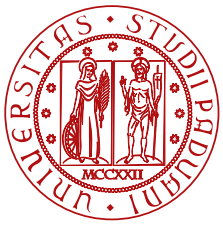


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Riconoscimento del parlato mediante OpenAI Whisper

Relatore
Prof. Beraldo Gloria

Laureando
Loda Enrico

ANNO ACCADEMICO 2023-2024

Data di laurea 19/07/2024

Sommario

Questa tesi si propone di implementare e analizzare un sistema di riconoscimento vocale in tempo reale in locale utilizzando OpenAI Whisper, un modello avanzato basato su tecniche di deep learning. Whisper rappresenta lo stato dell'arte nella comprensione del parlato umano e si distingue per essere un modello open source.

L'obiettivo principale è realizzare un sistema capace di effettuare una trascrizione in tempo reale in locale, con la prospettiva di poterlo applicare in contesti più ampi, ad esempio per l'interazione uomo-robot, o per la creazione di un chatbot basato sul linguaggio naturale.

Sono stati condotti test valutando l'accuratezza e la velocità dei vari modelli proposti da Whisper, con attenzione particolare sull'impatto della tecnologia CUDA sulla velocità della trascrizione.

I risultati ottenuti hanno evidenziato come non ci siano grosse differenze sulla qualità della trascrizione tra due modelli della stessa dimensione che utilizzano rispettivamente la CPU e i CUDA per l'inferenza, tuttavia è consigliabile possedere una GPU con tecnologia CUDA per garantire una trascrizione in italiano di qualità in tempo reale.

Indice

Capitolo 1

Introduzione

1.1 Cos'è il riconoscimento del parlato

Il riconoscimento del parlato è una tecnologia che consente ai computer di convertire il parlato umano in testo scritto. Questo processo ha rivoluzionato il modo in cui interagiamo con i dispositivi tecnologici, rendendo più semplice e veloce la comunicazione uomo-macchina.

Attualmente esistono diverse tecnologie che implementano il riconoscimento del parlato, tra cui chatbot come Siri di Apple e Amazon Alexa, software di traduzione vocale in tempo reale, sistemi di assistenza per persone con disabilità, software per autenticazione vocale e sistemi di trascrizione automatica di audio/video.

1.1.1 Componenti di uno speech-to-text

Per utilizzare uno speech-to-text sono necessari:

- un microfono per catturare il segnale audio
- un computer o dispositivo di elaborazione per processare l'audio e convertire il parlato in testo
- uno schermo o dispositivo di output per visualizzare il testo trascritto. In alcuni casi, può essere necessario un altoparlante per permettere al sistema di fornire una risposta vocale.

1.1.2 Panoramica di alcuni speech-to-text esistenti

Esistono già diversi speech-to-text implementati tramite API, ad esempio: Google Speech Recognition, CMUSphinx, Whisper API, Vosk API, IBM Speech to Text e Azure AI Speech. Si possono effettuare chiamate API facilmente a tutti gli speech-to-text appena citati utilizzando

la libreria SpeechRecognition. L'utilizzo di chiamate API comporta una minor riservatezza dei dati, garantendo però delle prestazioni indipendenti dall'hardware.

Non è invece altrettanto facile trovare sistemi di speech-to-text in locale che permettano la trascrizione in italiano, alcuni esempi sono DeepSpeech Italian Model e Coqui STT, che però non vengono più supportati. Whisper quindi sembra essere il miglior modello disponibile per una trascrizione in italiano in locale.

1.2 Scopo della tesi

L'obiettivo di questa tesi è creare un trascrittore audio testo in tempo reale, in grado di:

- operare in locale senza aver necessità di chiamate API per garantire la privacy
- aggiornare in modo dinamico il testo trascritto, senza avere necessità di terminare una frase prima di avere un riscontro sul terminale
- riconoscere e separare due frasi distinte tramite un invio
- possibilità di essere integrato per fare elaborazione sul testo trascritto

1.3 Struttura della tesi

La tesi è strutturata nei seguenti tre capitoli:

1.3.1 Capitolo 2

In questo capitolo si analizza Whisper, spiegando com'è stato allenato, come funziona e mettendo a confronto i diversi modelli forniti dal sistema con focus particolare sulla differenza tra trascrizione in inglese e in italiano. Si discuterà la piattaforma di calcolo in parallelo CUDA, citando alcune applicazioni, per poi concentrarci sulla libreria SpeechRecognition.

1.3.2 Capitolo 3

In questa sezione si commenterà il codice python del progetto, andando a esaminare le scelte implementative adottate sulla base dei risultati di alcuni test preliminari, soffermandoci su diversi parametri delle librerie utilizzate e sulle strutture dati implementate.

1.3.3 Capitolo 4

Infine discuteremo le modalità di testing e le metriche impiegate, mettendo a confronto i risultati ottenuti evidenziando le differenze di qualità della trascrizione tra modelli di dimensione differente, oltre che il vantaggio in tempistica dell'utilizzo della tecnologia CUDA rispetto all'inferenza su CPU.

Capitolo 2

Tecnologie e librerie utilizzate

2.1 Whisper

2.1.1 Cos'è Whisper

Whisper[1] è un sistema open source di riconoscimento vocale automatico (ASR) general-purpose addestrato su ∼680.000 ore di dati multilingue e multitask debolmente supervisionati raccolti dal web. L'uso di un set di dati così ampio e diversificato porta a una maggiore resistenza agli accenti, al rumore di fondo e al linguaggio tecnico. Inoltre consente il riconoscimento e la trascrizione in più lingue, oltre che la traduzione da queste lingue all'inglese.

È stato possibile allenare il modello su un dataset così grande perché è stato adottato un training "Large-Scale Weak Supervision"[1], ovvero prendendo diversi unlabeled dataset ed etichettandoli tramite euristiche, pattern recognition e propagazione di label. Questo ha permesso di ottenere un enorme dataset con del rumore, in quanto le label non saranno della stessa qualità di un normale dataset supervisionato, tuttavia abbiamo un trade-off positivo fornito dalla dimensione del training data[2].

Altri approcci esistenti utilizzano spesso set di dati di addestramento audio-testo più piccoli e più strettamente accoppiati o utilizzano un preallenamento audio ampio ma non supervisionato. Poiché Whisper è stato addestrato su un set di dati ampio e diversificato e non ha ricevuto fine-tuning specifico, non supera i modelli specializzati nelle prestazioni di LibriSpeech[2], un benchmark notoriamente competitivo nel riconoscimento vocale. Tuttavia, quando misuriamo le prestazioni zero-shot di Whisper su molti set di dati diversi, scopriamo che è molto più robusto e commette il 50% di errori in meno rispetto a questi modelli[2].

Circa un terzo del set di dati audio di Whisper non è in inglese e a Whisper viene affidato alternativamente il compito di trascrivere nella lingua originale o di tradurre in inglese,

[1] `https://github.com/openai/whisper`
[2] `https://paperswithcode.com/dataset/librispeech`

questo approccio è particolarmente efficace nell'apprendimento della traduzione dal parlato al testo. Whisper è allenato su un totale di ∼5.000 ore in italiano provenienti da diversi dataset: Multilingual LibriSpeech (MLS), Fleurs, VoxPopuli, Common Voice 9 e CoVOST 2.

2.1.2 Come funziona Whisper

Whisper da terminale accetta in input molteplici formati audio differenti e fornisce in output 5 file diversi: un file .txt con la trascrizione e i restanti file di formato diverso contenenti la trascrizione divisa in segmenti temporali per poter generare i sottotitoli in modo automatico.

L'architettura di Whisper si vede in Figura 2.1, si tratta di un semplice approccio end-to-end implementato come un encoder-decoder transformer. L'audio in ingresso viene suddiviso in segmenti da 30 secondi, convertito in uno spettrogramma log-Mel e quindi passato a un codificatore. Un decodificatore viene addestrato a predire la didascalia dell'audio corrispondente, insieme a dei token speciali che indirizzano il singolo modello a svolgere compiti quali l'identificazione della lingua, la marcatura temporale a livello di frase, la trascrizione multilingue del parlato e la traduzione del parlato in inglese.

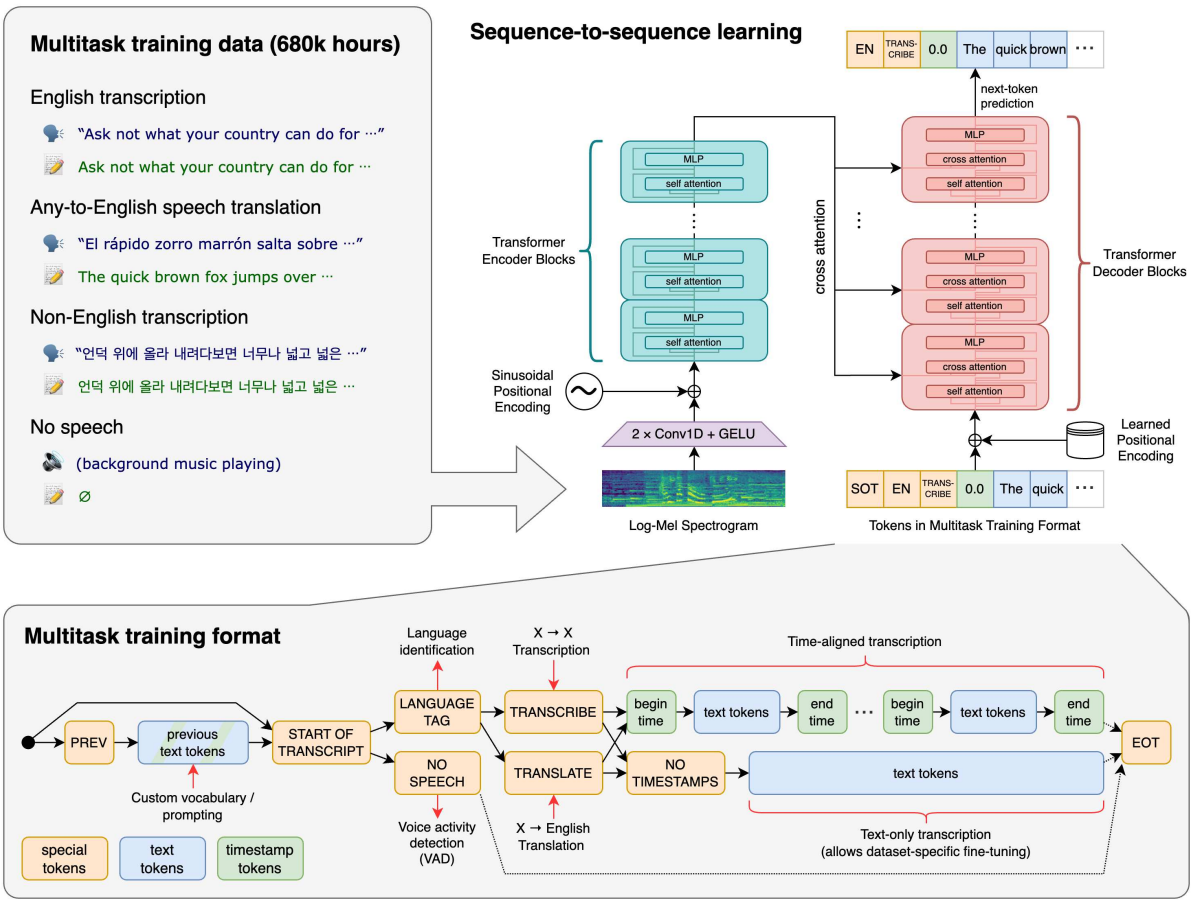


Figura 2.1: Architettura di Whisper

2.1.3 Modelli disponibili

Whisper offre 5 diverse dimensioni del modello, all'aumentare della dimensione del modello aumenta il numero di parametri utilizzati e l'accuracy mentre diminuisce la velocità della trascrizione. Il large-v2 ha la stessa architettura e lo stesso numero di parametri del large, tuttavia è stato allenato con 2.5x epochs e applicando SpecAugment[3], stochastic depth[4] e BPE drouput[5].

Di seguito troviamo la Tabella 2.1 che illustra le varie dimensioni dei modelli offerti da Whisper, con il corrispettivo numero di parametri e velocità relativa. Il parametro velocità relativa viene utilizzato per confrontare le tempistiche dei modelli di diversa dimensione, si ricava calcolando il quoziente tra durata audio e tempo per la trascrizione.

Dimensione	Milioni di parametri	velocità relativa
tiny	39	∼32x
base	74	∼16x
small	244	∼6x
medium	769	∼2x
large	1550	∼1x
large-v2	1550	∼1x

Tabella 2.1: Modelli e velocità relativa da github

Il Word Error Rate (WER) è una metrica classica per valutare modelli di speech recognition, indica la percentuale di parole trascritte errate dal modello. Più è basso il valore meno errori troveremo nella trascrizione.

Le Tabelle 2.2, 2.3 confrontano il Word Error Rate ottenuto dai diversi modelli tra l'inglese e l'italiano, in relazione al dataset sul quale vengono valutati.

Tabella 2.2: ↓ WER% per l'inglese [2]

Dimensione	Librispeech	Commonvoice9	VoxPopuli	FLEURS
tiny	15.7	28.8	11.6	12.4
base	11.7	21.9	9.5	8.9
small	8.3	14.5	8.2	6.1
medium	6.8	11.2	7.6	4.4
large	6.3	10.1	7.2	4.5
large-v2	6.2	9.4	7.0	4.2

Dimensione	Librispeech	Commonvoice9	VoxPopuli	FLEURS
tiny	41.7	44.5	40.5	29.8
base	31.1	30.5	30.8	17.9
small	21.4	16.0	22.9	9.8
medium	16.0	9.4	19.7	5.2
large	14.3	8.1	18.4	4.2
large-v2	13.8	7.1	19.0	4.0

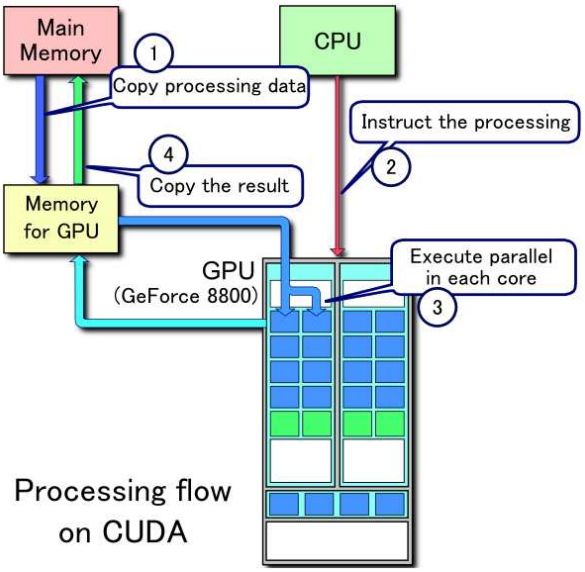
Tabella 2.3: ↑ WER% per l'italiano [2]

2.2 CUDA

La tecnologia CUDA (Compute Unified Device Architecture) è una piattaforma di calcolo parallelo sviluppata da NVIDIA che sfrutta la potenza di elaborazione di determinate unità di processing grafico della GPU per effettuare general-purpose processing (GPGPU).

Grazie alla parallelizzazione dei calcoli offerta da CUDA è stato possibile ridurre notevolmente i tempi di addestramento di Whisper, inoltre è possibile sfruttare i CUDA core della GPU per effettuare l'inferenza, consentendo di ottenere trascrizioni in tempi più rapidi sfruttando il calcolo in parallelo.

La tecnologia CUDA trova applicazione in diversi ambiti oltre al machine learning, tra cui: chimica computazionale, previsione meteorologica, bioinformatica, fluidodinamica computazionale (CFD) e rendering grafico 3D.



2.3 Speechrecognition

La libreria SpeechRecognition[3]è una libreria open-source in Python che permette di riconoscere il parlato umano e trasformarlo in testo. Normalmente SpeechRecognition, oltre che permettere di registrare e gestire facilmente l'audio in input da un microfono, costituisce un involucro dal quale è possibile utilizzare diverse API per effettuare il riconoscimento vocale, come Google Speech Recognition, IBM Speech to Text, Nuance Speech Recognition, e altre ancora.

In questa tesi è stata utilizzata per: accedere al microfono, creare un thread per ascolto in background e segmentare in chunk l'input audio.

[3]`https://pypi.org/project/SpeechRecognition/`

Capitolo 3

Implemetazione

3.1 Setup iniziale

3.1.1 Setting delle librerie

Questa sezione è dedicata alla spiegazione del setup delle librerie, le quali:

- Numpy[1], impiegata per la conversione dei dati in RAM in un formato accettabile come input audio da Whisper, in questo caso come una PCM 32-bit floating point normalizzata
- SpeechRecognition[2], utilizzata per: accedere al microfono, creare un thread in background per l'acquisizione dell'audio e segmentare l'input in chunk da 6 secondi
- Queue[3], impiegata per la creazione di una thread-safe queue di audio da mantenere in RAM: creare una coda in RAM ci permette di non dover effettuare scritture o letture in memoria secondaria che imporrebbero un ulteriore delay, con una thread-safe queue si intende una coda che garantisce un accesso sicuro a threads diversi non sincronizzati
- os[4], utilizzata per ripulire l'output della console
- keyboard[5] per rilevare determinati input della tastiera
- time[6] per determinare il tempo trascorso tra due frasi e per evitare cicli infiniti
- torch[7] per rilevare la presenza di hardware CUDA

[1] https://numpy.org/
[2] https://pypi.org/project/SpeechRecognition/
[3] https://docs.python.org/3/library/queue.html
[4] https://docs.python.org/3/library/os.html
[5] https://pypi.org/project/keyboard/
[6] https://docs.python.org/3/library/time.html
[7] https://pypi.org/project/torch/

A seguire il codice per l'import di tutte le librerie:

```
import speech_recognition as sr
import numpy as np
import Whisper
from queue import Queue
import os
import torch
import time
import keyboard
```

3.1.2 Parametri SpeechRecognition

Dalla documentazione di SpeechRecognition si definisce il Recognizer, un oggetto che contiene una collezione di parametri e funzionalità utili. Il parametro record_timeout indica la dimensione in secondi dei chunk audio ottenuti dal Recognizer di SpeechRecognition, 6 secondi è una dimensione arbitraria che secondo i risultati dei test preliminari sembra essere un buon valore per evitare di tagliare la frase garantendo comunque un'interazione reattiva.

Dopo aver creato un'istanza di Recognizer, disabilitiamo il dynamic_energy_threshold: se questo parametro fosse impostato su True l'energy threshold per l'attivazione del microfono cambierebbe col passare del tempo, cosa non voluta per evitare l'abbassamento eccessivo dell'energy threshold in caso di silenzi prolungati.

Si imposta ora il microfono utilizzato come sorgente con una frequenza di campionamento standard di 16 kHz. Per impostare l'energy threshold sfruttiamo la funzione adjust_for_ambient_noise che lo imposta in automatico basandosi sul rumore rilevato dal microfono. Di seguito viene riportato la porzione di codice di interesse:

```
#Dimensione chunk audio
record_timeout = 6
#Utilizziamo SpeechRecognizer per registrare l'audio perché offre una
#funzionalità interessante che consente di rilevare la fine del parlato.
recorder = sr.Recognizer()
#La compensazione dinamica dell'energia abbassa drasticamente la soglia
#di energia al punto da rendere recorder sempre attiva.
recorder.dynamic_energy_threshold = False
source = sr.Microphone(sample_rate=16000)
```

```
with source:
    recorder.adjust_for_ambient_noise(source)
```

3.1.3 Creazione e caricamento del modello, setting di parametri aggiuntivi

Il parametro phrase_timeout imposta i secondi di silenzio necessari per terminare una frase e iniziare la successiva. Il parametro phrase_time sarà utilizzato nel corpo del codice per determinare il tempo trascorso dall'ultima frase rilevata.

Come mostrato nel codice a seguire, si può quindi creare e caricare il modello di Whisper scelto (in questo caso quello di taglia media), inoltre si imposta un vettore di stringhe chiamato transcription in cui verrà salvato l'output del modello.

```
#Quantità di silenzio tra le registrazioni prima di iniziare
#una nuova riga nella trascrizione.
phrase_timeout = 3
#L'ultima volta in cui una registrazione è stata recuperata dalla queue.
phrase_time = None
#Carica il modello Whisper di dimensione "medium".
model =Whisper.load_model("medium")
transcription = ['']
```

3.1.4 Creazione queue e thread in background

Si definisce la funzione record_callback per inserire i dati rilevati dal microfono in una thread-safe queue, ovvero una struttura dati che garantisca scrittura e lettura di dati sulla memoria principale e permetta inoltre l'accesso in modo sicuro anche a thread non sincronizzati.

È quindi possibile creare un thread in background per permettere la costante collezione di dati audio mentre il programma continua l'esecuzione, ora basterà invocare stop_listening per terminare il Recognizer in background. Di seguito la porzione di codice interessata:

```
#Coda thread-safe per il passaggio di dati con record_callback.
data_queue = Queue()
def record_callback(_, audio:sr.AudioData) -> None:
    #Estrae i byte dall'oggetto audio.
    data = audio.get_raw_data()
```

```
    #Inserisce i dati nella coda thread-safe.
    data_queue.put(data)

#Crea un thread in background che gestirà la registrazione
#e invierà i dati grezzi alla coda.
stop_listening = recorder.listen_in_background(source, record_callback,
                 phrase_time_limit=record_timeout)
#Avvisa l'utente che il setup è terminato
print ("model loaded, start talking \n")
```

3.2 Trascrizione in real-time

Nelle sezioni precedenti è stato introdotto il setup principale, ora passiamo al vero corpo del codice, ovvero la trascrizione effettiva dei chunk audio, oltre alla gestione del print su terminale e alla risposta di determinati input da tastiera per mutare o terminare il programma.

Da questo punto fino alla terminazione del programma tutto il codice mostrato sarà contenuto all'interno di un while.

3.2.1 Conversione dei dati e trascrizione con Whisper

Quando è presente data audio nella queue, confrontiamo il tempo passato dalla frase precedente con il nostro parametro phrase_timeout per valutare se è una nuova frase o la continuazione della precedente.

Quindi si salva nella variabile audio_data la queue trovata e, dopo aver fatto il clear della queue, convertiamo il buffer in RAM in un formato che Whisper accetta come una PCM 32-bit floating point normalizzata. Dividiamo per 32768 perché i valori di un file audio codificato in formato int16 vanno da -32768 a 32767, in questo modo normalizziamo i valori dell'array audio_np.

Possiamo quindi trascrivere audio_np con il nostro modello, nei parametri devono essere specificati lingua e fp16, che indica di utilizzare i weights del modello come floating point a 16 bit per poter effettuare l'inferenza tramite CUDA e non con la CPU.

Una volta terminata l'elaborazione, Whisper ritorna un dizionario. Il testo trascritto viene salvato solo dopo essere stato processato con la funzione strip per eliminare spazi ridondanti. A seguire la porzione di codice interessata:

```
print("\n*press m to mute*\nTranscription:")
while True:
   try:
       now = time.time()
       #Recupera l'audio grezzo registrato dalla coda.
       if not data_queue.empty():
           print ('trovata queue, elaboro...')
           phrase_complete = False
           #Se è trascorso abbastanza tempo tra le registrazioni,
           #considera la frase completata.
           if phrase_time and now - phrase_time > phrase_timeout:
               phrase_complete = True
           #Salviamo il tempo di arrivo della registrazione corrente.
           phrase_time = now

           #Combina i dati audio dalla coda con byte join.
           audio_data = b''.join(data_queue.queue)
           data_queue.queue.clear()

           #Converte i dati da interi a 16 bit in virgola mobile a 32 bit
           #Normalizza considerando una PCM di 32768 Hz massimo.
           audio_np = np.frombuffer(audio_data,
                     dtype=np.int16).astype(np.float32) / 32768.0

           #Effettua e salva la trascrizione.
           result = model.transcribe(audio_np, language="it",
                    fp16=torch.cuda.is_available())
           text = result['text'].strip()
```

3.2.2 Gestione print sul terminale

Il codice riportato a seguire mostra come venga aggiunto un elemento in coda al vettore transcription in caso di nuova frase, altrimenti si aggiorna l'ultima frase salvata modificando l'ultimo elemento di transcription. Quindi si ripulisce il terminale e si stampa il vettore transcription aggiornato.

```
            #Aggiungi una nuova riga alla trascrizione
            #in caso di pausa sufficiente.
            if phrase_complete:
                transcription.append(text)
            #Altrimenti modifica quella esistente.
            else:
                transcription[-1] = transcription[-1]+" "+text

            #Cancella la console per stampare la trascrizione aggiornata.
            os.system('cls')
            for line in transcription:
                print(line)
            #Svuota lo stdout per evitare righe sovrapposte.
            print('', end='', flush=True)
        else:
            #Per evitare cicli infiniti.
            time.sleep(0.5)
```

3.2.3 Implementazione del mute e terminazione programma

Questo codice gestisce l'interazione con l'utente durante la registrazione audio e la trascrizione del testo. L'utente può terminare con 'spazio' e con 'Ctrl+C', può inoltre mutare temporaneamente il programma con 'm' e farlo ripartire con 'u'. Di seguito la porzione di codice interessata:

```
    #Siamo ancora all'interno del while.
    #Gestione delle interruzioni da tastiera (Ctrl+C).
    except KeyboardInterrupt:
        data_queue.queue.clear()
        break
    #Gestione terminazione programma premendo spazio.
    if keyboard.is_pressed("space"):
        data_queue.queue.clear()
        print("Stopping recording, wait...")
        time.sleep(0.2)
```

```
        break
    #Gestione del mute con il tasto "m"
    if keyboard.is_pressed("m"):
        data_queue.queue.clear()
        print("muted...muted...")
        print("*press u to unmute*")
        #Invocare stop_listening termina il recorder in background.
        stop_listening(wait_for_stop=False)
        time.sleep(0.5)
        while not keyboard.is_pressed("u"):
            time.sleep(0.2)
        stop_listening = recorder.listen_in_background(source,
                        record_callback, phrase_time_limit=record_timeout)
        print("unmuted, start talking")
        time.sleep(0.2)

#Una volta terminato il programma, stampiamo l'intera trascrizione.
print(transcription)
```

Il codice completo è presente su github.

Capitolo 4

Test effettuati

Questo capitolo è dedicato alla presentazione dei test eseguiti e dei relativi risultati.

4.1 Setup per la fase di testing

I test sono stati eseguiti su un computer con le seguenti specifiche:

- intel i5-9600k, 4.75 GHz, 6 core
- NVIDIA 3070, 8 GB VRAM, 5888 CUDA Cores
- RAM 16 GB, 3000 MHz, DDR4

4.2 Modalità di testing

Inizialmente si è preparato un dataset di 4 audio registrati dal medesimo, chiamati rispettivamente: ricetta, batteria, vestiario e lettura. I primi 3 sono lunghi 36 secondi mentre il rimanente è di 53 secondi. Il più lungo è l'unico creato leggendo da un copione, i restanti sono infatti improvvisati per simulare un discorso e non una dettatura. L'audio 'ricetta' è stato registrato da una donna per poter valutare una voce differente.

La fase di testing consiste nel utilizzare il codice sviluppato per far trascrivere i suddetti audio, salvando le trascrizioni ottenute con le relative tempistiche in un file csv tramite l'omonima libreria. Per valutare la trascrizione ottenuta da Whisper essa sarà confrontata con una trascrizione manuale che verrà considerata ground truth. Nei risultati riportati di seguito, con modello large si intende il modello large-v2.

I test sono stati strutturati per evidenziare l'impatto su tempistiche e precisione della trascrizione al variare di due parametri fondamentali:

- la dimensione del modello
- l'utilizzo dei CUDA core o della CPU per l'inferenza.

4.2.1 Metriche utilizzate

La precisione sarà calcolata con la seguente formula, considerando il numero di parole della trascrizione ground truth:

$$precisione = \frac{\#parole - \#errori}{\#parole} * 100$$

Il numero di errori sarà calcolato contando le differenze con la ground truth considerando anche la punteggiatura. Considereremo +1 errore in caso di parola errata/mancante e +0.5 errore per punteggiatura sbagliata o in caso di solo una lettera errata (e.g., cane/cani).

Per valutare la tempistica si calcoleranno i secondi necessari a scrivere una lettera, considerando il numero di caratteri della trascrizione di Whisper con la formula: $\frac{tempo\_trascrizione}{\#caratteri}$, dove per il tempo della trascrizione si considera la somma del tempo di trascrizione in secondi di ogni chunk, mentre per il numero di caratteri non vengono contati spazi o segni di punteggiatura.

A questo punto si può approssimare la velocità relativa di un determinato modello sulla mia macchina calcolando:

$$relative\_speed = \frac{lunghezza\_audio}{tempo\_trascrizione}$$

La velocità relativa normalmente permette di stimare il tempo di attesa di trascrizione di un audio lungo x tramite la formula:

$$tempo\_attesa = \frac{x}{relative\_speed}$$

Dato che si ha la conoscenza a priori della durata di un chunk audio, ovvero record_timeout, si può stimare il delay medio di trascrizione di una frase come:

$$delay\_medio = \frac{record\_timeout}{relative\_speed}$$

4.3 Risultati ottenuti

Dai test effettuati utilizzando CUDA si può confermare la velocità relativa $\sim$1x del large e $\sim$6x dello small dalla Tabella 2.1 , risulta invece maggiormente performante il medium che raggiunge una velocità relativa di $\sim$4x.

Data la grande latenza del large e la bassa precisione dello small, l'implementazione dello speech-to-text su questa macchina sfruttando la tecnologia CUDA risulta ottimale col modello medium, garantendo una buona qualità di trascrizione e un basso delay medio di $\frac{record\_timeout}{4} \approx 1{,}5s$.

Sfruttando invece la CPU per l'inferenza otteniamo una precisione paragonabile ai test precedenti a parità di dimensione del modello, tuttavia notiamo un significativo peggioramento delle tempistiche: il tempo di trascrizione del medium in queste condizioni risulta leggermente superiore ma comunque paragonabile alle tempistiche del large ottenute in precedenza, mentre lo small necessita di tempistiche pare al doppio rispetto all'uso del modello medium con CUDA.

Con questi rilevamenti possiamo concludere che non è possibile far girare in modo ottimale lo speech-to-text su questa macchina senza sfruttare CUDA, ed è consigliabile una scheda video predisposta a CUDA per poter utilizzare efficacemente il programma in locale.

audio	tempo trascrizione(s)	#lettere	secondi/lettera	%precisione
ricetta	37.011	360	0.103	96.83
batteria	40.194	380	0.105	96.58
vestiario	40.056	370	0.108	92.21
lettura	54.490	462	0.118	98.19

Tabella 4.1: Risultati del modello large utilizzando CUDA

audio	tempo trascrizione(s)	#lettere	secondi/lettera	%precisione
ricetta	9.499	382	0.025	89.24
batteria	8.801	379	0.023	92.47
vestiario	9.337	371	0.025	82.47
lettura	11.77	463	0.025	96.39

Tabella 4.2: Risultati del modello medium utilizzando CUDA

audio	tempo trascrizione(s)	#lettere	secondi/lettera	%precisione
ricetta	43.295	361	0.120	87.34
batteria	44.267	375	0.118	91.10
vestiario	42.929	357	0.120	79.87
lettura	65.359	466	0.140	96.39

Tabella 4.3: Risultati del modello medium senza utilizzare CUDA

audio	tempo trascrizione(s)	#lettere	secondi/lettera	%precisione
ricetta	5.875	392	0.015	79.75
batteria	5.647	369	0.015	78.76
vestiario	5.534	351	0.016	73.38
lettura	7.545	461	0.016	86.14

Tabella 4.4: Risultati del modello small utilizzando CUDA

audio	tempo trascrizione(s)	#lettere	secondi/lettera	%precisione
ricetta	17.156	387	0.044	80.34
batteria	17.921	376	0.048	80.82
vestiario	16.334	344	0.048	72.73
lettura	22.529	460	0.049	81.93

Tabella 4.5: Risultati del modello small senza utilizare CUDA

4.4 Tabella di confronto tra le trascrizioni

Nella tabella sottostante sono riportate le trascrizioni al fine di confrontare i vari modelli. Le parole in arancione corrispondono a errori di peso 0.5 (punteggiatura / solo una lettera errata) mentre le parole in rosso corrispondono a errori di peso 1 (parola errata / mancante / allucinata).

audio \ model	large	medium	medium NO CUDA	small	small NO CUDA	ORIGINALE
ricetta	Come fare una buona parmigiana di zucchine? Io ho provato, ho fatto le zucchine alla griglia, ho cotto il pomodoro, ho cotto le... no scusa, ho messo la provola perché non avevo la mozzarella, mi è rimasta un pochino bassa per i miei probabilmente dovevo fare qualche strato in più. Inoltre la mozzarella perde tanto liquido, come posso fare? L'ho schiacciata anche con le mani ma usando prodotto fresco purtroppo dopo la cottura è risultato molto liquido. Grazie!	come fare una parmigiana di zucchine? io ho provato, ho fatto le zucchine alla griglia cotto il pomodoro o cotto le... No scusa, ho messo la provola perché non avevo la mozzarella, mi è rimasta un po' più. bassa per i miei gusti probabilmente dovevo fare qualche strato in più inoltre la mozzarella perde tanto liquido. Come posso fare? L'ho schiacciata anche con le mani ma usando il profondo di spettacolo. fresco purtroppo dopo la cottura risultato molto liquido. Grazie.	come fare una parmigiana di zucchine? io ho provato, ho fatto le zucchine alla griglia con il pomodoro o con le... scusa ho messo la provola perché non avevo la mozzarella mi è rimasta un pochino bassa per i gusti probabilmente dovevo fare qualche strato in più inoltre la mozzarella perde tanto liquido come posso fare l'ho schiacciata anche con le mani ma usando il prodotto fresco purtroppo dopo la cottura risultato molto liquido. Grazie	come fare una buona parmigiana di zucchine? io ho provato, ho fatto le zucchine alla griglia un cotto di pomodoro o cotto di... No, scusa, ho messo la provola perché non avevo la mozzarella. Mi è rimasta un po' più... bassa per i miei gusti probabilmente dovrò fare qualche stratto in più. Inoltre la mozzarella per te perde tanto liquido. Come posso fare? L'ho schiacciata anche con le mani ma usando proprio la mano. il prodotto fresco, purtroppo dopo la cottura il risultato è molto liquido.	come fare una buona parmigiana di zucchine? io ho provato, ho fatto le zucchine la griglia un cotto di pomodoro o cotto di... No, scusa, ho messo la provola perché non avevo la mozzarella, mi è rimasta un po' bassa per i miei gusti probabilmente dovrò fare qualche stratto in più. Inoltre la mozzarella con la cipolla per te perde tanto liquido. Come posso fare? L'ho schiacciata anche con le mani ma usando per te. fresco, purtroppo dopo la cottura il risultato è molto liquido. Grazie!	Come fare una buona parmigiana di zucchine? Io ho provato, ho fatto le zucchine alla griglia, ho cotto il pomodoro, ho cotto le... no scusa, ho messo la provola perché non avevo la mozzarella, mi è rimasta un pochino bassa per i miei gusti, probabilmente dovevo fare qualche strato in più. Inoltre la mozzarella perde tanto liquido. Come posso fare? L'ho schiacciata anche con le mani ma usando prodotto fresco purtroppo dopo la cottura è risultato molto liquido. Grazie.
batteria	L'altro giorno ho provato a cambiare lo schermo del mio vecchio cellulare. Tuttavia, ho avuto il problema di staccare la batteria. La batteria era infatti incollata in modo particolarmente aggressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcol isopropilico ma non è bastato e ho spaccato la batteria facendo leva Per la prossima volta, quali sostanze vi consigli di utilizzare per sciogliere la colla che tiene la batteria alla scossa?	L'altro giorno ho provato a cambiare lo schermo del mio vecchio cellulare. Tuttavia ho avuto il problema di staccare la batteria. La batteria era infatti incollata in modo particolarmente adressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcol isopopilico. ma non è bastato e ho spaccato la battiglia facendo leva Per la prossima volta, quali sostanze vi consiglio di utilizzare per sciogliere la colla che tiene la batteria alla sua?	L'altro giorno ho provato a cambiare lo schermo del mio vecchio celulare. Tuttavia ho avuto il problema di staccare la batteria. La batteria era infatti incollata in modo particolarmente adressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcol isocopili, ma non è bastato e ho spaccarla a batteria facendo leva. Per la prossima volta, quale sostanza vi consiglio di utilizzare per sciogliere la colla che tiene la batteria alla sua?	L'altro giorno ho provato a cambiare lo schermo degli vecchi cellulare. tuttavia ho avuto il problema di staccare la batteria. la batteria è infatti incollata in modo particolarmente addressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcolisopopili. ma non è bastato e ho spaccare la battella facendo leva. Per la prossima volta, quali sostanzi devo segli di utilizzare per sciogliere la colla che ti navatteria la scuola?	L'altro giorno ho provato a cambiare lo schermo del mio vecchio cennudale. tuttavia ho avuto il problema di staccare la batteria. la batteria è infatti incollata in modo particolarmente addressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcolisocopili. ma non è bastato e ho spaccare la battiglia facendo leva. Per la prossima volta, quali sostanzi dovresti utilizzare per sciogliere la colla che ti nabattevi alla scuola? scopca	L'altro giorno ho provato a cambiare lo schermo del mio vecchio cellulare. Tuttavia, ho avuto il problema di staccare la batteria. La batteria era infatti incollata in modo particolarmente aggressivo alla scocca. Ho provato a rimuovere la sostanza collante con l'alcol isopropilico, ma non è bastato e ho spaccato la batteria facendo leva. Per la prossima volta, quale sostanza mi consigli di utilizzare per sciogliere la colla che tiene la batteria alla scocca?
audio \ model	large	medium	medium NO CUDA	small	small NO CUDA	ORIGINALE
vestiario	Ok, questo è un audio molto improvvisato. Stavo decidendo cosa mettermi. probabilmente mi metto una maglietta bianca Maglietta strana in realtà, perché ha il tessuto che è simile a un Apollo. Questa è una maglietta. t-shirt molto larga, molto grande, mi piace, probabilmente me la metto, i soliti pantaloncini poi non so se mettermi le scarpe o se uscire con le Birkenstock, perché la comodità insomma non si batte. Probabilmente uscirò con le Birkenstock.	Ok questo è un audio molto improvvisato stavo decidendo cosa mettermi probabilmente mi metterò una maglietta bianca, una maglietta strana perché ha il tessuto che è simile a un Apollo. Poi ha una maglietta, una t-shirt molto leggera. molto grande mi piace ovviamente me la metto soliti pantaloncimi nevi poi non so se metterò le scarpe o se uscirò con le birkenstock perché la comodità insomma non si batte. Probabilmente uscirò con le bieriche stock.	ok questo è un audio molto improvvisato Stavo decidendo cosa mettermi. Probabilmente mi metterò una maglietta bianca e una Maglietta strana, perché ha il tessuto che sembra una polo. Abbiamo una maglietta. molto largo molto grande mi piace ovviamente me la metto soliti pantaloncini neve poi non so se metterò le scarpe o se uscire con le birkenstock perché la comodità insomma non si parte probabilmente uscirò con le birkenstock	ok questo audio molto improvvisato stavo decidendo cosa metterò mi probabilmente mi metterò una maglietta biranca in amore una maglietta strana, perché ha il tessuto che è simile a un apolo e una maglietta molto larga, molto grande, mi piace, probabilmente me la metto, soliti pantalons non so se vedete le scarpe o si uscire con le birche stocche perché la comodità insomma non si bante, probabilmente uscivo a voler via a Kestok	Ok questo audio molto improvvisato Stavo decidendo cosa mettermi. Probabilmente mi metterò una maglietta bianca, maglietta strana, perché ha il tessuto che simile a una polo. Qui è una maglietta. molto larga, molto grande, mi piace, probabilmente me la metto, soliti pantalons cinele non so se vedete le scarpe o si uscire un birch in stocco perché la comodità insomma non si bante. Probabilmente uscio a voler via a Kestok.	Ok, questo è un audio molto improvvisato. Stavo decidendo cosa mettermi. Probabilmente mi metto una maglietta bianca. Una maglietta strana in realtà, perché ha il tessuto che è simile a una polo. Poi ho una maglietta, una t-shirt, molto larga, molto grande, mi piace, probabilmente me la metto, i soliti pantaloncini neri. Poi non so se mettermi le scarpe o se uscire con le Birkenstock, perché la comodità insomma non si batte. Probabilmente uscirò con le Birkenstock.
lettura	Oggi parliamo di Dario Argento. Dario Argento è un regista, sceneggiatore e produttore cinematografico italiano. è molto celebre sia in Italia sia all'estero soprattutto in Francia, Giappone e Stati Uniti. tant'è che il suo soprannome è Maestro del Brivido, avendo dedicato al cinema. horror e thriller la maggior parte della sua produzione. Fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure. concentrandosi sui meccanismi della suspense e del terrore, in particolar modo nei primi film, considerati i capostipiti del giallo all'italiana.	Oggi parliamo di Dario Argento. Dario Argento è un regista, sceneggiatore e produttore cinematografico italiano. è molto celebre sia in Italia sia all'estero, soprattutto in Francia, Giappone. e Stati Uniti. Tant'è che il suo sopranome è Maestro del Privido, avendo dedicato al cinema. horror e thriller la maggior parte della sua produzione. Fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure. concentrandosi sui meccanismi della suspens e del terrore, in particolar modo non nei primi film, considerati i capostipiti del giallo all'italiana.	Grazie Oggi parliamo di Dario Argento. Dario Argento è un regista, sceneggiatore e produttore cinematografico italiano. è molto celebre sia in Italia sia all'estero, soprattutto in Francia, Giappone e Stati Uniti. Tant'è che il suo soffanome è maestro del brivido, avendo dedicato al cinema. horror e thriller la maggior parte della sua produzione. Fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure, concentrandosi sui meccanismi della suspens e del terrore, in particolar modo nei primi film, considerati i capostipiti del giallo all'italiana.	Oggi parliamo di Davio Argetto. Dario Argento è un regista, scegliatore e produttore in cinematografico italiano. è molto celeboe sia in Italia sia all'estero soprattutto in Francia, Giappone e Stati Uniti. Tant'è che il suo sopannome è Maestro del Brivido, avvenno ridicato al cinema. Hover e Thriller la maggior parte della sua produzione. fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure. consentandosi sui meccanismi della suspense e del terrore, in particolar modo, in quei mie film, considerati i capostipiti del giallo all'italiana.	Oggi parliamo di Davio Argetto. Dario Argend, è un regista, finisciatore e produttore cinematologico italiano. è molto celebra, sia in Italia sia all'estero soprattutto in Francia, Giappone estati uniti. Tant'è che il suo soffa nome è Maestro del Brivido, avendo ridicato al cinema. Hover e Triller la maggior parte della sua produzione. fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure. consentandosi sui meccanismi della suspense ed del terrore, in particolar modo, in un nepo e mie film, considerati i capostipiti del giallo all'italiana.	Oggi parliamo di Dario Argento. Dario Argento è un regista, sceneggiatore e produttore cinematografico italiano. E' molto celebre sia in Italia sia all'estero, soprattutto in Francia, Giappone e Stati Uniti. Tant'è che il suo soprannome è Maestro del Brivido, avendo dedicato al cinema horror e thriller la maggior parte della sua produzione. Fin dagli esordi, Argento trasfonde nelle sue opere le proprie paure, concentrandosi sui meccanismi della suspense e del terrore, in particolar modo nei primi film, considerati i capostipiti del giallo all'italiana.

4.5 Test preliminare su piattaforma robotica

È stato effettuato un test preliminare di utilizzo di questa tecnologia su un robot sociale iElio presso la pediatria di Padova ed è stato impiegato un portatile senza CUDA per far girare il codice. Questo test preliminare ha evidenziato diversi pro, tra cui:

- semplicità di installazione del programma
- possibilità di implementare facilmente un LLM come llama per poter elaborare una risposta
- privacy dei dati sensibili, fondamentale lavorando con minori in un contesto ospedaliero

I principali problemi rinvenuti sono:

- tempistiche, confermando come la mancanza della tecnologia CUDA non permetta la trascrizione e un' eventuale risposta dal robot in tempi accettabili
- necessità di hardware esterno, il solo raspberry montato sul robot non garantirebbe il corretto funzionamento del programma

Il test è stato una buona prova sul campo del programma, purtroppo con la potenza di calcolo disponibile l'utilizzo del cloud rimane vantaggioso. Tuttavia avendo a disposizione un robot dotato di CUDA, ad esempio sfruttando un kit Jetson Nano, il programma potrebbe essere facilmente implementato in locale.

Capitolo 5

Conclusioni

Nel contesto di questa tesi è stata presa in esame l'implementazione e l'utilizzo di Whisper per il riconoscimento vocale automatico (ASR) in tempo reale in locale. Di seguito, riassumiamo i principali vantaggi e svantaggi di eseguire Whisper in locale rispetto all'utilizzo di soluzioni cloud. I principali vantaggi di eseguire in locale questo speech-to-text sono:

- privacy dei dati: assicura che i dati audio non vengano trasmessi a server remoti, proteggendo così la privacy degli utenti. Questo è particolarmente importante per applicazioni che gestiscono informazioni sensibili o personali.
- velocità di risposta: utilizzando un hardware adeguato possiamo ottenere trascrizioni rapide senza delay aggiuntivi causati dalla trasmissione dei dati in rete
- indipendenza da una connessione internet

Invece i principali svantaggi sono:

- risorse hardware: l'esecuzione locale di Whisper richiedere hardware specializzato, ad esempio schede grafiche compatibili con CUDA come è emerso dai risultati
- costi di setup iniziali: normalmente maggiori di un'implementazione in cloud
- scalabilità limitata: le soluzioni locali possono incontrare difficoltà nel gestire un numero elevato di richieste simultanee, a differenza delle soluzioni cloud

In sintesi, l'uso di Whisper in locale offre vantaggi significativi in termini di privacy e controllo, ma comporta anche sfide legate alle risorse hardware e alla gestione del sistema. La scelta tra un'implementazione locale e una basata su cloud dipenderà dalle specifiche esigenze di sicurezza, budget e capacità tecniche dell'organizzazione.

Bibliografia

[1] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey e C. Ré, *Training Complex Models with Multi-Task Weak Supervision*, 2018. arXiv: 1810.02840.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey e I. Sutskever, *Robust Speech Recognition via Large-Scale Weak Supervision*, 2022. arXiv: 2212.04356.

[3] D. S. Park, W. Chan, Y. Zhang et al., «SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,» in *Interspeech 2019*, ser. interspeech 2019, ISCA, 2019. doi: 10.21437/interspeech.2019-2680. indirizzo: http://dx.doi.org/10.21437/Interspeech.2019-2680.

[4] G. Huang, Y. Sun, Z. Liu, D. Sedra e K. Weinberger, *Deep Networks with Stochastic Depth*, 2016. arXiv: 1603.09382.

[5] I. Provilkov, D. Emelianenko e E. Voita, *BPE-Dropout: Simple and Effective Subword Regularization*, 2020. arXiv: 1910.13267.