From Audio To Text Project - 'Programming with Python' Course

Dott. Zicca Davide

April 14, 2021

1 Motivazione e scelta modelli

L'iniziativa per questo progetto parte da un'idea più ampia di Stock Forecasting basato su un mio modello (che è attualmente in uno stage iniziale di progettazione) che tiene in considerazione una serie di variabili per cercare di catturare il *sentiment* del mercato.

Di assoluta necessità era poter convertire dei file audio (ad esempio: interviste di grandi hedge funders, CEO/CFO di multinazionali, interviste di Bloomberg o Thomson Reuters, interviste a direttori di Banche Centrali, etc.) in testo. L'idea è quella poi di passare dei testi, o piccoli gruppi di parole chiave, al mio modello, che successivamente in base a dei weights (da definire in base all'importanza del soggetto dell'intervista e al contenuto del testo stesso) proietterà l'andamento delle stocks, considerato un determinato intervallo di confidenza.

Tale progetto è in forte connessione ad un altro progetto che è anche in fase finale, ovvero il progetto di Data Clustering con il Professore Salvatore Daniele Tomarchio e il Professore Salvatore Ingrassia (tale progetto è frutto del corso *Data Analysis & Statistical Learning* a cui ho partecipato durante il Master in Data Science). L'idea è quella di visualizzare il numero di cluster all'interno di un dataset. L'obiettivo ultimo è quello di visualizzare ed estrapolare blocchi di termini che vengono ripetuti maggiormente all'interno di interviste (di cui sopra).

La curiosità, scaturita dall'idea di poter creare un modello di Stock Forecasting, mi ha spinto alla ricerca di metodologie efficaci per estrapolare da un file audio, delle informazioni di valore per effettuare un *tuning* del mio modello di Stock Forecasting [1].

I modelli scelti sono i seguenti: Baevski et al., 2020 e Zhang, 2017.

Il primo modello utilizzato è basato sul pacchetto 'SpeechRecognition' e presenta diverse API che si possono richiamare. Tuttavia, in questo progetto, il focus è stato fatto sull'algoritmo gratuito di Google (è presente anche un altro elaborato da Google che può essere utilizzato previa iscrizione).

Il secondo modello implementato utilizza diversi pacchetti: *jovian*, *Transformer*, *librosa* e *torch*. Usa un algoritmo di riconoscimento vocale sviluppato da *Facebook AI* (https://ai.facebook.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio/) e che presenta una serie di migliorie che si possono inserire per migliorare la performance. Cito, in particolare, la possibilità di *trainare* il modello inserendo vocaboli di test.

Riporto sinteticamente le operazioni eseguite per entrambi i modelli utilizzati:

[1] Altri metodi ed API, per convertire i file audio in testo, supportate dalla letteratura richiedevano costi che non giustificavano performance leggermente migliori rispetto ai due metodi che propongo in questo progetto.

- Caricamento delle librerie necessarie
- Caricamento del file audio (uguale per entrambi i modelli)
- Richiamo delle funzione necessarie
- *Print* della trascrizione del file audio

2 Modelli e implementazione

Riporto di seguito i codici eseguiti in *Jupyter Python Notebook* nei quali ho inserito alcuni commenti esplicativi delle operazioni eseguite:

[1]:
```
# Zicca Davide

# pip install SpeechRecognition
# https://pypi.org/project/SpeechRecognition/
```

[2]:
```
# Importo la libreria necessaria
import speech_recognition as sr
```

[3]:
```
sr.__version__
```

[3]:
```
'3.8.1'
```

[4]:
```
r = sr.Recognizer()
```

[5]:
```
# Importo File Audio
Jim_Simons = sr.AudioFile('Jim_Simons_trading_2.wav')
with Jim_Simons as source:
    audio = r.record(source)
```

[6]:
```
type(audio)
```

[6]:
```
speech_recognition.AudioData
```

[7]:
```
# Utilizzo il Google Speech Recognition. Le alternative (CMU Sphinx, Google␣
 ↪Cloud Speech API,
# Wit.ai, Microsoft Bing Voice Recognition, Houndify API, IBM Speech to Text,
# Snowboy Hotword Detection) richiedo dei token oppure l'iscrizione sul loro␣
 ↪sito proprietario
# per poter utilizzare i loro algoritmi di riconoscimento vocale.
# Tuttavia, il Google Speech Recognition risulta essere fra i più consistenti e␣
 ↪migliori da utilizzare.
# La presenza di funzioni (come 'adjust_for_ambient_noise' consente di␣
 ↪migliorare i risultati di
# conversione audio.
r.recognize_google(audio)
```

[7]:
```
'goodbye my background as a mathematician we managed funds was trading as the
trumpet by mathematical formulas we have only in a highly liquid publicly traded
securities meaning with your trade and credit default swaps and collateralized
debt obligations or some of those alphabet Soup things that George was just
referring to our training miles actually tend to be contrarian happened buying
stocks recently out of favour and selling those recently in favour'
```

[8]:
```
Jim_Simons = sr.AudioFile('Jim_Simons_trading_2.wav')
with Jim_Simons as source:
    r.adjust_for_ambient_noise(source) # in caso di audio con rumori di␣
 ↪sottofondo, tende a migliorarne la conversione
    audio2 = r.record(source)
```

[9]:
```
r.recognize_google(audio2)
```

[9]:
```
"ground as a mathematician we managed funds was trading as the trumpet by
mathematical formulas we have only and highly liquid publicly traded securities
meaning with don't trade and credit default swaps and collateralized debt
obligations or someone else alphabet Soup things that George was just referring
to our training models actually tend to be contrarian happened buying stocks
recently out of favour and selling those recently invited"
```

[10]:
```
Jim_Simons = sr.AudioFile('Jim_Simons_trading_2.wav')
with Jim_Simons as source:
    r.adjust_for_ambient_noise(source, duration=0.7)
    audio3 = r.record(source)
```

[11]:
```
r.recognize_google(audio3)
```

[11]:
```
'background as a mathematician we managed funds was trading as the trumpet by
mathematical formulas we are only in a highly liquid publicly traded securities
meaning with young trade and credit default swaps and collateralized debt
obligations or some of those alphabet Soup things that George was just referring
to our training models actually tend to be contrarian often buying stocks
recently out of favour and selling those recently in favour'
```

[12]:
```
r.recognize_google(audio, show_all=True) # mostra tutte le possibili␣
 ↪trascrizioni, piuttosto che mostrare
# solo quella scelta di default e ritenuta come migliore. In caso di file audio␣
 ↪poco chiari, potrebbe essere
# utile leggere tutte le possibili trascrizioni.
```

[12]:
```
{'alternative': [{'transcript': 'goodbye my background as a mathematician we
managed funds was trading as the trumpet by mathematical formulas we have only
in a highly liquid publicly traded securities meaning with your trade and credit
default swaps and collateralized debt obligations or some of those alphabet Soup
things that George was just referring to our training miles actually tend to be
```

```
contrarian happened buying stocks recently out of favour and selling those
recently in favour',
   'confidence': 0.94311428},
  {'transcript': 'goodbye my background as a mathematician we managed funds was
trading as the trumpet by mathematical formulas we have only in a highly liquid
publicly traded securities meaning with your trade and credit default swaps and
collateralized debt obligations or some of those alphabet Soup things that
George was just referring to our training miles actually tend to be contrarian
often buying stocks recently out of favour and selling those recently in
forever'},
  {'transcript': 'goodbye my background as a mathematician we managed funds was
trading as the trumpet by mathematical formulas we are only in a highly liquid
publicly traded securities meaning with your trade and credit default swaps and
collateralized debt obligations or some of those alphabet Soup things that
George was just referring to our training miles actually tend to be contrarian
happened buying stocks recently out of favour and selling those recently in
favour'},
  {'transcript': 'goodbye my background as a mathematician we managed funds was
trading as the trumpet by mathematical formulas we have only in a highly liquid
publicly traded securities meaning with your trade and credit default swaps and
collateralized debt obligations or some of those alphabet Soup things that
George was just referring to our training miles actually tend to be contrarian
happened buying stocks recently out of favour and selling those recently in
forever'},
  {'transcript': 'goodbye my background as a mathematician we managed funds was
trading as the trumpet by mathematical formulas we are only in a highly liquid
publicly traded securities meaning with your trade and credit default swaps and
collateralized debt obligations or some of those alphabet Soup things that
George was just referring to our training miles actually tend to be contrarian
often buying stocks recently out of favour and selling those recently in
forever'}],
 'final': True}
```

[13]:
```
# Utilizzo 'offset' e 'duration' per estrapolare una trascrizione riferita ad un␣
 ↪istante
# temporale di interesse.
# I valori inseriti in 'offset' e 'duration' sono da intendersi come secondi␣
 ↪della traccia audio.
Jim_Simons = sr.AudioFile('Jim_Simons_trading_2.wav')
with Jim_Simons as source:
    audio4 = r.record(source, offset=3, duration=2)

r.recognize_google(audio4)
```

[13]:
```
'we managed'
```

3 Metodo Alternativo

[14]:
```
# Installo jovian
# !pip install jovian --upgrade
# Installo Transformer
# !pip install -q transformers
# Installo librosa
# !pip install librosa
# Installo torch
# !pip install torch
```

[15]:
```
# Importo le libraries
import jovian
# Libreria per gestire i file audio
import librosa
# Importo Pytorch
import torch
# Importo Wav2Vec tokenizer
from transformers import Wav2Vec2ForCTC, Wav2Vec2Tokenizer
```

[16]:
```
# Carico l'audio di interesse
# Audio in formato 16 kHz
# https://www.youtube.com/watch?v=MExqWefHv2A&ab_channel=Pairtrading
# 'What is the Best Trading Strategy?' --> Jim Simons

import IPython.display as display
display.Audio("Jim_Simons_trading_2.wav", autoplay=False)
```

[16]:
```
<IPython.lib.display.Audio object>
```

[17]:
```
# Importo 'Wav2Vec pretrained model'
tokenizer = Wav2Vec2Tokenizer.from_pretrained("facebook/wav2vec2-base-960h")
model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-960h")
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\transformers\models\wav2vec2\tokenization_wav2vec2.py:356:
FutureWarning: The class `Wav2Vec2Tokenizer` is deprecated and will be removed
in version 5 of Transformers. Please use `Wav2Vec2Processor` or
`Wav2Vec2CTCTokenizer` instead.
  warnings.warn(
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at
facebook/wav2vec2-base-960h and are newly initialized:
['wav2vec2.masked_spec_embed']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[18]:
```
# Passo il file audio a 'librosa'
audio, rate = librosa.load("Jim_Simons_trading_2.wav", sr = 16000)
```

[19]:
```
# Mostro l'audio
audio
```

[19]:
```
array([ 0.        ,  0.        ,  0.        , ..., -0.00018311,
       -0.00018311, -0.00018311], dtype=float32)
```

[20]:
```
# Mostro il rate per assicurarmi che sia compatibile --> 16 kHz
rate
```

[20]:
```
16000
```

[21]:
```
# Funzioni necessarie per il modello Wav2Vec2
input_values = tokenizer(audio, return_tensors = "pt").input_values
```

[22]:
```
input_values
```

[22]:
```
tensor([[-8.9660e-05, -8.9660e-05, -8.9660e-05,  ..., -5.2166e-03,
         -5.2166e-03, -5.2166e-03]])
```

[23]:
```
# Storing logits (non-normalized prediction values) --> funzioni necessarie per␣
 ↪il modello Wav2Vec2
logits = model(input_values).logits
```

[24]:
```
# Storing predicted id's --> funzioni necessarie per il modello Wav2Vec2
prediction = torch.argmax(logits, dim = -1)
```

[25]:
```
# Passing the prediction to the tokenzer decode to get the transcription -->␣
 ↪funzioni necessarie per il modello Wav2Vec2
transcription = tokenizer.batch_decode(prediction)[0]
print(transcription) # Mostro la trascrizione del file audio di prova
```

```
VEN BY MY BACKGROUND AS A MATHEMATICIAN WE MANAGE FUNDS WHOSE TRADING IS
DETERMINED BY MATHEMATICALFONIOS WE OPERATE ONLY IN A HIGHLY LIQUID PUBLICLY
TRADED SECURITIES MEANING WE DON'T TRADE AND CREDIT THE FALSWAPS OR COLLATERIZED
DAD OBLIGATIONS OR SOME OF THOSE ALPHABITE SOUP THINGS THAT GEORGE WAS JUST
REFERRING TO OUR TRADING MODELS ACTUALLY TEND TO BE CONTRARIAN OFTEN BUYING
STOCKS RECENTLY OUT OF FAVOR AND SELLING NOSE RECENTLY IN FAVOR
```

4 Conclusione

Nonostante la scarsa qualità del file audio (caratteristica di mio interesse poichè spesso le interviste possono essere di bassa qualità a causa di una registrazione di bassa qualità o a causa di uno *slang* di difficile interpretazione per gli algoritmi), è possibile constatare una buona performance dei risultati ottenuti con le due metodologie. Il *goal* finale di testare più metodologie per estrapolare informazioni da file audio è stato raggiunto. Step successivi:

- Creare algoritmo di backtesting dei risultati ottenuti (comparando istantaneamente i risultati da 2 o più metodologie di conversione di file audio in testo o parole chiave)

- Inserimento degli estratti di testo nel modello di Stock Forecasting
- Tuning del modello

References

Baevski, Alexei et al. (2020). *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. arXiv: 2006.11477 [cs.CL].

Zhang, A. (2017). *Speech Recognition Python Package*. URL: https://github.com/Uberi/speech_recognition.