Sintesi e riconoscimento vocale con Python

(autore: Vittorio Albertoni)

Premessa

Sintesi vocale e riconoscimento vocale sono le tecnologie attraverso le quali possiamo interagire con una macchina utilizzando la voce.

Con la sintesi vocale otteniamo che un computer, ricevendo un input da file, produca un output verso un altoparlante imitando la voce umana. E' ciò che si chiama TTS, che sta per Text To Speech.

Con il riconoscimento vocale otteniamo che un computer, ricevendo un input espresso con voce umana, produca un output consistente in un testo scritto su file, come risultato di una dettatura. E' ciò che si chiama STT, che sta per Speech To Text.

Indice

1 Installare ciò che serve

Esistono tre pacchetti Python che ci possono divertire con la sintesi e con il riconoscimento vocale: PyAudio, SpeechRecognition e Pyttsx3.

Se lavoriamo su Linux o Mac prima di tutto occorre installare una libreria che difficilmente è installata di default sul computer, la libreria `portaudio19-dev`: senza di essa non riusciamo ad installare e a far funzionare PyAudio.
Su Linux la installiamo con
`sudo apt install portaudio19-dev`
oppure riccorrendo a Synaptic.
Su Mac la installiamo con
`brew install portaudio`.

Installiamo poi i pacchetti con `pip`, in questo ordine
`pip` (o `pip3`) `install pyaudio`
`pip` (o `pip3`) `install speechrecognition`
`pip` (o `pip3`) `install pyttsx3`

Dobbiamo, infine, installare un sintetizzatore vocale e possiamo utilmente scegliere eSpeak.

All'indirizzo *http://espeak.sourceforge.net/download.html* troviamo i compilati per Windows e Mac e il source per Linux.

Per Linux possiamo più semplicemente installare con
`sudo apt install espeak`
o attraverso il gestore di pacchetti Synaptic[1].

[1]Chi ha la fortuna di lavorare su Linux può utilizzare eSpeak da terminale utilizzando i comandi descritti all'indirizzo *http://espeak.sourceforge.net/commands.html*. Sempre sul sistema Linux, è possibile installare il componente di collegamento con Python con `sudo apt install python3-espeak`. In tal modo su può usare eSpeak come fosse un modulo Python, importandolo con `from espeak import espeak`.

2 TTS

Il modulo Python che rende possibile il Text To Speech sui tre più diffusi sistemi operativi è Pyttsx3[2].

All'indirizzo *https://pyttsx3.readthedocs.io/en/latest/engine.html* si trova il manuale in lingua inglese, utile per chi voglia andare oltre le istruzioni basilari di cui parlo qui.

Pyttsx3 usa per default la lingua inglese del Regno Unito e può essere configurato per usare un'altra lingua: tra le tante cito gli identificatori per le principali lingue europee (`italian`, `english`, `german`, `french`, `spanish`). Per l'inglese esiste la variante per gli USA (`english-us`).

Il modulo si importa con

```
import pyttsx3
```

Si costruisce un oggetto parlante, che possiamo chiamare recitante, con

```
recitante = pyttsx3.init()
```

Se scriviamo nella IDLE il nome di questo oggetto seguito da un punto ci viene mostrato l'elenco dei suoi metodi.
Tra questi i più importanti sono:
`say()` che accetta tra le parentesi tonde la stringa da recitare,
`runAndWait()` che dà il via alla recitazione.

E' quanto basta per ottenere la recitazione di una frase in lingua inglese UK.

Se vogliamo utilizzare un'altra lingua, prima di utilizzare il metodo `say()`, abbiamo a disposizione il metodo `setProperty()` che accetta tra le parentesi tonde il nome di una proprietà e il suo valore, separati da virgola.
Nel caso della lingua, la proprietà è la stringa `'voice'` e il valore è una stringa contenente l'identificatore della lingua.
Altra proprietà che possiamo utilmente modificare è la velocità di lettura espressa da un numero intero che indica le parole lette in un minuto. Il valore di default è 200 e va un tantino svelto.
In questo caso la proprietà è la stringa `'rate'` e il valore è un numero intero, per esempio 150 per rallentare un po' la velocità di lettura.

Esempi:

Il seguente script

```
import pyttsx3
recitante = pyttsx3.init()
recitante.say('lieutenant')
recitante.runAndWait()
```

ci fa sentire la strana pronuncia inglese UK della parola lieutenant.

Quest'altro

```
import pyttsx3
recitante = pyttsx3.init()
recitante.setProperty('voice', 'italian')
recitante.say('Ciao, Vittorio!')
recitante.runAndWait()
```

ci fa sentire un saluto in lingua italiana.

Quest'altro ancora ci recita i versi della canzone tedesca Lili Marleen contenuti in un file di testo, con una velocità un tantino inferiore a quella di default

```
import pyttsx3
recitante = pyttsx3.init()
recitante.setProperty('voice', 'german')
recitante.setProperty('rate', 160)
f = open('/home/vittorio/Documenti/lili_marleen.txt', 'r')
recitante.say(f.read())
recitante.runAndWait()
```

Se facciamo un po' di prove ci accorgiamo che la qualità della lettura non è gran che, soprattutto se lavoriamo non con semplici messaggi ma con testi di una certa dimensione e di una qualche espressività.

[2] In altra sede (allegato «tts_stt.pdf» al mio articolo «Software libero per sintesi e riconoscimento vocale» dell'aprile 2018 sul mio blog all'indirizzo *www.vittal.it*) ho presentato il modulo gtts, che utilizza il tts di Google e che funziona solo con collegamento internet attivo.

3 STT

Il modulo Python che rende possibile il Speech To Text sui tre più diffusi sistemi operativi è SpeechRecognition.

All'indirizzo *https://pypi.org/project/SpeechRecognition/* si trovano la library reference in lingua inglese e numerosi esempi. A questa documentazione può ricorrere chi voglia andare oltre le istruzioni basilari di cui parlo qui.

Al fine di non reinventare la ruota, SpeechRecognition si avvale di alcuni sistemi di riconoscimento vocale raggiungibili attraverso la rete: i due migliori sono quello di Google e quello di IBM. Qui mostrerò l'utilizzo di quello di Google.

Per questo motivo, perché SpeechRecognition funzioni, occorre utilizzarlo con collegamento internet attivo.

Il modulo si importa utilmente con la seguente formulazione

```
import speech_recognition as sr
```

Si costruisce l'oggetto di riconoscimento, che possiamo chiamare riconoscitore, con

```
riconoscitore = sr.Recognizer()
```

Se scriviamo nella IDLE il nome di questo oggetto seguito da un punto ci viene mostrato l'elenco dei suoi metodi.
Tra questi i più importanti sono:
`listen()` per catturare il segnale da microfono,
`record()` per catturare il segnale da file audio,
`recognize_google()` per convertire il segnale in testo scritto.

Il segnale viene catturato con una istruzione `with` che specifica la provenienza del segnale stesso, con questa sintassi
`with` <origine_ segnale> `as` <sorgente>:
<istruzione_di_cattura>
e il segnale così catturato viene convertito in testo, passando alla funzione `recognize_google()` come primo parametro quanto è stato catturato e come secondo parametro la lingua con la sintassi
`language =` <codice_ISO_lingua>
(rammento i codici per le principali lingue europee: `it`, `en`, `de`, `fr`, `es`).

Esempi:

Il seguente script accetta la dettatura di un testo, conferma quanto il computer ha capito e chiede se deve memorizzare quanto ha capito in un file di testo:

```
import speech_recognition as sr
riconoscitore = sr.Recognizer()
with sr.Microphone() as source:
    audio = riconoscitore.listen(source)
testo = riconoscitore.recognize_google(audio, language = "it")
print('Ho capito:  \n', testo)
scelta = input("Vuoi salvare in un file di testo?  (si/no)")
if scelta == 'si' or scelta == 'SI':
    f = open('/home/vittorio/Documenti/testo.txt', 'w')
    f.write(testo)
    f.close()
    print('Ho salvato nel file testo.txt.')
else:
    print('Non ho salvato il testo.')
print('Ciao')
```

Quest'altro fa le stesse cose ascoltando un file audio:

```
import speech_recognition as sr
riconoscitore = sr.Recognizer()
voce = sr.AudioFile('/home/vittorio/Documenti/testo.wav')
with voce as source:
    audio = riconoscitore.record(source)
testo = riconoscitore.recognize_google(audio, language = "it")
print('Ho capito:  \n', testo)
scelta = input("Vuoi salvare in un file di testo?  (si/no)")
```

```
if scelta == 'si' or scelta == 'SI':
    f = open('/home/vittorio/Documenti/testo.txt', 'w')
    f.write(testo)
    f.close()
    print('Ho salvato nel file testo.txt.')
else:
    print('Non ho salvato il testo.')
print('Ciao')
```

* * *

A conclusione di questo manualetto propongo questo esempio di dialogo tra uomo e computer: il computer chiede il nome all'interlocutore e, uditolo da microfono, recita un saluto personalizzato.

```
import pyttsx3
import speech_recognition as sr
recitante = pyttsx3.init()
recitante.setProperty('voice', 'italian')
recitante.say('Come ti chiami?')
recitante.runAndWait()
riconoscitore = sr.Recognizer()
with sr.Microphone() as source:
    audio = riconoscitore.listen(source)
testo = riconoscitore.recognize_google(audio, language = "it")
f = open('/home/vittorio/Documenti/saluto.txt', 'w')
f.write('Ciao' + testo)
f.close()
f = open('/home/vittorio/Documenti/saluto.txt', 'r')
recitante.say(f.read())
recitante.runAndWait()
```