1 Introduzione

I protocolli di turn-taking sono fondamentali in molti aspetti della nostra vita. Questi protocolli definiscono chi sta parlando o compiendo una determinata azione in un determinato istante, considerando che un'altra persona sta aspettando che quell'azione finisca per iniziare il proprio turno. Gli umani sono generalmente molto bravi in questo tipo d'interazione, capendo istantaneamente di chi è il turno e quando sta alla prossima persona, tutto questo in breve tempo e con pochissime sovrapposizioni. D'altra parte invece, conversational agents diversi dagli umani, come per esempio assistenti vocali o robot sociali, hanno problemi a gestire questo tipo d'interazione portando a frequenti incomprensioni, interruzioni e lunghi tempi di risposta. Quello che noi diamo per scontato, è un'attività molto difficile per questo tipo di agenti che hanno bisogno di alcuni indizi per capire quando un turno è finito e quando è il loro turno di parlare. Il principale indizio che i protocolli di turn-taking prendono in considerazione è il silenzio e in particolare il rilevamento dell'attività vocale o voice activity detection (VAD).

Lo scopo di questa tesi è quello di sviluppare un sistema di turn-taking management robusto al rumore, basato su vad usando un modello neurale, da utilizzare all'interno dell'androide Abel. Abel è un androide di nuova generazione iperrealistico che si trova attualmente nel centro di ricerca Enrico Piaggio dell'Università di Pisa. Abel è costituito da una testa e da un torso mossi da servomotori di ultima generazione. L'androide è anche dotato di una videocamera e dei microfoni integrati per emulare le capacità uditive di un essere umano. Uno dei problemi principali che Abel aveva nell' interazione con le persone era capire appunto quando una persona aveva finito il proprio turno. Questo perché la rilevazione della fine del turno era basata sul rilevamento del livello di rumore e non sul rilevamento della voce. In situazioni di rumore di sottofondo quindi Abel non riusciva a distinguere il rumore dalla voce, e questo portava a tempi di risposta molto lunghi, che

rendevano la conversazione complicata. L'idea è stata quindi di aggiornare il modulo di turn-taking di Abel usando un voice activity detector neurale capace di distinguere la voce anche da situazioni di rumore molto intenso. Inoltre per garantire al modello una potenza computazionale slegata dalle capacità hardware presenti in loco con Abel, e per mantenere un' architettura service oriented il sistema di elaborazione dell'audio si trova in remoto rispetto alla posizione del robot.

2 Literature Review

Lo studio della letteratura in questa tesi ha lo scopo di presentare una panoramica della ricerca in ambito di turn-taking nell'interazione uomo-robot e di mostrare quali sono i principali metodi usati per l'attività di voice activity detection.

In un recente articolo vengono presentati alcuni dei concetti fondamentali riguardo al turn taking ed in particolare quali sono gli indizi principali che facilitano l'operazione di coordinamento e quali sono le tecniche principali utilizzate per l'end of turn detection e prediction. Gli indizi maggiormente utilizzati sono:

- Indizi verbali: il completamento di una unità sintattica.
- Metrica: con metrica ci si riferisce agli aspetti non verbali del dialogo come: intonazione e timbro.
- Respiro: il respiro è intrinsecamente legato alla parola perché prima di parlare dobbiamo respirare
- Sguardo: lo sguardo è fondamentale nell'operazione di turn-taking principalmente in una conversazione a più parti, in quanto lo sguardo ha un ruolo importante nella selezione del prossimo interlocutore
- Gesti: in uno studio è stato riscontrato che alcuni gesti sono molto

efficaci come segnali di mantenimento del turno perché chi ascoltava non ha quasi mai provato a prendere il turno durante questi gesti.

I modelli di end of turn detection e prediction sono gli aspetti più studiati del turn-taking e il loro scopo è determinare quando l'utente ha terminato il proprio turno e il sistema può iniziare a parlare. In particolare l'articolo presenta tre tipi di modelli:

- Silence-based: la fine del turno è rilevata utilizzando un vad, ed una soglia di silenzio è usata per capire quando il turno è finito ed il sistema può iniziare a parlare.
- Ipu-based: in questo caso le pause inter-unità sono rilevate usando un vad. Qui viene presa una decisione in base ad altri indizi rilevati
- Modelli continui: la voce dell'utente è analizzata in maniera continua e incrementale in modo predire la fine del turno.

Passiamo ora a vedere quali sono i metodi principali usati nell'attività di voice activity detection. La voice activity detection è una tecnica di analisi del segnale usata per distinguere segmenti di audio dove è presente la voce oppure no. Questa operazione viene fatta in due step: l'estrazione delle features dal segnale, e l'utilizzo di uno schema per effettuare la classificazione. Le principali features che vengono utilizzate sono:

- Energia: è la potenza del segnale.
- Zero-crossing rate: rappresenta il rapporto con cui il segnale attraversa l'asse dello zero. Segnali con un zero-crossing rate alti sono spesso classificati come voce.
- Armonicità: perchè la voce produce un suono ricco armonicamente
- Formant structure: che si basa sull'analisi dei coefficienti spettrali di frequeza di Mel

- Stazionarietà: la voce è tipicamente un segnale non stazionario
- Modulazione: La struttura temporale del parlato è dominata da un caratteristico picco di modulazione energetica a circa 4 Hz.

3 Methods and Methodologies

In questa sezione analizzeremo i principali strumenti utilizzati per lo svolgimenti di questo lavoro che sono: SileroVAD, WebRTC e aiortc

3.1 SileroVAD

Lo strumento SileroVAD è stato presentato per la prima volta in un articolo sul The Gradient "One voice detector to rule them all" nel 2022. SileroVAD è un vad neurale che dimostra buone performance e risultati con le seguenti caratteristiche:

- Licenza permissibile
- Alta qualità
- Altamente portabile
- Supporta audio di 8 e 16 kHz
- Supporta segmenti audio superiori a 30ms
- Trainato su più di 100 lingue
- Ha una bassa latenza

In particolare SileroVAD usa una rete neurale multi-head attention based e come feature da estrarre utilizza la trasformata di fourier a tempo breve.

3.2 WebRTC

WebRTC è un'insieme di APIs e protocolli open source che permettono una comunicazione real time tra browsers. Lo standard Webrtc copre due tecnologie differenti: l'utilizzo dei dispositivi di acquisizione multimediale e la connettività peer-to-peer. La connettività è gestita dall'interfaccia RTCPeerConnection. Questa ha bisogno di un meccanismo di signaling, ovvero di un meccanismo per lo scambio di messaggi di coordinazione e controllo. Per procedere con l'effettivo scambio dei dati però WebRTC necessita anche dello scambio d'informazioni di connettività che permette ai peer di comunicare anche dietro a NAT e Firewall. Questo è possibile grazie all'ICE framework. L'idea alla base di questo framework è quella di trovare il percorso migliore per connettere i peer.

3.3 aiortc

aiortc è una libreria che permette di utilizzare WebRTC in Python. Abbiamo deciso di utilizzare questa libreria perchè l'implementazione di WebRTC è integrata nei browsers e non esistono dei bindings per Python. Inoltre l'API JavaScript utilizza una MediaStack interna che rende difficile l'elaborazione dell'audio o del video contenuti nello streaming. Invece la libreria aiortc è abbastanza facile da utilizzare è permette di accedere allo streamig audio in maniera abbastanza diretta. Questo anche grazie ai Media Helpers forniti dalla libreria che non fanno parte di WebRTC ma che facilitano la manipolazione dei media stream.

4 Implementation

In questa sezione andrò a presentare quali sono i dettagli implementativi di questo progetto.

4.1 Software Architecture

L'architettura software di questo sistema è dunque composta da un JavaScript client che si connette ad un Python aiohttp server usando la tecnologia WebRTC per trasferire dati audio real time. Il server Python utilizza la libreria aiortc per ricevere questi dati. Il server contiene inoltre un modulo Analyzer che si occupa della parte di gestione del turn-taking effettuando l'operazione di voice activity detection usando SileroVAD con il framework PyTorch. Inoltre il server comunica lo stato corrente della conversazione all'esterno grazie alla tecnologia websockets e il tutto è inserito all'interno di un docker container per garantire i vantaggi offerti dalla dockerizzazione.

4.2 JavaScript Client

Partendo dal client, come già specificato, il suo scopo è di trasmettere l'audio acquisito dal microfono di Abel al server tramite WebRTC. Nell' interfaccia è possibile selezionare l'origine dell'audio e tramite il bottone di Start si fa partire la connessione con il server e la trasmissione dati. Sotto possiamo visualizzare lo stato della connessione, e una serie di metriche di connessione tra cui il round trip time, jitter e pacchetti parsi. Con lo start, il client, va a creare l'oggetto RTCPeerConnection, acquisire i device selezionati per aggiungerli allo streaming e procede alla fase di signaling utilizzando uno scambio di richieste e risposte HTTP in cui viene create l'offerta di connessione, si aspetta che il processo di ICE gathering state sia completato, e si invia l'offerta di connessione al server tramite richiesta HTTP Post.

4.3 Python server

Il Python server ha lo scopo di ricevere l'audio in real time e di processarlo passandolo alla classe Analyzer che andrà a effettuare l'attività di voice activity detection. Il server è stato realizzato usando la libreria aiohttp e utilizza la libreria aiortc per ricevere e processare l'audio in arrivo dal client. La traccia

audio ricevuta dal server viene intercettata dalla classe AudioTrackProcessing che deriva dalla classe MediaStreamTrack che in WebRTC rappresenta una singola media track all'interno di uno stream di dati. Per intercettare e processare l'audio in ingresso abbiamo sovrascritto il metodo asincrono recv() che viene chiamato automaticamente dalla libreria quando una determinata traccia viene utilizzata. Sovrascrivendo il metodo recv() abbiamo avuto accesso ai singoli frame della traccia che vengono poi passati alla analyze() function della classe Analyzer dove vengono effettivamente processati.

4.4 VAD Analyzer

Il frame per essere utilizzato da SileroVAD ha prima bisogno di essere preprocessato. Questo perchè l'audio frame in ingresso dalla connessione ha una serie di caratteristiche che dipendono dal device di acquisizione come per esempio il sampling rate e il numero di canali. Inoltre vogliamo controllare la lunghezza dei frame in ingresso perchè il modello può processare solamente frame di lunghezza superiore a 30ms. Per fare questo abbiamo effettuato un resampling dei frame in ingresso e abbiamo utilizzato un buffer per gestire i samples dei frame in ingresso in modo tale da ottenere il frame della lunghezza desiderata. Una volta ottenuti i frame nel formato desiderato gli abbiamo dovuti normalizzare e create un tensore in modo tale da essere utilizzati dal modello in PyTorch. Adesso l'idea è quella di procedere all'inferenza del modello sul frame in ingresso e ottenere un valore che rappresenta la probabilità che l'audio chunk contenga voce. Questo valore viene poi utilizzato per gestire lo stato della conversazione.

Vediamo ora come viene gestito lo stato della conversazione. Il sistema utilizza un modello silence-based quindi cerca di capire se il turno della persona è finito analizzando se il frame audio contiene voce oppure no. In particolare per determinare la fine del turno solitamente viene utilizzato una valore soglia, per cui se l'audio in arrivo contiene silenzio per un determinato peri-

odo di tempo viene rilevata la fine del turno. Questo valore soglia è un valore critico per le performance del sistema perchè noi vogliamo che sia corto in quanto voglia ottenere una buona reattività del sistema ma non può essere troppo corto in quanto un valore corto può confondere una pausa nella frase con un cambio di turno. Un valore lungo invece ci assicura che il turno è effettivamente finito ma rallenta la reattività del sistema. Per risolvere parzialmente le criticità di questo valore abbiamo deciso di utilizzare due valori soglia, uno per rilevare un cambio di turno potenziale e uno per un cambio di turno definitivo. In questo modo quando si rileva silenzio pari al primo valore si rileva un cambio di turno potenziale. Qui si inizierà già l'elaborazione della frase dell'utente e prima di rispondere si aspetta che anche il secondo valore venga superato. Se invece tra il primo e il secondo valore sogli si rileva della voce si ritorna nello stato iniziale perchè vuol dire che il silenzio era causato da una pausa. In questo modo abbiamo la certezza della fine del turno da un valore soglia più lungo ma anche una reattività migliore perchè abbiamo iniziato l'elaborazione in anticipo.

5 Experiments and results

In questa sezione andrò a presentare gli esperimenti e i risultati ottenuti in termini di performance del sistema. Questo è stato testato su un dataset da noi creato il Libri-Demand dataset e abbiamo confrontato le performance del sistema utilizzando come vad sia SileroVad che il vad di Webrtc.

5.1 Performance metrics

Per testare accuratamente le performance del sistema avevamo bisogno di un dataset composto da diverse parti parlate con diversi tipi di rumori di sottofondo e diversi livelli di rumore. Visto che non abbiamo trovato nessun dataset che rispecchiava le nostre necessità abbiamo deciso di creare noi un dataset che abbiamo chiamato Libri-Demand. Questo dataset è ottenuto

dalla fusione di due dataset LibriSpeech e il DEMAND dataset. LibriSpeech è un dataset composto da più di 1000 ore di parlato derivato da audio books. Il DEMAND dataset invece è un dataset contenente 6 categorie di rumori. Le parti clean contengono 1 ora di clean speech derivato da LibriSpeech combinate insieme con aggiunta di silenzio artificiale. Per ottenere i noisy datasets abbiamo unito ai clean datasets i rumori del DEMAND dataset ottenendo un totale di 5 ore di noisy speech. Per normalizzare il livello del rumore rispetto al volume della voce abbiamo applicato il rumore con un SNR cioè il signal to noise ratio di 0db, ovvero sessioni con lo stesso livello di voce e rumore. Infine per testare le performance del sistema con diversi livelli di rumore abbiamo costruito l'ultima parte combinando clean speech con le categorie di rumore, ognuna con 5 livelli di SNR per un totale di 7 ore di noisy speech.

In questo grafico troviamo un riassunto delle metriche di performance quali: accuracy, precision, recall e f1 score per entrambi i modelli di vad Silero e WebRTC e per i due dataset male noisy e female noisy, quelli con i rumori di sottofondo a 0db. Dal grafico notiamo come il dataset con voci femminili ottiene valori leggermente più alti rispetto alle voci maschili, questo probabilmente perchè l'estrazione delle feature vocali da una voce femminile risulta leggermente più semplice. Otteniamo valori buoni con un f1 score di 0.91 e 0.93 utilizzando SileroVAD mentre otteniamo 0.89 in entrambi i dataset utilizzando WebRTC.

Per comparare correttamente i due modelli abbiamo ottenuto il grafico della ROC curve per entrambi i dataset e da questi grafici possiamo notare come il sistema che usa SileroVAD ottiene dei risultati nettamente migliori rispetto al sistema con WebRTC.

Analizzando i risultati del dataset noisy sessions vediamo le differenze di entrambi i modelli con diversi livelli di rumore. E infine un grafico che mostra le accuracy della categoria "Total" per entrambi i modelli. Dai test risulta

quindi che il sistema mostra buone performance considerando le diverse categorie di rumore e gli scenari molto rumorosi presentati.

6 Conclusions and future works

In questa tesi abbiamo presentato lo sviluppo di un turn-taking management system robusto per conversational agents e inizialmente pensato per essere utilizzato sull'Androide Abel presente in Università. Inizialmente abbiamo presentato un revisione della letteratura corrente riguardo i temi di turn-taking e voice activity detection. Dopodiché abbiamo presentato le tecnologie che abbiamo utilizzato nello sviluppo della nostra soluzione e la sua implementazione. Infine abbiamo testato le performance del sistema sul Libri-Demand dataset da noi creato, confrontandone le performance utilizzando come vad sia Silero che il vad di Webrtc. Una futura espansione di questo sistema può essere quella d'includere un modulo di speaker diarization, capace di rilevare diverse voci di diversi interlocutori e capace di tenere traccia delle stato di molteplici conversazioni contemporaneamente.