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. I conversational agents, 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. 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.

2 Methods and Methodologies

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

2.1 SileroVAD

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.

2.2 WebRTC

WebRTC è un'insieme di APIs e protocolli open source che permettono una comunicazione real time tra browsers. 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.

2.3 aiortc

aiortc è una libreria che permette di utilizzare WebRTC in Python. Abbiamo deciso di utilizzare questa libreria perchè 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.

3 Implementation

3.1 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 We-

bRTC 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.

3.2 VAD Analyzer

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 periodo 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.

4 Experiments and results

4.1 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. Per ottenere i noisy datasets abbiamo unito ai clean datasets i rumori del DEMAND dataset ottenendo un totale di 5 ore di noisy speech normalizzato con un SNR pari a 0db. 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 in 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.

5 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. 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.