call_3_channel_2_text)
```

```
"hey Dave is this any better do I order products are currently on July 1st and I haven't received the product a three-week step down this parable 6987 5"
```

```
# Sentiment analysis on customer channel of call_3
sid.polarity_scores(call_3_channel_2_text)
```

```
{'neg': 0.0, 'neu': 0.892, 'pos': 0.108, 'compound': 0.4404}
```

Sentence by sentence

```
call_3_paid_api_text = "Okay. Yeah. Hi, Diane. This is paid on this call and obvi...
```

```
# Import sent tokenizer
from nltk.tokenize import sent_tokenize
# Find sentiment on each sentence
for sentence in sent_tokenize(call_3_paid_api_text):
  print(sentence)
  print(sid.polarity_scores(sentence))
```

Sentence by sentence

```
Okay.
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.2263}
Yeah.
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.296}
Hi, Diane.
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
This is paid on this call and obviously the status of my orders at three weeks ago,
and that service is terrible.
{'neg': 0.129, 'neu': 0.871, 'pos': 0.0, 'compound': -0.4767}
Is this any better?
{'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404}
Yes...
```

Time to code!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Named entity recognition on transcribed text

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke

Machine Learning Engineer/YouTube Creator

datacamp

Installing spaCy

```
# Install spaCy
$ pip install spacy
```

```
# Download spaCy language model
$ python -m spacy download en_core_web_sm
```

Using spaCy

```
import spacy


# Load spaCy language model
nlp = spacy.load("en_core_web_sm")
```

```
# Create a spaCy doc
doc = nlp("I'd like to talk about a smartphone I ordered on July 31st from your
Sydney store, my order number is 40939440. I spoke to Georgia about it last week.")
```

spaCy tokens

```
# Show different tokens and positions
for token in doc:
  print(token.text, token.idx)
```

```
I 0
'd 1
like 4
to 9
talk 12
about 17
a 23
smartphone 25...
```

spaCy sentences

```
# Show sentences in doc
for sentences in doc.sents:
  print(sentence)
```

```
I'd like to talk about a smartphone I ordered on July 31st from your Sydney store,
my order number is 4093829.
I spoke to one of your customer service team, Georgia, yesterday.
```

spaCy named entities

Some of spaCy's built-in named entities:

- PERSON People, including fictional.
- ORG Companies, agencies, institutions, etc.
- GPE Countries, cities, states.
- PRODUCT Objects, vehicles, foods, etc. (Not services.)
- DATE Absolute or relative dates or periods.
- TIME Times smaller than a day.
- MONEY Monetary values, including unit.
- CARDINAL Numerals that do not fall under another type.

spaCy named entities

```
# Find named entities in doc
for entity in doc.ents:
  print(entity.text, entity.label_)
```

```
July 31st DATE
Sydney GPE
4093829 CARDINAL
one CARDINAL
Georgia GPE
yesterday DATE
```

Custom named entities

```
# Import EntityRuler class
from spacy.pipeline import EntityRuler
```

```
# Check spaCy pipeline
print(nlp.pipeline)
```

```
[('tagger', <spacy.pipeline.pipes.Tagger at 0x1c3aa8a470>),
 ('parser', <spacy.pipeline.pipes.DependencyParser at 0x1c3bb60588>),
 ('ner', <spacy.pipeline.pipes.EntityRecognizer at 0x1c3bb605e8>)]
```

Changing the pipeline

```
# Create EntityRuler instance
ruler = EntityRuler(nlp)
```

```
# Add token pattern to ruler
ruler.add_patterns([{"label":"PRODUCT", "pattern": "smartphone"}])
```

```
# Add new rule to pipeline before ner
nlp.add_pipe(ruler, before="ner")
```

```
# Check updated pipeline
nlp.pipeline
```

Changing the pipeline

```
[('tagger', <spacy.pipeline.pipes.Tagger at 0x1c1f9c9b38>),
 ('parser', <spacy.pipeline.pipes.DependencyParser at 0x1c3c9cba08>),
 ('entity_ruler', <spacy.pipeline.entityruler.EntityRuler at 0x1c1d834b70>),
 ('ner', <spacy.pipeline.pipes.EntityRecognizer at 0x1c3c9cba68>)]
```

Testing the new pipeline

```
# Test new entity rule
for entity in doc.ents:
    print(entity.text, entity.label_)
```

```
smartphone PRODUCT
July 31st DATE
Sydney GPE
4093829 CARDINAL
one CARDINAL
Georgia GPE
yesterday DATE
```

Let's rocket and practice spaCy!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Classifying transcribed speech with Sklearn

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke
Machine Learning Engineer/YouTube creator

datacamp

Inspecting the data

```
# Inspect post purchase audio folder
import os
post_purchase_audio = os.listdir("post_purchase")
print(post_purchase_audio[:5])
```

```
['post-purchase-audio-0.mp3',
  'post-purchase-audio-1.mp3',
  'post-purchase-audio-2.mp3',
  'post-purchase-audio-3.mp3',
  'post-purchase-audio-4.mp3']
```

Converting to wav

```
# Loop through mp3 files
for file in post_purchase_audio:
  print(f"Converting {file} to .wav...")
  # Use previously made function to convert to .wav
  convert_to_wav(file)
```

```
Converting post-purchase-audio-0.mp3 to .wav...
Converting post-purchase-audio-1.mp3 to .wav...
Converting post-purchase-audio-2.mp3 to .wav...
Converting post-purchase-audio-3.mp3 to .wav...
Converting post-purchase-audio-4.mp3 to .wav...
```

Transcribing all phone call excerpts

```
# Transcribe text from wav files
def create_text_list(folder):
  text_list = []
  # Loop through folder
  for file in folder:
    # Check for .wav extension
    if file.endswith(".wav"):
      # Transcribe audio
      text = transcribe_audio(file)
      # Add transcribed text to list
      text_list.append(text)
  return text_list
```

Transcribing all phone call excerpts

```
# Convert post purchase audio to text
post_purchase_text = create_text_list(post_purchase_audio)
print(post_purchase_text[:5])
```

```
['hey man I just water product from you guys and I think is amazing but I leave a li
 'these clothes I just bought from you guys too small is there anyway I can change t
 "I recently got these pair of shoes but they're too big can I change the size",
 "I bought a pair of pants from you guys but they're way too small",
 "I bought a pair of pants and they're the wrong colour is there any chance I can ch
```

Organizing transcribed text

```
import pandas as pd
# Create post purchase dataframe
post_purchase_df = pd.DataFrame({"label": "post_purchase", "text": post_purchase_text})
# Create pre purchase dataframe
pre_purchase_df = pd.DataFrame({"label": "pre_purchase", "text": pre_purchase_text})
```

```
# Combine pre purchase and post purhcase
df = pd.concat([post_purchase_df, pre_purchase_df])
```

```
# View the combined dataframe
df.head()
```

Organizing transcribed text

```
   label                                                text
0  post_purchase  yeah hello someone this morning delivered a pa...
1  post_purchase  my shipment arrived yesterday but it's not the...
2  post_purchase  hey my name is Daniel I received my shipment y...
3  post_purchase  hey mate how are you doing I'm just calling in...
4   pre_purchase  hey I was wondering if you know where my new p...
```

Building a text classifier

```
# Import text classification packages
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split
```

```
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X=df["text"],
    y=df["label"],
    test_size=0.3)
```

Naive Bayes Pipeline

```
# Create text classifier pipeline
text_classifier = Pipeline([
  ("vectorizer", CountVectorizer()),
  ("tfidf", TfidfTransformer()),
  ("classifier", MultinomialNB())
])
```

```
# Fit the classifier pipeline on the training data
text_classifier.fit(X_train, y_train)
```

Not so Naive

```
# Make predictions and compare them to test labels
predictions = text_classifier.predict(X_test)
accuracy = 100 * np.mean(predictions == y_test.label)
print(f"The model is {accuracy:.2f}% accurate.")
```

```
The model is 97.87% accurate.
```

Let's practice!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Congratulations!

SPOKEN LANGUAGE PROCESSING IN PYTHON

Daniel Bourke

Machine Learning Engineer/YouTube creator

What you've done

1. Converted audio files into soundwaves with `Python` and `NumPy`.
2. Transcribed speech with `speech_recognition`.
3. Prepared and manipulated audio files using `PyDub`.
4. Built a spoken language processing pipeline with `NLTK`, `spaCy` and `sklearn`.

What next?

- Practice your skills with a project of your own.
- Check out `speech_recognition` 's `Microphone()` class.

One last transcription

```
one_last_transcription = transcribe_audio("congratulations.wav")


print(one_last_transcription)
```

```
Congratlutions on finishing the Spoken Language Processing with Python course!
You should be proud.
Now get out there and recognize some speech!
```

Keep learning!

SPOKEN LANGUAGE PROCESSING IN PYTHON