Summarizing Podcasts with Python

Jan 28, 2024 • 5 min read
Tags: Python, Speech Recognition, OpenAI, Whisper

Overview

I have been working on my **Podcast Player & Summarizer** project and I need to build the component for summarizing a podcast. The main application is built with Next.js but I need the summarizer code to be in Python so I can integrate it to my **Audio Transcriber API**.

In this blog post, I will walk you through the process of summarizing podcasts with Python. We will use the Whisper model for transcribing the podcast audio, and then we'll use the Chat Completions API to generate the summary.

Where to find the podcast MP3 file?

You can get the URL of the MP3 file from the RSS feed of your favorite podcast show. For example, one of my favorite podcast shows is Motley Fool Money and here is their RSS Feed.

If you're unable to get the RSS feed of your favorite shows, you can try to get the feed from other sources such as Apple Podcasts or search on Listen Notes.

Downloading the MP3 file

In this tutorial, we will download the MP3 from What Real Estate and Stock Investors Have In Common episode of the Motley Fool Money podcast.

Let's start by installing the required packages for the code. Install using `pip`:

```
pip install requests openai whisper tiktoken
```

The first step is to download the MP3 file and save it to local file:

```
import requests

url =
'https://www.podtrac.com/pts/redirect.mp3/pdst.fm/e/chrt.fm/track/1E7F5E/traf
fic.megaphone.fm/ARML5127344001.mp3?updated=1706200150'
```

```
filepath = 'podcast.mp3'

try:
    # Download the audio file
    r = requests.get(url, stream=True)
    r.raise_for_status()

    # Save the file to local disk
    with open(filepath, 'wb') as f:
        for chunk in r.iter_content(chunk_size=8192):
            f.write(chunk)

except (requests.HTTPError, requests.RequestException) as e:
    print(f'Failed to download file: {e}')
except (IOError) as e:
    print(f'Failed to save file: {e}')
```

We are using the `stream=True` when making the GET request to retrieve the data in chunks rather than load everything into memory at once. This is useful if the MP3 file is large. The downloaded audio then saved to *podcast.mp3* file in the current directory.

Transcribing the audio

We will use the OpenAI Whisper model to transcribe the downloaded MP3 file in 5 lines of code:

```
import whisper

model = whisper.load_model('base')
result = model.transcribe(filepath)
podcast_transcript = result['text']
```

It might take a while for the model to complete the transcribing process, depending on the size of the audio file.

Generating the summary

Now that we have the transcript of the podcast, we can create a prompt to generate the summary using the Chat Completions API.

Get your OpenAI API key and paste to the following code:

```
from openai import OpenAI, BadRequestError

openai = OpenAI(api_key='your_openai_api_key')
```

Create the prompt:

```
prompt = (
    'Summarize the following transcript from a podcast episode.\n'
    'Cover the main topics, insights, and any notable examples by the
speakers.\n'
    'The summary should be concise yet comprehensive.\n\n'
    'Here is the podcast transcript:\n'
    f'{podcast_transcript}'
)

tokenizer  = tiktoken.encoding_for_model('gpt-3.5-turbo-16k')
tokens = tokenizer.encode(prompt)
prompt = tokenizer.decode(tokens[:16000])
```

Keep in mind that each model has maximum number of tokens for the prompt. The `gpt-3.5-turbo-16k` model has maximum context window of 16,385 tokens. In the code above, we simply cut off the prompt into 16,000 tokens if the podcast transcript exceeds that number.

Now we can submit the prompt to the Chat Completions API and retrieve the summary:

```
try:
    completion = openai.chat.completions.create(
        model='gpt-3.5-turbo-16k',
        messages=[
            {'role': 'system', 'content': 'You are a helpful assistant.'},
            {'role': 'user', 'content': prompt}
        ]
    )

    content = completion.choices[0].message.content
    print(f'Summary: {content}')

except BadRequestError as e:
    print(f'Failed calling OpenAI API: {e}')
```

The result

Here is the result of the code from summarizing the “What Real Estate and Stock Investors Have In Common” episode on my local computer:

The podcast episode features Dave Meyer, Vice President of Growth and Analytics at Bigger Pockets, discussing his new book "Start with Strategy" with the host. The episode covers various aspects of real estate investing, emphasizing the importance of vision, self-knowledge, and strategic planning. The conversation delves into the data on rental properties, diversification in different markets, cash position considerations, and long-term investing challenges. Notable points include the need

for a clear vision, balancing risk and return, self-inventory for investors, and the importance of allocating time, skills, and capital in real estate deals. Meyer also offers insights on scaling strategies, the role of partnerships, and the need for periodic portfolio evaluation and reallocation. Throughout the episode, he highlights the need to treat investment portfolios as businesses. Meyer also reflects on personal mistakes, such as not outsourcing work earlier and efficiently reallocating capital for better performance. Ultimately, the episode emphasizes the significance of creating a business plan for real estate investing to align with personal goals and values, thus transforming investing and everyday life.

Key points:
- Importance of vision and strategic planning in real estate investing.
- Considerations for risk, return, and diversification across markets.
- The role of self-knowledge and self-inventory in investment decision-making.
- Scaling strategies, including vertical and horizontal approaches.
- Challenges and lessons learned, such as outsourcing and capital reallocation.
- Treating investment portfolios as businesses and the need for periodic evaluation and reallocation.

Overall it gives a very good summary and I am very satisfied with the result.