Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Analisi vocale per il riconoscimento dell'identità del parlatore

Laureando

Matteo Bressan

Relatore

Prof. Carlo Ferrari

Correlatore

Prof. Federico Avanzini

20 Aprile 2015

Anno Accademico 2014/2015

Ai miei genitori

Essere giovani vuol dire tenere aperto l'oblò della speranza, anche quando il mare è cattivo e il cielo si è stancato di essere azzurro.

Bob Dylan

Indice

Sommario

In questo lavoro è stato realizzato un sistema di riconoscimento del parlatore indipendente dal testo, adatto sia a scopi di verifica dell'identità che di identificazione del parlatore. Il sistema è stato addestrato utilizzando dei dati di esempio e genera i modelli per i singoli parlatori basandosi su Modelli a Misture Gaussiane. Come feature si è scelto di utilizzare i *Mel Frequency Cepstral Coefficients (MFCC)*. I test sono stati effettuati utilizzando audio di scarsa qualità, in modo da testare il sistema in condizioni non ideali, ottenendo un *Equal Error Rate (EER)* del 4% nel caso di voci maschili e del 6.24% nella controparte femminile. Tuttavia, l'utilizzo di file particolarmente rumorosi senza l'applicazione di nessun tipo di filtro o pre-elaborazione ha messo in serie difficoltà il corretto riconoscimento, rivelandosi il principale problema nell'impiego di questo tipo di tecnologia.

Introduzione

L'utilizzo della voce come parametro biometrico è stato oggetto di molti studi negli ultimi anni, ed è tutt'ora motivo di interesse. La volontà di eseguire le operazioni che facciamo tutti i giorni in maniera più immediata, la necessità di rendere possibili molte azioni a persone che non hanno la possibilità di interagire in maniera tradizionale, e l'aumento di interesse verso nuove e più affidabili misure di sicurezza rendono questo problema di grande interesse. Per questi motivi, le tecnologie in grado di processare ed analizzare in tempo reale la nostra voce sono probabilmente destinate ad entrare nella nostra vita di tutti i giorni.
In questa tesi è stato realizzato un sistema di riconoscimento del parlatore indipendente dal testo, in grado di svolgere entrambi i compiti di verifica ed identificazione.
Il Capitolo 1 introduce il problema del riconoscimento del parlatore. Vengono inoltre descritte le principali tecniche utilizzate per la risoluzione di questo problema, ponendo particolare attenzione alle tecniche adottate per la realizzazione del sistema oggetto della tesi. Infine vengono introdotte le metriche di valutazione utilizzate per la misura delle prestazioni di questi sistemi.
Il Capitolo 2 presenta il tipico schema di funzionamento di un sistema di riconoscimento del parlatore, descrivendo nel dettaglio tutte le fasi che compongono il processo di riconoscimento.
Nel Capitolo 3 viene inizialmente descritta ed analizzata la libreria Alize, utilizzata per l'implementazione del software. Vengono inoltre esposte le scelte progettuali e presentato l'approccio adottato per la realizzazione del sistema.
Il Capitolo 4 presenta infine le modalità di test e i risultati ottenuti.

Capitolo 1

Il problema del riconoscimento del parlatore

Gli esseri umani possono essere identificati sfruttando varie caratteristiche che li rendono unici, conosciute come misure biometriche. Alcune di queste sono puramente fisiologiche e non possono essere alterate dagli individui, come ad esempio l'impronta digitale, l'iride o la struttura del DNA. Altre misure sono invece una combinazione di fattori fisiologici e comportamentali, tra le quali troviamo la voce. Le misure puramente fisiologiche sono generalmente considerate più affidabili perché possono variare molto poco per un individuo ma presentano differenze significative tra individui differenti.
La voce presenta sì delle differenze ragionevoli tra persone differenti, ma allo stesso tempo può variare sensibilmente anche per il singolo individuo: basta pensare all'invecchiamento, ma anche ad un semplice raffreddore. Questa è una delle principali problematiche dovute all'utilizzo della voce come parametro biometrico.
L'utilizzo di questa misura presenta però anche dei vantaggi, che la rendono un argomento interessante ed oggetto di studio. Il principale di questi sta nel fatto di essere poco invasivo e disponibile senza il minimo sforzo da parte dell'utente. Parlare è infatti considerata una cosa normale, e generalmente non viene vista come una "misura". La voce può in effetti essere catturata anche se ci troviamo di fronte ad una persona non particolarmente collaborativa, o addirittura senza che ce se ne accorga. Inoltre la tecnologia necessaria è alla portata di tutti, basta infatti un semplice microfono, che al giorno d'oggi è presente anche in oggetti di uso comune come lo smartphone. Bisogna però considerare che la qualità dell'audio è un fattore rilevante ai fini del riconoscimento, quindi l'utilizzo di microfoni di qualità scadente, una cattiva conversione analogico-digitale o un elevato rumore di sottofondo possono avere un elevato impatto negativo sulle prestazioni del sistema.

Un sistema di riconoscimento del parlatore ha quindi lo scopo di capire *chi* sta parlando, e non va confuso, come spesso accade, con un sistema di riconoscimento del parlato che è finalizzato invece a capire *cosa* viene detto. Un sistema di questo tipo riceverà quindi in input una o più registrazioni della voce di una persona sconosciuta, e restituirà in output la presunta identità del parlatore.
Ci sono alcune distinzioni che possono essere fatte per il problema del riconoscimento del parlatore. Possiamo innanzitutto individuare due grandi classi, che cercano di rispondere a due diversi interrogativi:

- **Identificazione del parlatore** (*Speaker Identification*): il sistema ha lo scopo di capire chi sta parlando nella registrazione ricevuta in input. Verrà quindi fatto un confronto con i parlatori conosciuti e restituito quello che con maggiori probabilità è quello corretto.

- **Verifica del parlatore** (*Speaker Verification*): in questo caso l'input sarà accompagnato dall'identità dal presunto parlatore (successivamente indicato anche come *target*), e il sistema dovrà verificare se questa identità è corretta o meno.

Queste due classi possono essere ulteriormente caratterizzate. Una scelta importante deriva dalla tipologia delle frasi ricevute in input, e il sistema può essere:

- **Indipendente dal testo**, cioè senza restrizioni sul dizionario utilizzato dal parlatore. Non essendo quindi a conoscenza del possibile contenuto delle frasi il compito diventa più difficile, ma il sistema risulta più semplice da usare per l'utente.

- **Dipendente dal testo**, in cui la difficoltà dipende da grandezza e complessità del dizionario. Alcuni utilizzi, come l'autenticazione, possono limitarsi a richiedere delle sequenze di numeri o frasi precise, mentre in altri casi sono necessari dei dizionari più ampi con una conseguente difficoltà che si avvicina a quella del caso indipendente dal testo.

Un'ultima distinzione che è possibile fare, ma solo per i sistemi di *speaker identification*, è la seguente:

- **Closed-set**: è il caso più semplice, e indica che il sistema è certo che l'audio ricevuto in input sia relativo ad un parlatore conosciuto.

- **Open-set**: in questo caso il sistema deve essere pronto a trovarsi in una situazione in cui deve rispondere "parlatore sconosciuto". Il problema è più difficile in quanto il sistema deve valutare il caso in cui

nessun parlatore conosciuto ha una probabilità abbastanza alta di essere quello giusto, e non può dare come risposta semplicemente quello con la probabilità più alta. C'è quindi bisogno di stabilire una soglia, che spesso non è facile da trovare.

In questa tesi è stato realizzato un sistema di riconoscimento del parlatore indipendente dal testo, in grado di svolgere entrambe le funzioni di identificazione e verifica. Per il caso di identificazione del parlatore è stata adottata una soluzione closed-set.

1.1 Stato dell'arte

Il problema del riconoscimento del parlatore è attualmente affrontato utilizzando tecniche di *machine learning* (apprendimento automatico). Come è noto, queste tecniche prevedono due fasi esecutive: una fase di *training* (allenamento) in cui il sistema "impara" a prendere delle decisioni sulla base di alcuni dati di esempio, e una fase di *testing* in cui il sistema si trova a dover prendere delle decisioni a partire da nuovi dati a lui sconosciuti.
Nel nostro caso l'input è dato da un file audio contente la registrazione della voce di una persona. Durante la fase di training uno o più file sono dati in input al sistema, in modo che questo possa generare un modello che rappresenti il più fedelmente possibile le caratteristiche vocali di quell'individuo. Per fare questo è necessaria una fase di estrazione di *feature*. Queste ultime rappresentano quelle particolarità che caratterizzano la voce di una persona e verranno descritte più dettagliatamente in seguito. Una volta estratte le feature è possibile effettuare il training del sistema per generare un modello che permetta di riconoscere un determinato parlatore. In Figura 1.1 si può vedere uno schema che rappresenta la fase di training.

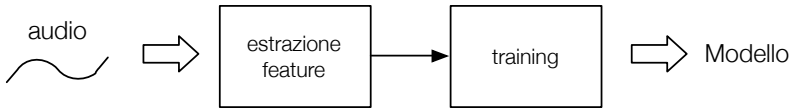

Figura 1.1: Schema di funzionamento della fase di training.

Per la fase di testing invece l'input è rappresentato solamente da un file audio, per il quale vengono estratte le feature per essere poi confrontate con i vari modelli conosciuti in una fase che viene detta *matching*. Durante questa fase vengono prodotti degli *score* (Sezione 1.1.3) che rappresentano la probabilità che un determinato modello sia quello corretto (cioè che il parlatore associato

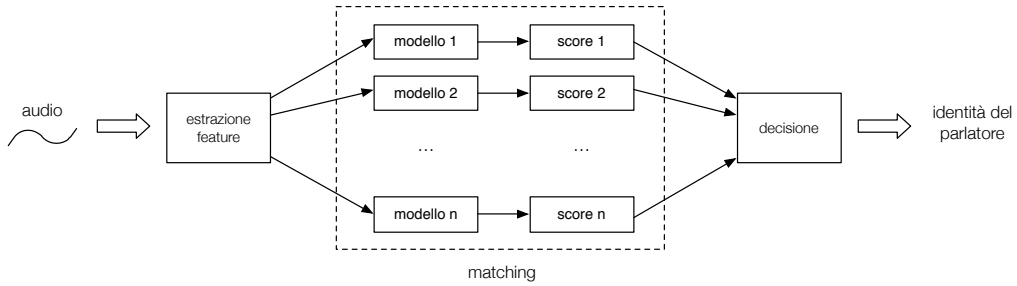

Figura 1.2: Schema di funzionamento della fase di test.

a quel modello sia effettivamente quello che parla nella registrazione ricevuta in input). Questi score sono poi utilizzati per prendere la decisione finale: nel caso di *Identificazione* viene solitamente fornito in output il modello che ha ottenuto lo score più alto, mentre per la *Verifica* la decisione è più delicata perché si basa su una soglia, che può essere determinata durante la fase di training. In Figura 1.2 possiamo vedere uno schema rappresentante i passi appena descritti. Entriamo ora un po' più nel dettaglio sul modo di affrontare le fasi di estrazione delle feature e di training.
Abbiamo detto che le feature sono dei dati che rappresentano le caratteristiche vocali di una persona. Queste particolarità ci permettono, nella vita di tutti i giorni, di riconoscere senza troppi sforzi una persona che conosciamo sentendola semplicemente parlare, e includono informazioni come il timbro, l'intensità e molte altre, che il nostro cervello elabora senza che noi ce ne rendiamo conto. Tutte queste informazioni devono però essere codificate in modo da poter essere interpretate da una macchina, e questo si può fare analizzando il segnale audio registrato tramite il microfono.
Le feature possono essere classificate nei due seguenti modi: feature di **basso livello** e feature di **alto livello**. Le prime sono le più semplici da ottenere perché legate direttamente al segnale fisico, come le caratteristiche dello spettro. Le seconde invece sono quelle che ci danno informazioni di tipo linguistico, come il dialetto o il modo di esprimersi (idioletto), ma sono spesso difficili da ottenere. Tra le feature di basso e alto livello ce ne sono altre che negli anni si sono rivelate le più efficaci, rappresentando un ottimo compromesso. Le due più conosciute sono i *Mel-Frequency Cepstral Coefficients (MFCC)* e i *Linear Prediction Coefficients (LPC)*. Gli MFCC sono attualmente i più utilizzati, perché riescono ad ottenere le migliori prestazioni nella maggior parte dei casi, e sono quindi stati scelti per la realizzazione del sistema oggetto di questa tesi. Questi verranno analizzati più nel dettaglio nella Sezione 1.1.1.

Per quanto riguarda invece la fase di training e generazione dei modelli, abbiamo ancora una volta due principali approcci:

- **Modelli di Markov Nascosti (HMM)**: è l'approccio più antico e modella il singolo parlatore con una catena di Markov. E' basato sull'analisi dei fonemi ed è quindi più utilizzato nei sistemi dipendenti dal testo.

- **Modelli a Misture Gaussiane (GMM)**: ogni parlatore è modellato da una somma di distribuzioni Gaussiane. Questo approccio ottiene i migliori risultati nei sistemi indipendenti dal testo.

Dal momento che il sistema realizzato è indipendente dal testo, è stato scelto di utilizzare un approccio basato su GMM. I Modelli a Misture Gaussiane sono presentati nella Sezione 1.1.2. Altri approcci sono stati testati e sono attualmente oggetto di ricerca, come le reti neurali e le *Support Vector Machines*.

1.1.1 Mel-Frequency Cepstral Coefficients

Le varie feature che possono essere estratte dalla voce di una persona risiedono solitamente nello spettro del segnale audio. Lo spettro può essere rappresentato in vari modi, ma probabilmente il metodo più utilizzato si basa proprio sui *Mel-Frequency Cepstral Coefficients (MFCC)*. Questi sono stati sviluppati inizialmente per fare riconoscimento del parlato, con l'obiettivo di essere il più possibile indipendenti dalla persona, ma si sono rivelati efficaci anche per il problema del riconoscimento del parlatore e sono tutt'ora considerate delle feature ad alte prestazioni.
Gli MFCC sono ottenuti mediante un particolare *filterbank*, ossia una serie di filtri passa-banda posti in sequenza lungo l'asse delle frequenze. Questo filterbank è stato concepito basandosi su alcuni concetti di psicoacustica, in modo da simulare le proprietà del sistema uditivo umano: le bande dei filtri sono più strette per le basse frequenze e più ampie alle alte frequenze. Questo si traduce in una maggior risoluzione per le medie e basse frequenze. Inoltre, le frequenze centrali di ogni filtro non sono linearmente distribuite, ma seguono una funzione logaritmica: la *scala Mel*, anch'essa scelta per le sue proprietà psico-acustiche.
In psicoacustica la conversione da Hertz a Mel è la seguente:

$$f_{Mel} = 2595 \log_{10} \left(1 + \frac{f_{Hz}}{700}\right)$$

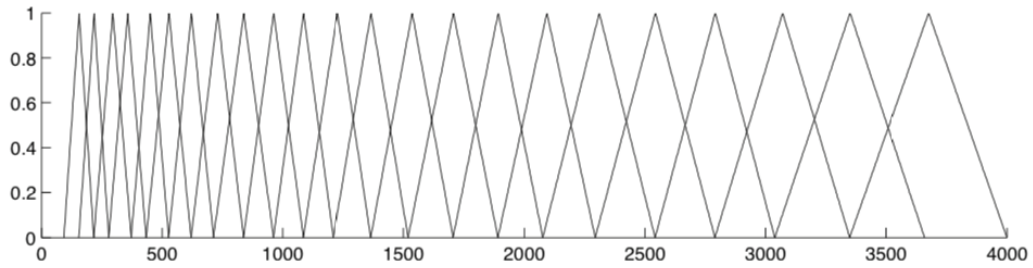

Figura 1.3: Filterbank utilizzato per il calcolo degli MFCC.

e viceversa

$$f_{Hz} = 700\left(10^{\frac{f_{Mel}}{2595}} - 1\right)$$

Il filterbank, che si può vedere in Figura 1.3, lavora nel range di frequenze $[64, 4000]$ Hz. Il numero di filtri passa-banda è 24 e sono solitamente progettati con una risposta triangolare, sovrapponendosi parzialmente in frequenza. Vediamo ora come vengono calcolati gli MFCC:

- Viene inizialmente diviso il segnale in *frame* di 20-30ms, sovrapposti per 10ms. Questo viene fatto perchè possiamo assumere che l'audio non vari in un intervallo così piccolo, e che il segnale sia quindi in una condizione stazionaria per la breve durata del frame.

- Ogni frame viene moltiplicato per una finestra di *Hamming* e viene poi applicata la trasformata veloce di Fourier (FFT) per ricavarne lo spettro. Viene poi considerato solamente il modulo della trasformata.

- Vengono calcolati i logaritmi naturali dei coefficienti così ottenuti. In caso di coefficienti con valori molto piccoli è possibile incontrare difficoltà nel calcolo dei logaritmi, ed è stata quindi introdotta una funzione che fissa il valore minimo per un coefficiente a $1.93 \cdot 10^{-22}$. Al termine di questo passo avremo quindi che il valore minimo di un coefficiente è $\log(1.93 \cdot 10^{-22}) = -50$.

- Infine viene applicata la trasformata coseno, il cui scopo è mantenere solamente le informazioni rilevanti e ridurre la dimensionalità dei dati. Normalmente vengono mantenuti 12 o 19 coefficienti.

I passi appena descritti sono rappresentati schematicamente in Figura 1.4.
Gli MFCC sono spesso forniti insieme ai loro *Delta* e *Delta-Delta*. Questo permette di ottenere informazioni riguardo all'evoluzione temporale che non

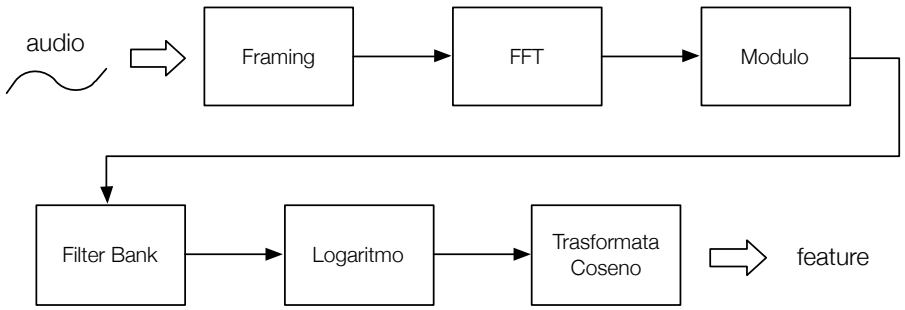

Figura 1.4: Schema di calcolo per gli MFCC.

vengono rappresentate dai semplici MFCC, spesso detti coefficienti statici proprio per questo motivo. Si è visto che aggiungendo queste informazioni, il sistema riesce a generare del modelli più accurati, migliorando di conseguenza le prestazioni generali. I Delta possono essere calcolati nel seguente modo:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

dove d_t rappresenta il coefficiente delta per il frame t, calcolato sulla base dei coefficienti statici c_{t+n} e c_{t-n}. Tipicamente N assume il valore 2.
I coefficienti Delta-Delta (Accelerazione) sono calcolati allo stesso modo, ma partendo dai Delta invece che dai coefficienti statici.

1.1.2 Modelli a misture gaussiane

L'utilizzo dei modelli a misture Gaussiane si è affermato per la risoluzione del problema del riconoscimento del parlatore per la loro semplicità di implementazione e perché poco costosi computazionalmente. Inoltre si è visto che questa tecnica porta ad ottenere degli ottimi risultati, in particolare su sistemi indipendenti dal testo.
Possiamo assumere che i vettori di feature $\mathbf{X}$ siano generati da una funzione di densità di probabilità associata ad una mistura di distribuzioni Gaussiane, data da:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i p_i(\mathbf{x})$$

dove w_i rappresenta il peso della distribuzione i, e $0 \leq w_i \leq 1$ per $1 \leq i \leq M$, $\sum_{i=1}^{M} w_i = 1$. Assumendo che le distribuzioni risiedano in uno spazio

D-dimensionale, ogni distribuzione ha la forma:

$$p_i(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \times \exp\left[-\frac{1}{2}(x_i - \mu_i)\Sigma_i^{-1}(x_i - \mu_i)\right]$$

dove μ_i è il vettore $D \times 1$ della media della distribuzione, Σ_i è la sua $D \times D$ matrice di covarianza e $\mathbf{x}$ è il punto/vettore nel quale la distribuzione è valutata.
Le distribuzioni Gaussiane hanno delle matrici di covarianza Σ_i piene, ma in genere queste si assumono diagonali in modo da ridurle ad un vettore $D \times 1$. In questo modo quando è richiesto il calcolo della matrice di covarianza inversa, durante la fase di training, questa può essere calcolata in modo molto efficiente.
I vettori delle medie μ_i e delle covarianze Σ_i vengono quindi raggruppati in due matrici μ e Σ e il modello generato dalla mistura viene indicato dalla tripla $\lambda = \{w, \mu, \Sigma\}$.
Dato $\mathbf{X}$ in input, la generazione del modello avviene con l'utilizzo dell'algoritmo *Expectation-Maximization (EM)*. L'obiettivo di questo algoritmo è quello di modificare iterativamente i parametri di una GMM con lo scopo di adattare un modello ad un dato insieme di feature. Alla fine della fase di training la GMM dovrebbe avere una distribuzione di probabilità il più simile possibile a quella, sconosciuta, che rappresenta l'insieme di feature in input. Per inizializzare i parametri del modello può essere utilizzato l'algoritmo *K-Means*.

La probabilità a posteriori che il t-esimo vettore di feature $\mathbf{x_t}$ sia generato dalla componente $p_m(\mathbf{x_t})$ può essere scritta come:

$$P(m|\mathbf{x_t}, \lambda) = \frac{w_m p_m(\mathbf{x_t})}{p(\mathbf{x_t}|\lambda)}$$

I parametri del modello λ vengono stimati sulla base di $P(m|\mathbf{x_t}, \lambda)$ nel seguente modo:

$$\mu_\mathbf{m} = \frac{\sum_{t=1}^{T} P(m|\mathbf{x_t}, \lambda)\mathbf{x_t}}{\sum_{t=1}^{T} P(m|\mathbf{x_t}, \lambda)}$$

$$\mathbf{\Sigma_m} = \frac{\sum_{t=1}^{T} P(m|\mathbf{x_t}, \lambda)\mathbf{x_t}\mathbf{x_t^T}}{\sum_{t=1}^{T} P(m|\mathbf{x_t}, \lambda)} - \mu_\mathbf{m}\mu_\mathbf{m}^\mathbf{T}$$

$$w_m = \frac{1}{T}\sum_{t=1}^{T} P(m|\mathbf{x_t}, \lambda)$$

I due passaggi dell'algoritmo EM consistono nel calcolare $P(m|\mathbf{x_t}, \lambda)$ dato il modello corrente, e aggiornare il modello utilizzando le formule appena

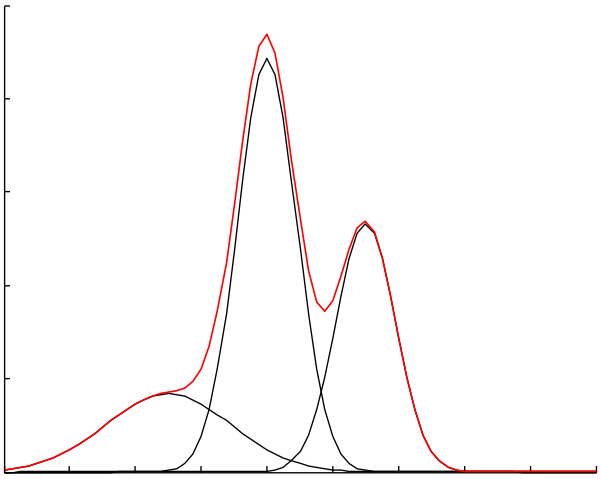

Figura 1.5: Esempio di una GMM. In nero sono rappresentate le singole distribuzioni e in rosso il modello della mistura Gaussiana.

esposte. Questi due passaggi vengono ripetuti finché non viene raggiunto un criterio di convergenza, o viene effettuato un determinato numero di iterazioni. In Figura 1.5 si può vedere un esempio di mistura Gaussiana.

1.1.3 Scoring

Vediamo ora come vengono calcolati gli score e cosa rappresentano. Data una porzione di parlato Y e un ipotetico parlatore S, il compito è di determinare se Y è stato detto da S. Consideriamo le seguenti due ipotesi:

- H_0: Y è stato detto proprio dal parlatore S;
- H_1: Y non è stato detto da S.

Sia ora $p(Y|H_i)$ la funzione di densità di probabilità di H_i valutata per la porzione di parlato Y. Definiamo il **rapporto di verosimiglianza**:

$$\frac{p(Y|H_0)}{p(Y|H_1)}$$

Nel nostro caso specifico H_0 è rappresentata da un modello, che possiamo chiamare λ_{hyp}, mentre Y può essere visto come un insieme di feature $X = \{x_i\}_{i=1}^{N}$. Analogamente, l'ipotesi H_1 è rappresentata dal modello $\lambda_{\overline{hyp}}$. Il rapporto di verosimiglianza può essere quindi riscritto come:

$$\frac{p(X|\lambda_{hyp})}{p(X|\lambda_{\overline{hyp}})}$$

Solitamente però vengono utilizzati i logaritmi delle probabilità per una maggior maneggevolezza e stabilità dei valori. Definiamo quindi il **rapporto di log-verosimiglianza**:

$$\Lambda = \log p(X|\lambda_{hyp}) - \log p(X|\lambda_{\overline{hyp}})$$

Più è alto questo valore, più è probabile che l'insieme di feature X appartengano al presunto parlatore S.
Molto spesso questo valore viene normalizzato per essere reso il più possibile indipendente dal parlatore. La formula generale per la normalizzazione è la seguente:

$$\Lambda^{norm}(X) = \frac{\Lambda(X) - \mu(X,S)}{\sigma(X,S)}$$

dove i valori μ e σ dipendono dalle caratteristiche delle feature X e dal parlatore S, e sono solitamente calcolate utilizzando un database differente (o un suo sottoinsieme) rispetto a quello utilizzato per valutare le prestazioni del sistema. Tra le normalizzazioni più utilizzate ci sono *H-norm*, *T-norm* e *Z-norm*. Alcune di queste, utilizzate anche nel sistema realizzato, sono descritte più nel dettaglio nella Sezione 2.3.1.

1.2 Valutazione delle prestazioni

Le prestazioni di un sistema di riconoscimento del parlatore possono essere valutate misurando quanto sono differenti le medie degli score dei parlatori sinceri e degli impostori. Possiamo definire questi due insiemi nel seguente modo: gli score relativi ai parlatori sinceri sono quelli derivanti da un test nel quale l'input Y appartiene al parlatore S, mentre quelli relativi agli impostori sono risultanti da test in cui l'input Y non appartiene al parlatore S.
I diagrammi di *Detection Error Trade-off (DET)*, descritti in dettaglio nella prossima sezione, sono stati sviluppati con questo scopo e sono ormai diventati lo standard per la valutazione delle prestazioni di sistemi di verifica del parlatore. Inoltre in questi diagrammi è possibile determinare un punto di lavoro che rappresenta il compromesso desiderato tra diversi tipi di errori. In un sistema di riconoscimento del parlatore possiamo infatti definire due possibili errori:

- **Miss** o **Falso Negativo**: si verifica quando l'input Y appartiene proprio al presunto parlatore S, ma il sistema sostiene il contrario;
- **Falso Positivo**: si verifica quando l'input Y non appartiene al parlatore S, ma il sistema lo associa ad esso.

Questi due errori possono avere differenti impatti, in relazione alla situazione in cui il sistema viene utilizzato. Per esempio in un sistema utilizzato per lo sblocco di una porta, qualche falso negativo può essere accettato, mentre i falsi positivi rappresentano casi ben più gravi. Un altro esempio può essere la ricerca a scopo investigativo, all'interno di una serie di conversazioni, di porzioni di audio in cui parla una determinata persona. In questo caso un falso negativo può significare la perdita di informazioni fondamentali agli scopi dell'indagine, e quindi non può essere tollerato.
Questi due tipi di errore però devono essere rappresentati in modo semplice e significativo per poterne trarre delle considerazioni sulle prestazioni di un sistema e a questo scopo sono state sviluppate varie tipologie di grafici. Inoltre sono state inoltre introdotte nel tempo delle metriche globali diventate ormai degli standard, come l'*Equal Error Rate (EER)* e il *Detection Cost Function (DCF)*.
Tutte queste metriche di valutazione si applicano bene al caso di verifica del parlatore. Nel caso si voglia invece valutare un sistema di riconoscimento del parlatore adibito ad eseguire un compito di identificazione, vengono normalmente utilizzati due indici basati sul *rank* ottenuto dal parlatore corretto nella risposta del sistema. Se consideriamo la lista di parlatori restituita in risposta ad una registrazione audio ricevuta in input, ordinata in ordine decrescente di score, possiamo definire il rank di un parlatore come la posizione che questo assume nella lista ordinata.
Il primo indice viene detto *Mean Rank* ed è banalmente definito come la media del rank assunto dal parlatore corretto. Il secondo indice è detto *Mean Reciprocal Rank (MRR)*, ed è definito nel seguente modo:

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{rank_i}$$

dove n è il numero di test eseguiti, mentre $rank_i$ è il rank del parlatore corretto nella risposta relativa al test i.
L'indice di Mean Rank fornisce un'idea della posizione che assume in media il parlatore corretto nella lista ordinata. Un valore di questo indice pari a 1 indica che il sistema non sbaglia mai e corrisponde ad una precisione del 100%. Di conseguenza, quanto più il suo valore è vicino ad 1, tanto più le prestazioni del sistema sono elevate.
L'MRR, a differenza di Mean Rank, è un indice normalizzato e assume valori compresi tra 0 e 1, con il caso ideale di un sistema che non sbaglia mai rappresentato dal valore uno. Questo indice può essere visto come una misura della precisione del sistema, che però tiene in considerazione la posizione della risposta corretta nella lista ordinata: se il sistema risponde in modo sbagliato

ma la risposta corretta risiede comunque nelle prime posizioni sta ad indicare che gli score relativi ai due casi sono molto simili, e quindi che il parlatore (sbagliato) fornito in risposta ha probabilmente una voce con caratteristiche simili a quello corretto, indicando quindi che il sistema non ha dato una risposta totalmente insensata.

1.2.1 Curve DET

I primi grafici che vennero utilizzati per rappresentare le prestazioni di un sistema di riconoscimento del parlatore furono le *curve ROC (Receiver Operating Characteristic)*. In questi grafici l'asse delle ascisse rappresenta il numero di falsi positivi, mentre sull'asse delle ordinate si trova il numero di identificazioni corrette. Tuttavia, questi grafici hanno il grande svantaggio di non essere di facile lettura, rendendo difficile il confronto tra diversi sistemi. Nel corso degli anni si è invece stabilita la convenzione di utilizzare un differente grafico per la rappresentazione delle prestazioni di questi sistemi, chiamato *Detection Error Trade-off (DET)*. In una curva DET l'asse delle ascisse rappresenta la probabilità di avere un falso positivo, mentre l'asse delle ordinate rappresenta la probabilità di miss.
Per ricavare i dati necessari a tracciare una curva DET bisogna aver completato la fase di testing del sistema. Questa è eseguita seguendo uno schema preciso: una porzione di audio Y è testata rispetto ad un modello λ e il relativo score $\Lambda(Y, \lambda)$ viene calcolato e memorizzato, ripetendo questa operazione per diversi input e modelli.
Terminata questa fase, essendo a conoscenza del fatto che un determinato input Y appartenga o meno al parlatore modellato da λ, è possibile ottenere i due insiemi di score relativi ai parlatori sinceri e agli impostori, rispettivamente Λ_t e Λ_i. A questo punto, le probabilità di avere un falso positivo o un miss possono essere calcolate nel seguente modo:

$$P_{fa} = \frac{\#(\text{impostore identificato come parlatore sincero})}{|\Lambda_i|}$$
$$P_{miss} = \frac{\#(\text{parlatore sincero identificato come impostore})}{|\Lambda_t|}$$

Come si può notare, queste due probabilità dipendono dalle condizioni nelle quali è stato fatto il test, e in particolare dalla soglia θ utilizzata che stabilisce la linea di confine tra le due ipotesi H_0 e H_1.
La curva DET indica le prestazioni del sistema per ogni possibile soglia θ: ogni punto rappresenta infatti le probabilità di miss e di falso positivo per un determinato valore della soglia. Inoltre, prima di disegnare la curva,

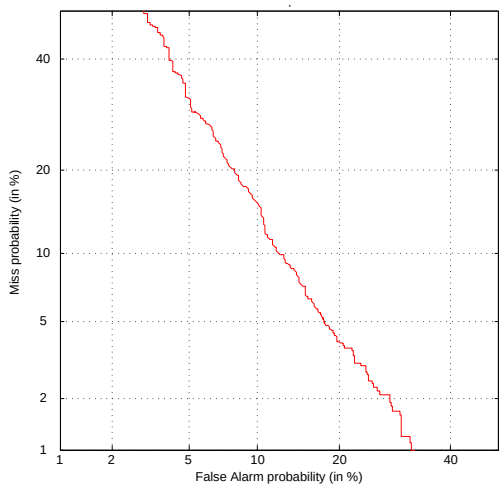

Figura 1.6: Un esempio di curva DET.

le probabilità P_{miss} e P_{fa} vengono mappate alla loro deviazione standard tramite una funzione di ripartizione. Questo è equivalente ad assumere che i valori degli score siano generati da due distribuzioni Normali, con il risultato che il grafico risulta molto più leggibile per via della stessa scala degli assi orizzontale e verticale. In Figura 1.6 si può vedere un esempio di curva DET: aumentando la differenza tra le medie degli score Λ_t e Λ_i la curva tenderà ad avvicinarsi all'origine, mentre si allontanerà se le medie sono più vicine.
Nella curva DET è possibile determinare il punto nel quale le due probabilità P_{miss} e P_{fa} assumono lo stesso valore. Questo punto è detto Equal Error Rate (EER) ed è la più semplice ed utilizzata metrica di misura delle prestazioni di un sistema di verifica del parlatore. Esso corrisponde alla scelta ottima di θ, dando la stessa importanza ai due tipi di errori.
Una metrica più raffinata è rappresentata invece dall'indice di Detection Cost Function (DCF), definito come:

$$DCF = C_{miss}P_{miss}P_{H_0} + C_{fa}P_{fa}P_{H_1}$$

dove P_{miss} e P_{fa} sono definite come sopra, P_{H_0} e P_{H_1} sono le probabilità *a priori* di effettuare un test rispetto al parlatore sincero o un impostore, C_{miss} e C_{fa} rappresentano il costo dei due errori.
A differenza delle due probabilità, gli altri valori presenti nella formula per il calcolo del DCF sono dipendenti dal contesto in cui il sistema verrà utilizzato e vanno tarati di conseguenza (ad esempio, se il contesto richiede di minimizzare i falsi positivi, è possibile assegnare un alto valore a C_{fa} per trovare il punto di lavoro, e la relativa soglia θ, che ci garantisce le prestazioni richieste). Il DCF è in definitiva meno generale dell'Equal Error Rate ma è

più significativo nel caso l'obiettivo sia testare il sistema per uno specifico utilizzo.

Capitolo 2

Modello del sistema

Abbiamo visto nel precedente capitolo quali sono le tecniche principalmente utilizzate per affrontare il problema del riconoscimento del parlatore. In questo capitolo verrà analizzato il modello tipico di un sistema di questo tipo, e descritte nel dettaglio tutte le fasi che compongono il processo di riconoscimento.

2.1 Estrazione delle feature

La fase di estrazione delle feature prevede di estrarre dalla registrazione audio fornita in input le informazioni significative che consentono di identificare le caratteristiche rilevanti della voce del parlatore.
Si è scelto di utilizzare gli MFCC come feature, e in particolare di mantenere 19 coefficienti con l'applicazione della trasformata coseno. La scelta di utilizzare 19 coefficienti è stata presa, dopo aver fatto alcuni test, per le migliori prestazioni garantite rispetto al classico utilizzo di 12 coefficienti. Maggiori informazioni su questi test possono essere trovare nella Sezione 4.4.
Sono state inoltre adottate alcune strategie per migliorare ulteriormente le performance delle feature:

- Energy detection
- Normalizzazione

L'algoritmo di Energy Detection consente di analizzare i singoli frame ed individuare quelli che presentano un'energia del segnale molto bassa: questi infatti corrispondono a momenti di silenzio e non sono quindi rilevanti ai fini della successiva generazione dei modelli. Vengono quindi contrassegnati i frame che contengono del parlato e quelli che invece non contengono informazioni utili. In questo modo i frame etichettati come irrilevanti possono

essere ignorati in tutte le fasi successive, guadagnando sia nell'accuratezza dei modelli generati che nella velocità di esecuzione.
La normalizzazione delle feature ha invece lo scopo di migliorare le prestazioni della successiva fase di training nel caso in cui si abbiano file audio provenienti da diverse sorgenti. Gli effetti del rumore ambientale di sottofondo possono influire in maniera piuttosto negativa sulle prestazioni di un sistema per il riconoscimento del parlatore, come pure differenti caratteristiche di microfono, canale di trasmissione dell'audio e via dicendo. Per ridurre questi effetti le feature vengono spesso normalizzate con un semplice algoritmo che calcola media μ e varianza σ^2 dell'intero insieme dei vettori di feature. Ogni vettore viene poi normalizzato utilizzando la seguente formula:

$$\mathbf{x}' = \frac{\mathbf{x} - \mu}{\sigma}$$

2.2 Training dei modelli

La teoria sostiene che sia possibile generare il modello di un parlatore inizializzandone i parametri con dei valori casuali per poi allenarlo e renderlo un modello concreto. Tuttavia si è visto che questo metodo porta a problemi di *under-training*. Se tutti i modelli vengono generati indipendentemente, infatti, alcune distribuzioni della mistura probabilmente non vengono adattate durante la fase di training, mantenendo i loro valori di inizializzazione. Questo fatto può portare a situazioni in cui un nuovo vettore di feature che non è stato visto dal sistema durante la fase di training possa essere classificato secondo una distribuzione casuale. Se il livello di under-training è molto alto si rischia di avere quindi un sistema che risponde in modo casuale.
Per evitare questo problema viene spesso utilizzato un *modello di background*. Questo particolare modello viene generato utilizzando l'algoritmo EM, normalmente con 5-15 iterazioni, partendo da un gran numero di feature estratte da diversi parlatori, con lo scopo di rappresentare la voce in generale. Il modello di background viene poi utilizzato come punto di partenza per generare i modelli dei singoli parlatori. Adattando i modelli a partire da uno più generale si riesce ad evitare il problema, in quanto il punto di partenza non è più un insieme di distribuzioni casuali, ma una mistura che modella già la voce umana. Chiaramente, se il modello di background risulta avere un elevato livello di under-training, il problema può comunque verificarsi anche con questo approccio, ma un buon numero di iterazioni dell'algoritmo EM durante la generazione del modello riduce notevolmente la probabilità che questo accada.

2.2.1 Modello di background

Il modello di background è difficile da definire, in quanto dovrebbe essere in grado di modellare l'intero spazio delle alternative rispetto all'ipotetico parlatore ricevuto in input. Ci sono vari metodi per ottenere un modello di background, ma i due più utilizzati sono:

- definire il modello come un grande numero di singoli modelli di parlatori;
- generare il modello come una singola GMM a partire da feature provenienti da diversi parlatori.

Il primo approccio identifica un gruppo di parlatori (conosciuto in letteratura come ***Cohort***) e definisce la probabilità di essere un impostore in funzione della probabilità di essere uno di questi parlatori. Il metodo funziona discretamente ma la scelta di parametri come i parlatori da utilizzare e il loro numero non è banale, inoltre diventa molto costoso computazionalmente all'aumentare della dimensione del *cohort*. Questo approccio dà i risultati migliori nel caso in cui i parlatori siano scelti specificatamente per ogni target.
Il secondo approccio utilizza un modello completamente nuovo (conosciuto come ***Universal Background Model (UBM)*** o ***World Model***) generato prendendo come input delle registrazioni audio provenienti da un gran numero di parlatori differenti, diversi dal target. Utilizzare un tal numero di feature in ingresso implica tempi molto lunghi per la fase di training, ma si ha il vantaggio di un modello unico utilizzabile per qualsiasi target. Inoltre è possibile generare più modelli di background da utilizzare in base al contesto, per esempio uno per le voci maschili e uno per quelle femminili. L'approccio world model è attualmente il più utilizzato ed è generalmente chiamato *GMM-UBM*. Si è scelto di utilizzare proprio questo approccio per la realizzazione del sistema oggetto della tesi.

2.2.2 Target

La strategia utilizzata per generare i modelli relativi ai singoli target viene detta *Model Adaptation*, dal momento che il modello del parlatore è ottenuto adattando il modello di background generato precedentemente. Il criterio utilizzato è detto *Maximum A Posteriori (MAP)* e prevede di generare il modello in modo da massimizzare la probabilità a posteriori che il presunto parlatore S sia quello corretto, data in ingresso la registrazione Y. Si nota che questo è il criterio opposto a *Maximun Likelihood (ML)*, utilizzato per la

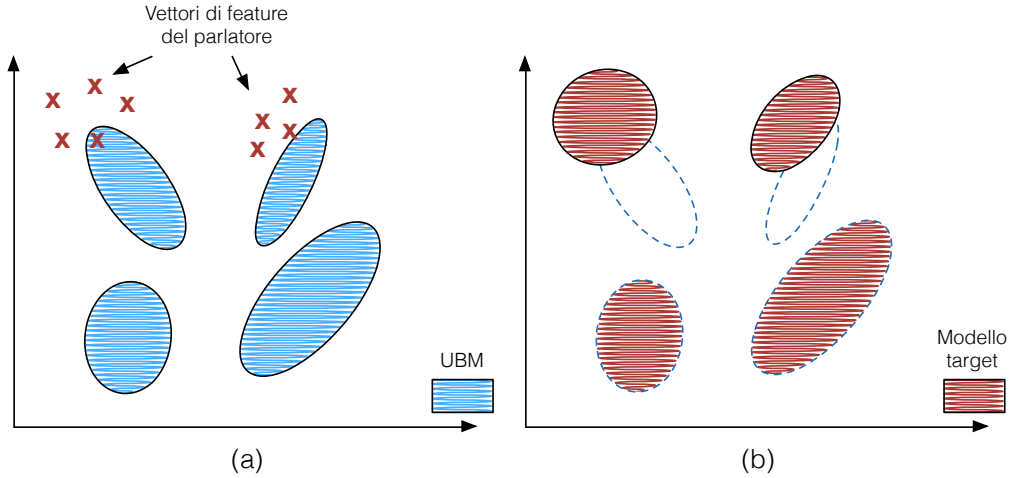

Figura 2.1: Rappresentazione grafica delle due fasi necessarie per adattare un modello target. (a) I vettori di feature sono mappati nelle misture del modello UBM precedentemente calcolato. (b) I parametri delle misture adattate vengono determinati sulla base dei nuovi dati e del precedente modello.

fase di training del modello di background, che prevede di massimizzare la probabilità che l'input Y sia generato dal modello relativo al parlatore S.
Come per l'algoritmo EM, anche MAP si svolge in due fasi. La prima è identica alla fase di "Expectation" dell'algoritmo EM, nella quale vengono mappati probabilisticamente i vettori di feature in ingresso nelle misture relative al modello di background precedentemente determinato (Figura 2.1(a)). La seconda fase invece prevede di adattare le "vecchie" misture sulla base dei nuovi dati. In questa fase è presente un parametro α, detto *fattore di rilevanza*, che indica quanti nuovi dati devono essere osservati in una mistura affinché i parametri di quest'ultima vengano adattati (Figura 2.1(b)).
E' interessante far presente che adattare tutti i parametri della mistura Gaussiana (w_i, μ_i, Σ_i) non è sempre la scelta migliore. Dopo aver generato il world model molti sistemi adattano solamente la media μ per i modelli target. E' stato infatti dimostrato che generalmente adattare anche gli altri due parametri non porta a significativi incrementi sulle prestazioni finali del sistema. Questo metodo funziona bene grazie alla sua robustezza rispetto a feature che non sono state viste nei dati utilizzati per la fase training. Una GMM indipendente restituisce infatti uno score molto basso nel caso di feature nuove, mentre il modello di background riesce probabilmente a modellare anche questi dati: questo si traduce in una riduzione del rapporto di verosimiglianza. Un modello adattato invece è stato ottenuto a partire dal world model e

quindi nel calcolo delle probabilità entrambi i modelli di background e target subiscono gli effetti di nuovi dati non incontrati nella fase di training, senza più ridurre lo score come avveniva nel caso di modelli indipendenti.

2.3 Testing

La fase di testing prevede di ricevere in input dei vettori di feature estratte da una o più registrazioni audio e il corrispondente modello target. Vengono quindi calcolati i rapporti di log-verosimiglianza per ogni coppia (feature relative ad una registrazione audio, target), che costituiscono poi la risposta del sistema. Nel caso del sistema realizzato, che utilizza un approccio GMM-UBM, il modello $\lambda_{\overline{hyp}}$ rappresenta il modello di background. Possiamo quindi riscrivere la formula per il calcolo del rapporto di log-verosimiglianza vista nella Sezione 1.1.3 nel seguente modo:

$$\Lambda = \log p(X|\lambda_{hyp}) - \log p(X|\lambda_{ubm})$$

Gli score vengono in seguito normalizzati in modo da renderli il più possibile indipendenti dal parlatore.

2.3.1 Normalizzazione degli score

Negli ultimi anni sono state proposte varie tecniche per normalizzare gli score risultanti dalla fase di testing. Tra le più importanti ci sono le normalizzazioni H, T e Z.
La **normalizzazione H** *(Handset Normalization, H-norm)* viene utilizzata nei casi in cui le registrazioni utilizzate per i test siano state effettuate con apparecchi differenti, ad esempio telefoni cellulari e fissi. Questa particolare normalizzazione può portare in questi casi a dei grandi guadagni nelle performance generali del sistema.
Nella **normalizzazione T** *(Test Normalization, T-norm)* vengono effettuati degli ulteriori test che servono a misurare gli score ottenuti da una determinata registrazione ricevuta in input rispetto ad una serie di modelli di impostori.
La **normalizzazione Z** *(Zero Normalization, Z-norm)* è simile alla normalizzazione T ma prevede di effettuare dei test aggiuntivi che misurano gli score ottenuti utilizzando diverse registrazioni di impostori testate rispetto ad uno specifico modello target.
Nel caso di queste ultime due normalizzazioni, l'insieme di score ottenuti dai test aggiuntivi viene utilizzato per normalizzare media e varianza degli score ottenuti dai test originali. In alcuni casi, inoltre, possono essere utilizzate

delle combinazioni di due diverse normalizzazioni in modo da cercare di ottenere il meglio da entrambe. Spesso come combinazioni sono utilizzate le normalizzazioni TZ e ZT.

Capitolo 3

Implementazione del sistema

Il software è stato realizzato in linguaggio `C` e testato su sistema operativo Ubuntu 12.04. Per la realizzazione sono state utilizzate le seguenti librerie e strumenti:

- `Alize 3.0`: descritta in dettaglio nella prossima sezione, è stata utilizzata per la realizzazione di tutte le fasi del sistema di riconoscimento,
- `PortAudio v19`: realizzata in `C`, fornisce delle API per la riproduzione e registrazione di suoni. E' stata utilizzata nella fase di *preprocessing* e per la registrazione della voce tramite il microfono del pc,
- `Libsndfile 1.0.25`: libreria scritta in `C` per la lettura e scrittura di file audio. Utilizzata in combinazione con la libreria PortAudio per il salvataggio dei file contenenti le registrazioni,
- `SPro Tools 5.0`: finalizzato all'analisi di segnali audio, mette a disposizione degli strumenti che implementano i più famosi algoritmi per l'estrazione delle feature della voce. Utilizzato nella fase di estrazione delle feature, elaborate poi dalla libreria Alize.

In questo capitolo viene analizzata nel dettaglio la libreria Alize e discusso l'approccio adottato per l'implementazione del sistema di riconoscimento.

3.1 La libreria Alize

La libreria *Alize*, ben conosciuta nel settore, è una piattaforma open-source per l'autenticazione biometrica che si pone l'obiettivo di permettere lo sviluppo di applicazioni biometriche mettendo a disposizione degli strumenti ad alto e basso livello. La libreria, realizzata in `C++` e arrivata alla sua versione 3.0, presenta due moduli:

- la libreria vera e propria: un framework statistico generico, focalizzato su strutture dati ed algoritmi utilizzati nel campo della ricerca sulla voce e sul linguaggio;

- gli strumenti *LIA_RAL*: una serie di funzioni, basate sulla libreria Alize, specifiche per il problema del riconoscimento del parlatore.

Entrambi i moduli utilizzano *GNU Autotools* per l'indipendenza dalla piattaforma: sono infatti utilizzabili sui principali sistemi Windows, Linux e MacOSX.
Per la realizzazione del sistema si è scelto di non implementare da zero tutte le funzioni necessarie, ma di affidarsi agli strumenti offerti da LIA_RAL. Questo modulo propone una serie di *tool* che implementano tutte le fasi necessarie per il funzionamento di un sistema di riconoscimento del parlatore. Tuttavia non è presente uno scheletro per la realizzazione di un sistema funzionante, ed è necessario affidarsi ad una documentazione non aggiornata per l'ultima versione degli strumenti e in parte disponibile solamente in lingua francese. Sono comunque presenti alcuni esempi di file di configurazione e di utilizzo.
La fase di estrazione delle feature non è implementata direttamente nella libreria, ma è prevista la lettura di file contenenti i vettori di feature precalcolati. I formati supportati sono quelli generati dai due software *SPro* e *HTK*.

3.1.1 File di configurazione

Tutti gli strumenti messi a disposizione dal modulo LIA_RAL si interfacciano con l'utente tramite l'utilizzo dei seguenti file di configurazione:

- *Configuration file* (`.cfg`): contengono tutti i parametri di configurazione per gli algoritmi implementati dal tool (ad esempio il numero di iterazioni da eseguire nella fase di training per la generazione del modello). E' presente un file di configurazione per ogni tool offerto dal modulo LIA_RAL;

- *Index file* (`.ndx`): questi file sono utilizzati nelle fasi di training e test. Nella fase di training indicano i file audio associati ad ogni modello, mentre nella fase di test rappresentano la lista di test da effettuare;

- *List file* (`.lst`): sono semplicemente delle liste di file, utilizzate per la generazione del modello di background e per l'estrazione delle feature.

```
****************************************
*      TrainWorld specific options     *
****************************************
baggedFrameProbability  0.3
baggedFrameInitProbability  0.7
normalizeModel  true
baggedMinimalLength  3
baggedMaximalLength  10
inputFeatureFilename  ../lst/UBM.lst
```

Figura 3.1: Parte di un file `.cfg`

I file `.cfg` contengono alcune impostazioni comuni tra i vari tool, come il livello di verbosità, ma sono per la maggior parte specifici per lo strumento associato. La loro struttura prevede una suddivisione in sezioni, per ognuna delle quali sono presenti una serie di coppie parametro-valore. Un esempio è riportato in Figura 3.1.
La maggior parte di questi valori devono essere settati a mano andando a modificare direttamente il file, mentre alcune opzioni sono modificabili direttamente passando dei parametri da riga di comando. I parametri che non possono essere impostati da riga di comando sono generalmente quelli che non vengono modificati spesso, come il numero di iterazioni e altri parametri specifici degli algoritmi implementati. Generalmente infatti, questi parametri vengono impostati durante la fase di settaggio del sistema, in modo da trovare i valori che permettono di raggiungere le prestazioni migliori per la specifica situazione in cui andrà utilizzato, dopodiché non vengono più modificati in quanto potrebbero andare a influire sensibilmente sul comportamento dell'intero sistema. I valori che rappresentano invece l'input per gli algoritmi implementati, che solitamente varia ad ogni esecuzione, prevedono anche di essere impostati da riga di comando in modo da essere modificati agevolmente. Il parametro `inputFeatureFilename` presente nell'esempio di Figura 3.1, ad esempio, può essere impostato lanciando il programma nel seguente modo:

```
TrainWorld --inputFeatureFilename ../lst/UBM.lst
```

Ci sono tuttavia alcuni parametri che necessitano di essere modificati tra un'esecuzione e l'altra, che però prevedono solo la modifica tramite il file di configurazione. Per risolvere il problema, ed evitare di modificare continuamente i file, il sistema realizzato prevede di ricevere in input tutto il necessario all'esecuzione per poi inserire automaticamente le impostazioni

nei relativi file cfg senza l'intervento manuale.

3.1.2 Componenti principali

Vediamo ora quali sono i tool utilizzati per la realizzazione del sistema e le loro principali funzioni. Si nota che il modulo LIA_RAL mette a disposizione altri strumenti oltre a quelli descritti in questa sezione, per i quali si rimanda alla documentazione di Alize in caso di interesse.
Seguendo un ordine che cerca di rispecchiare il più possibile quello di esecuzione, i tool utilizzati sono i seguenti:

- `EnergyDetector`: analizza l'energia del segnale (nel caso di vettori di feature generati con SPro l'energia è rappresentata dal primo elemento del vettore) ed effettua una distinzione tra frame contenenti parlato (alta energia) e frame corrispondenti a silenzio (bassa energia).
  - *Input*: un file `.prm` contenente i vettori di feature estratti da una singola registrazione o un file `.lst` contenente una lista di file;
  - *Output*: un file `.lbl` per ogni file di feature, contenente la classificazione parlato/silenzio per ogni frame.
- `NormFeat`: normalizza le feature per ridurre gli effetti negativi dovuti a differenze di canale di trasmissione, qualità delle registrazioni e così via.
  - *Input*: le feature da normalizzare, con le stesse modalità di `Energy Detector`;
  - *Output*: un nuovo file `.prm` per ogni file di feature, contenente i vettori normalizzati.
- `TrainWorld`: provvede alla generazione del modello di background. Inizialmente tutte le distribuzioni vengono impostate utilizzando media e varianza globali e tutti i pesi sono uguali. Successivamente il modello viene generato iterativamente utilizzando l'algoritmo EM con criterio ML. I parametri più importanti per questo tool sono:
  1. il numero di distribuzioni,
  2. il numero di iterazioni,
  3. la quantità di feature da utilizzare (basata sulla probabilità di utilizzare l'i-esimo vettore di feature).

I tempi di esecuzione sono molto variabili in base ai parametri utilizzati e possono diventare molto lunghi nei casi in cui il numero di distribuzioni e di iterazioni sia particolarmente alto.

  - *Input*: un file `.lst` contenente la lista di feature da utilizzare per la generazione del modello;
  - *Output*: un file `.gmm` contenente la singola GMM che rappresenta il modello di background.

- `TrainTarget`: genera uno o più modelli target. Questi vengono adattati a partire dal modello di background utilizzando un algoritmo EM modificato che sfrutta il criterio MAP. I parametri più importanti sono:

  1. il numero di iterazioni,
  2. il fattore di rilevanza α, relativo all'algoritmo MAP,
  3. la quantità di feature da utilizzare.

  I tempi di esecuzione dipendono dal numero di iterazioni e dal numero di modelli che si intende generare in una singola esecuzione. Impiega comunque meno tempo rispetto alla generazione dell'UBM.

  - *Input*: un file `.ndx` contenente i modelli e le relative feature, formattato nel seguente modo:

    ```
    modello1 featureFile1
    modello2 featureFile1 featureFile2 ..
    ..
    ```

    dove la prima riga rappresenta l'identità del parlatore, e le successive i nomi dei file contenenti le feature da utilizzare per la generazione del modello.
  - *Output*: uno o più file `.gmm` contenenti le GMM che rappresentano i singoli modelli target.

- `ComputeTest`: effettua i test e genera gli score calcolando il rapporto di log-verosimiglianza. Uno dei pochissimi parametri presenti per questo tool rappresenta il numero di distribuzioni da utilizzare per il calcolo della log-verosimiglianza. E' possibile infatti utilizzare solamente le top-k, in modo da rendere i test indipendenti dal numero di distribuzioni utilizzate per la generazione dei modelli ed accorciare sensibilmente i tempi di esecuzione.

  - *Input*: il file `.gmm` relativo al modello di background, un file `.ndx` che contiene le feature associate a ciascun modello coinvolto nei test (lo stesso utilizzato come input per il tool `TrainTarget`) e un altro file `.ndx` contenente i test da effettuare, strutturato nel seguente modo:

    ```
    featureFile1 modello1
    featureFile2 modello2 modello1 ..
    ..
    ```

    dove la prima colonna indica il nome del file contenente le feature estratte dalla registrazione audio utilizzata come test, mentre le successive indicano i presunti parlatori verso i quali calcolare gli score.

  - *Output*: un file `.res` in cui ogni riga indica il risultato di uno specifico test, nel seguente modo:

    ```
    s modello d testFile score
    ```

    dove s rappresenta il sesso (M o F) e d la decisione presa (il tool restituisce semplicemente 1 se lo score è positivo, 0 altrimenti).

- `ComputeNorm`: normalizza gli score in modo da renderli il più possibile indipendenti dal parlatore. Come spiegato nella sezione 2.3.1, nel caso si decida di normalizzare gli score è necessario effettuare dei test aggiuntivi utilizzando un insieme di impostori.

  - *Input*: il file `.res` relativo al test del quale si vogliono normalizzare gli score, il tipo di normalizzazione (T, Z, ..) e il file `.res` con i risultati dei test aggiuntivi;
  - *Output*: un unico file `.res` contenente i test originali con gli score normalizzati.

3.2 Approccio utilizzato

L'implementazione del sistema si è sviluppata in due parti, utilizzando due diversi approcci. La prima implementazione prevede l'utilizzo di un microfono per la registrazione dell'audio utilizzato, sia per generare i modelli dei parlatori che per effettuare i test. Questa versione ha permesso di prendere confidenza con il problema e verificare il funzionamento del sistema in una situazione reale, ma non di trarre delle conclusioni sulle effettive prestazioni in quanto il numero di persone coinvolte era limitato. Infatti per testare il funzionamento di questa prima implementazione si è fatto affidamento a

parenti e amici per la raccolta delle voci da utilizzare nei test, in quanto un database dedicato è stato disponibile solo in un secondo momento.
Con lo scopo di effettuare un'analisi approfondita sulle prestazioni del sistema è stata realizzata quindi una seconda versione, con un funzionamento leggermente differente, che prevede l'utilizzo di un database di file audio progettato appositamente per questo genere di test.
Il software realizzato è composto da due elementi principali:

- una libreria che sfrutta i tool offerti da Alize LIA_RAL
- il sistema vero e proprio, basato sulla libreria realizzata

Questo ha permesso di ottenere una libreria pronta all'uso per la realizzazione di un sistema di riconoscimento del parlatore, che permette sia l'utilizzo tramite registrazione diretta dal microfono del computer, che l'utilizzo classico tramite file offerto dalla libreria Alize. I vari metodi implementati permettono infatti di specificare tramite l'utilizzo di alcuni *flag* se l'input proviene da un file o deve essere registrato al momento.
Il sistema vero e proprio funziona da terminale, e consente di specificare al lancio da riga di comando quale operazione eseguire (generare un modello di background, eseguire dei test, ...) indicando anche il percorso dei file da utilizzare come input e i parametri necessari, in modo da non dover modificare il codice sorgente ogni volta che si vuole cambiare qualche impostazione riguardante le modalità di esecuzione.
Vedremo ora una descrizione dettagliata e separata dei due metodi di funzionamento del sistema.

3.2.1 Funzionamento tramite registrazione diretta

Questo approccio cerca di simulare l'utilizzo in una situazione reale, come potrebbe banalmente essere lo sblocco di una porta. Siamo quindi in un caso di verifica del parlatore, che prevede un singolo test e una risposta binaria.
La registrazione dell'audio avviene tramite il microfono del computer ed è stata realizzata utilizzando la libreria PortAudio. Per la registrazione, la libreria prevede la chiamata ad una *callback* alla quale vengono passati vari parametri come il numero di canali su cui registrare, la frequenza di campionamento, la durata della registrazione e infine una struttura dati appositamente progettata. Questa struttura dati comprende un array nel quale vengono salvati i dati letti attraverso il microfono, la posizione dell'array dalla quale iniziare a inserire i nuovi dati e il numero totale di *frame* richiesti.
Il numero di frame si può ricavare a partire dai parametri dati in input alla

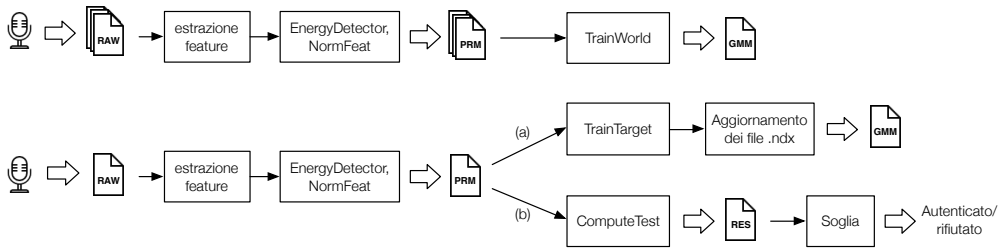

Figura 3.2: Schema di funzionamento tramite registrazione diretta dal microfono del computer.

callback: una registrazione di s secondi a n canali e con una frequenza di campionamento f sarà formata in totale da un numero di frame pari a

$$\#frame = n \cdot s \cdot f$$

Dopo la lettura dei dati provenienti dal microfono è necessario salvarli su file. Per estrarre le feature è stato infatti utilizzato il software SPro, che non prevede un'estrazione "al volo" ma solamente tramite file. Sono stati utilizzati inoltre i due tool `EnergyDetector` e `NormFeat` offerti dalla libreria Alize. La scrittura del file è stata implementata utilizzando la libreria libsndfile. Come per la registrazione, anche in questo caso è necessario specificare le caratteristiche dell'audio (numero di canali, frequenza di campionamento, ...), oltre al formato desiderato. Per quanto riguarda quest'ultimo, il software SPro legge solamente file `RAW` e `WAV`, quindi nel nostro caso la scelta è limitata ad uno di questi due formati.
Il sistema realizzato prevede di salvare file RAW PCM 16bit mono con una frequenza di campionamento a 8 Khz. La bassa frequenza di campionamento è stata scelta per simulare le condizioni delle linee telefoniche, che presentano una scarsa qualità audio. Questo permette di avere un elevato livello di sfida per il sistema, e di testarne quindi le prestazioni in una situazione ben lontana dalle condizioni ideali.
Vediamo ora come vengono generati i modelli. Il modello di background richiede banalmente che vengano registrate le voci di un certo numero di persone, per poi utilizzare nella generazione del modello le feature estratte dai file audio slavati. Questo, una volta salvati i file, avviene tramite l'utilizzo della funzione `TrainWorld`, in modo analogo al caso di funzionamento tramite file descritto nella prossima sezione.
La generazione dei modelli target richiede invece in questo caso dei piccoli accorgimenti. Il fatto che questa versione cerchi di simulare una situazione di utilizzo reale deve prevedere infatti che alcuni modelli siano aggiunti in

seguito alla messa in funzione del sistema, ad esempio per garantire l'accesso ad una nuova persona. Come per l'UBM, una volta ottenuta la registrazione della voce della persona della quale si vuole generare il modello, la procedura è analoga a quella dell'approccio descritto nella Sezione 3.2.2, cioè tramite l'utilizzo della funzione `TrainTarget`. La differenza risiede nel fatto che una volta ottenuto il modello bisogna informare il sistema della sua esistenza. E' necessario infatti modificare due index file: il primo tiene traccia di tutti i modelli e dei file di feature associati, mentre il secondo indica i test da effettuare, o più precisamente i modelli verso i quali verrà effettuato il test. Entrambi questi due index file sono poi utilizzati da `ComputeTest`.
Veniamo infine alla fase di testing. L'autenticazione avviene tramite la registrazione della voce della persona, il salvataggio del file e il lancio di `ComputeTest` che calcola il rapporto di log-verosimiglianza ottenuto utilizzando il modello del presunto parlatore fornito in input insieme alla registrazione. Ottenuto lo score, una soglia (determinata sperimentalmente) decide se questo è abbastanza alto da confermare l'identità del parlatore.
Una rappresentazione grafica dello schema di funzionamento è mostrata in Figura 3.2.

3.2.2 Funzionamento tramite file

Per testare le effettive prestazioni del sistema è stato necessario apportare delle modifiche rispetto alla prima versione che utilizzava il microfono come periferica di input. La prima e ovvia modifica è stata l'eliminazione della fase di registrazione diretta della voce in favore dell'utilizzo di un database progettato appositamente per i test. Il database fornisce dei file audio precedentemente registrati e suddivisi per parlatore (per una descrizione dettagliata si rimanda alla Sezione 4.1).
La seconda differenza riguarda il modo in cui vengono gestite le fasi di testing e di generazione dei modelli target. Infatti, se prima era necessario prevedere la possibilità di aggiungere dei modelli in qualsiasi momento, con l'utilizzo di un database questo non è più necessario: tutti i modelli utilizzabili sono disponibili fin da subito e l'unica cosa da fare è decidere quali andranno utilizzati per la generazione del modello di background e quali come target. Sarà quindi necessario preparare subito tutti i file di configurazione, per poi lanciare le varie fasi in sequenza ed ottenere i risultati da analizzare.
Il database utilizzato è composto da 630 persone, per ognuna delle quali sono presenti 10 registrazioni audio. La quantità di materiale è dunque molto elevata, e preparare manualmente tutti i file di configurazione diventa un lavoro impossibile da completare in tempi accettabili. Sono stati quindi realizzati degli script *Python* che generano automaticamente tutti i file necessari per la

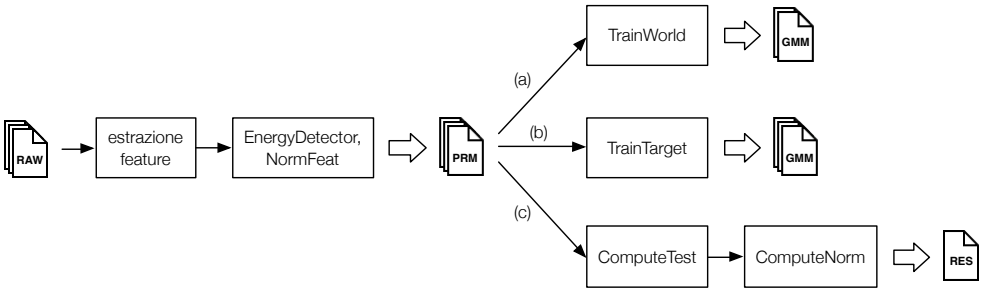

Figura 3.3: Schema di funzionamento con l'utilizzo di file contenenti audio precedentemente registrato. Una volta generati i file `prm` contenenti i vettori di feature, è possibile utilizzarli per le tre operazioni indicate dalle frecce (a), (b) e (c).

corretta esecuzione dei test. Sono stati realizzati degli script per la creazione dei seguenti file:

- index file contenenti i nomi dei modelli e i relativi file audio associati,
- index file con i test da effettuare,
- index file con i test aggiuntivi necessari per la normalizzazione degli score,
- list file contenente i nomi dei modelli da utilizzare per la generazione dell'UBM,
- list file contenente il nome dei file audio dai quali estrarre le feature.

Tutti i file sopracitati, ad eccezione dell'ultimo, vengono generati in due versioni: una relativa ai modelli maschili e una per quelli femminili. Tutti i test infatti vengono effettuati separatamente per uomini e donne a causa delle grandi differenze nelle loro caratteristiche vocali.
Vista la presenza di una grande quantità di file differenti, si è scelto di organizzare i file di configurazione in più sottocartelle, ognuna delle quali ne contiene una singola tipologia (ndx, lst, ...). Questo permette di avere i file organizzati in modo da trovare rapidamente quello che si cerca, nel caso sia necessario apportare qualche modifica.
Passiamo ora al funzionamento delle varie fasi del sistema di riconoscimento. Dal momento che sono presenti fin da subito tutti i modelli utilizzabili, la fase di estrazione delle feature viene eseguita una sola volta in modo da ottenere subito i file con i vettori di feature per tutte le registrazioni presenti nel database. Anche qui vengono generali i file `lbl` con l'utilizzo di `EnergyDetector`

e le feature vengono normalizzate tramite la funzione `NormFeat`.
Una volta ottenuti i file `prm` contenenti i vettori di feature, la prima cosa da fare è la generazione del modello di background. Questa operazione viene eseguita utilizzando il tool `TrainWorld`, che va a leggere i file di feature precedentemente creati e il file `lst` contenente i modelli da utilizzare.
Dopo aver prodotto l'UBM, è possibile passare ai modelli target. Questi vengono generati dal tool `TrainTarget`, che legge i file delle feature e l'index file in cui sono indicate le feature associate a ciascun modello.
Veniamo infine alla fase di testing, che si svolge in modo differente a quanto visto per l'approccio presentato nella precedente sezione. Infatti se prima veniva effettuato un solo test, tra la voce registrata e il presunto parlatore, in questo caso la necessità di valutare le prestazioni complessive del sistema richiede di effettuare un grande numero di test. Avendo a disposizione l'intero database, questi test possono essere eseguiti tutti insieme ed è sufficiente indicarli tutti nel relativo index file dato in ingresso a `ComputeTest`. In questo caso gli score vengono anche normalizzati mediante `ComputeNorm`, e questo richiede di effettuare dei test aggiuntivi come spiegato nella sezione 2.3.1. Una rappresentazione grafica dello schema di funzionamento è mostrata in Figura 3.3.
Si nota che in questo modo è possibile valutare le prestazioni del sistema per un utilizzo adibito sia a verifica che ad identificazione del parlatore.

Capitolo 4

Sperimentazione e risultati

In questo capitolo vengono presentati i test eseguiti con il sistema realizzato e i relativi risultati ottenuti. Viene inoltre descritto dettagliatamente il database utilizzato.
La sperimentazione è stata preceduta da una breve fase di preprocessing che ha permesso di adattare il database fornito al problema in esame. Successivamente sono stati eseguiti dei test sui principali parametri personalizzabili messi a disposizione dalla libreria Alize, in modo da capire come e in che modo questi influenzano le prestazioni. Questi test hanno permesso inoltre di trovare i valori ottimali per il caso in esame. Infatti molti parametri sono dipendenti dall'input e, anche in una situazione reale, vanno tarati in base a quello che ci si aspetta sarà ricevuto in ingresso dal sistema.
L'ultima sezione di questo capitolo presenta infine i risultati ottenuti, utilizzando i parametri individuati dai precedenti test, separatamente per i due casi di verifica ed identificazione.

4.1 Database utilizzato

I test sono stati effettuati utilizzando il database *TIMIT* che, nonostante sia un po' datato, presenta delle caratteristiche tutt'ora interessanti. Il database, progettato inizialmente per testare sistemi di riconoscimento del parlato e rivelatosi successivamente efficace anche nel caso di riconoscimento del parlatore, è in lingua inglese, e i parlatori sono stati selezionati dalle 8 maggiori regioni dialettali degli Stati Uniti in modo da includere persone con pronunce leggermente diverse dovute alle loro origini. Sono presenti in totale 630 parlatori, 438 uomini e 192 donne, per ognuno dei quali sono fornite 10 brevi registrazioni audio.

Tipo	Frasi pronunciate	Parlatori	Totale	Frasi/parlatore
Shibboleth (A)	2	630	1260	2
Compatte (X)	450	7	3150	5
Diverse (I)	1890	1	1890	3
Totale	2342		6300	10

Tabella 4.1: Organizzazione del database TIMIT

Queste registrazioni possono essere di tre tipi:

- ***shibboleth* dialettali**: uguali per tutti i parlatori, sono utili per valutare le differenze dialettali. Sono identificate dalla lettera A;
- **foneticamente compatte**: progettate per dare una buona copertura fonetica. Identificate dalla lettera X;
- **foneticamente diverse**: frasi lette da libri esistenti, con lo scopo di aggiungere varietà nel contesto fonetico. Indicate con la lettera I.

Le dieci registrazioni presenti per ogni parlatore sono così composte: 2 shibboleth dialettali uguali per tutti, 5 frasi foneticamente compatte, ognuna delle quali è letta da sette parlatori differenti, e infine 3 frasi foneticamente diverse, ognuna della quali viene letta da un solo parlatore. La Tabella 4.1 presenta un riassunto della struttura del database appena descritta.
Per ogni registrazione sono inoltre presenti dei file ausiliari contenenti la trascrizione di parole e fonemi associati all'istante temporale in cui vengono detti. Tuttavia queste informazioni non sono state utilizzate in quanto pensate per l'analisi di un sistema di riconoscimento del parlato.
I nomi dei file hanno una struttura precisa e consentono di identificarne il contenuto. La struttura del nome indica il sesso del parlatore, un numero identificativo, il tipo di registrazione contenuta e il numero della registrazione. Per esempio il file

```
F_111_i_2
```

indica che la registrazione è relativa al parlatore donna (F) numero 111, e contiene la seconda frase foneticamente diversa (i).
I file sono forniti in formato `raw` pcm 16 bit. La frequenza di campionamento relativamente elevata, aggiunta alla pulizia dell'audio fanno sì che questo database rappresenti delle condizioni pressoché ideali.

Per questo motivo viene fornita una seconda versione di ogni registrazione, che presenta l'aggiunta di rumore di fondo in modo da simulare una situazione di utilizzo in ambiente rumoroso, come può essere una strada o un locale affollato.

4.2 Preprocessing

Il database fornito era già stato utilizzato da un'altra persona, che lo aveva leggermente elaborato per renderlo adatto agli scopi del suo lavoro. Si è resa quindi necessaria una fase di preprocessing in modo da ricavare le registrazioni originali a partire da quelle modificate.
Ai file originali era stato aggiunto ulteriore rumore di fondo. Un'analisi del flusso audio delle registrazioni modificate ha evidenziato che erano stati creati semplicemente dei file stereo a partire dalle registrazioni mono originali, nelle quali era stato aggiunto un canale contenente il rumore. E' stato quindi sufficiente estrarre il canale contente la registrazione originale e salvare un nuovo file audio contente solamente il canale corretto.
In aggiunta, è stata eseguita un'operazione di *downsampling* di tutti i file ad una frequenza di campionamento di 8 kHz. La maggior parte delle ricerche in questo campo vengono infatti eseguite sulle linee telefoniche, la cui digitalizzazione è stata standardizzata utilizzando una frequenza di campionamento di 8 kHz. Questa scelta consente quindi di eseguire i test secondo gli standard adottati per questo genere di sistemi, ma anche di simulare il caso peggiore: ci si troverà infatti difficilmente ad avere una qualità inferiore in un utilizzo reale.

4.3 Modalità di test

In questa sezione verranno presentate le modalità con le quali sono stati eseguiti i test sul database TIMIT.
Vediamo inizialmente com'è stato suddiviso il database. La prima suddivisione riguarda il numero di parlatori da utilizzare per la generazione del modello di background, e il numero di parlatori singoli utilizzati per effettuare i test. Questa suddivisione è la stessa nei i due casi di verifica ed identificazione, a differenza delle modalità di test vere e proprie che vengono trattate separatamente nelle due relative sottosezioni.
Come visto nella Sezione 4.1, il database è composto da 630 parlatori, dei quali 438 uomini e 192 donne. La suddivisione nei due sottoinsiemi è stata eseguita seguendo le direttive degli autori del database, che suggerivano di

	Parlatori sinceri	Impostori	Test	Background
Uomini	72	40	112	326
Donne	36	20	56	136
Totale			168	462

Tabella 4.2: Suddivisione del database

utilizzare circa il 30% dei parlatori come target, e i rimanenti per la generazione del modello di background. Del totale di 630 parlatori ne sono stati quindi usati 462 per l'UBM, mentre dei rimanenti 168 sono stati generati i singoli modelli utilizzati per i test.
Il database è composto per circa il 70% da uomini, quindi si è cercato di mantenere questa proporzione anche nei due sottoinsiemi appena descritti: il sottoinsieme di parlatori utilizzati per il modello di background è formato da 326 uomini e 136 donne, mentre il sottoinsieme di 168 parlatori singoli è formato da 112 uomini e 56 donne. La Tabella 4.2 riassume queste suddivisioni.
Per quanto riguarda la generazione del modello di background sono state utilizzate tutte le registrazioni disponibili di ogni parlatore. Per i modelli target invece si è deciso di utilizzare sette registrazioni (le due shibboleth, le tre foneticamente diverse e due delle foneticamente compatte) per la generazione del modello, e le rimanenti tre per i test. Questo ha permesso di testare il sistema con dati nuovi, non incontrati durante la fase di training.
Infine, la fase di normalizzazione degli score richiede che vengano effettuati dei test aggiuntivi su un sottoinsieme di impostori selezionati tra i parlatori utilizzati per i test. E' stato quindi necessario effettuare un'ulteriore suddivisione e si è scelto di etichettare come impostori circa il 35% dei target, sia per gli uomini che per le donne. I valori esatti sono riportati nella Tabella 4.2.

4.3.1 Verifica del parlatore

Il problema verifica può essere visto come un caso di autenticazione, in cui vengono forniti in ingresso una registrazione audio e un modello target. Il sistema deve essere in grado di dire se l'identità del parlatore relativo alla registrazione fornita corrisponde a quella del target.
I test sono quindi stati eseguiti nel seguente modo: ogni modello target è stato testato verso tutte le registrazioni audio di parlatori sinceri disponibili,

in modo da poter verificare se il sistema riesce ad "autenticare" solamente i parlatori delle tre registrazioni corrette. Avendo a disposizione 72 target e 3 registrazioni per ognuno di essi, sono stati effettuati 216 test per ogni singolo target con un totale di 15552 test.
Per la normalizzazione degli score sono stati inoltre necessari i seguenti ulteriori test (vedi Sezione 2.3.1):

- *T-norm*: ognuna delle tre registrazioni audio disponibili per i test di ogni parlatore sincero è stata testata verso ogni modello di impostore;
- *Z-norm*: ognuna delle tre registrazioni audio disponibili per i test di ogni impostore è stata testata verso ogni modello di parlatore sincero;
- *ZT-norm*: ognuna delle tre registrazioni audio disponibili per i test di ogni impostore è stata testata verso ogni modello di impostore, tranne quello corretto.

Le prestazioni in questo caso sono state valutate sulla base delle curve DET e dei relativi EER e sono riportate nella Sezione 4.5.1.
Le curve DET sono state tracciate utilizzando uno script MATLAB fornito dal *National Institute of Standards and Technology (NIST)*.

4.3.2 Identificazione del parlatore

Il problema di identificazione è leggermente diverso da quello visto nella precedente sezione, e prevede di essere testato e valutato in un modo differente. In questo caso viene fornita in ingresso solamente una registrazione audio, e il sistema deve essere in grado di stabilire il corretto parlatore tra quelli conosciuti. Per valutarne le prestazioni, ogni registrazione disponibile per i test dei parlatori sinceri è stata testata verso ogni modello target, in modo da valutare se il sistema riesce a riconoscere il corretto parlatore. In un sistema ideale di questo tipo il modello corretto ottiene lo score più alto in ogni test relativo alle sue registrazioni.
Il numero totale di test effettuati è anche qui 15552, più quelli necessari alla normalizzazione degli score che vengono eseguiti allo stesso modo del caso precedente.
Per misurare le prestazioni in questo caso è stato necessario effettuare alcune operazioni sui risultati: questi sono stati divisi in blocchi, ognuno dei quali contenente i test eseguiti per una particolare registrazione, e ogni blocco è stato ordinato in modo decrescente secondo lo score. Questo ha permesso di calcolare agilmente precisione e indice di *mean rank*.
Inoltre sono stati estratti da ogni blocco i top-k modelli, in modo da valutare

entro quante posizioni dalla prima si trova il modello corretto. I risultati di questi test sono presentati nella Sezione 4.5.2.

4.4 Configurazione dei parametri

La quantità di parametri presenti nell'implementazione degli algoritmi della libreria Alize permette di adattare il sistema a qualsiasi esigenza. Molti di questi, però, non possono essere stabiliti a priori ma vanno determinati a seconda del caso in modo sperimentale. In questa sezione vengono esposti i parametri che più influenzano le prestazioni del sistema e la risposta dello stesso alla loro variazione.
Iniziamo con la fase di estrazione delle feature che prevede la scelta del **numero di coefficienti MFCC** da utilizzare. Come visto nella Sezione 1.1.1, normalmente vengono utilizzati 24 filtri e mantenuti 12 o 19 coefficienti con l'applicazione della trasformata coseno. I test effettuati hanno mostrato un netto miglioramento delle prestazioni con l'utilizzo di 19 coefficienti invece dei classici 12, riducendo l'EER da circa 7% a 4%. Tuttavia, utilizzando più di 19 coefficienti, le prestazioni peggiorano nuovamente in quanto è noto che i coefficienti di livello alto rappresentano cambiamenti rapidi dell'energia del segnale, che non sono rappresentativi per le caratteristiche vocali dell'individuo e tendono a peggiorare le prestazioni se considerati. Utilizzando meno di 12 coefficienti, invece, non si hanno abbastanza informazioni per rappresentare in modo adeguato la voce e le prestazioni generali del sistema ne risentono negativamente.
Passiamo ora alla fase di training. Per quanto riguarda la generazione del modello di background, un parametro fondamentale è il **numero di distribuzioni Gaussiane** da utilizzare. Questo parametro è strettamente dipendente dall'input in quanto il valore scelto dipende dalla quantità delle registrazioni, dalla loro durata e da quante informazioni utili si riescono ad estrarre da esse. Non esiste una regola per determinare a priori un valore adatto a questo parametro, ma va impostato a seconda del caso e solitamente dopo aver effettuato qualche test sul sistema da utilizzare. Normalmente viene utilizzato un numero di distribuzioni pari ad una potenza di due.
Per quanto riguarda il sistema realizzato, sono stati effettuati dei test con differenti valori per il parametro in questione, ricavando che il numero di distribuzioni ottimali per il database utilizzato è 512. I risultati dei test possono essere visti in Figura 4.1.
L'andamento delle prestazioni riscontrato a seconda del numero di distribuzioni Gaussiane utilizzato può essere spiegato nel seguente modo: utilizzando un numero di distribuzioni troppo basso non si riesce a modellare adeguata-

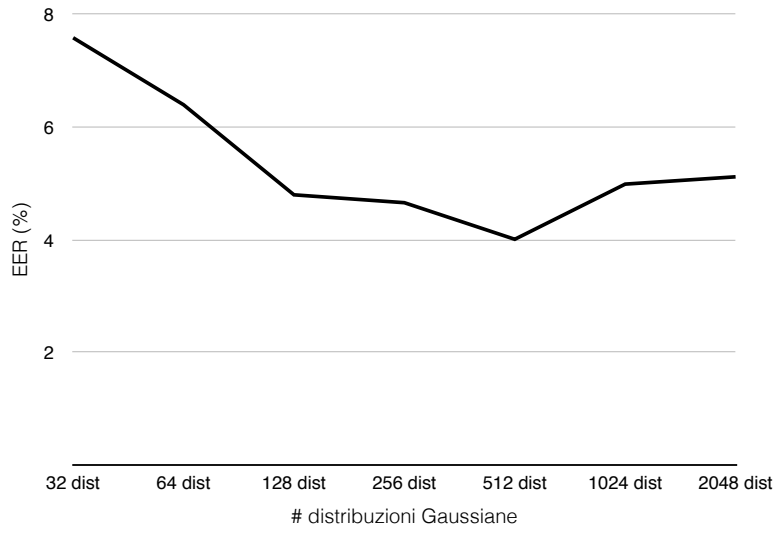

Figura 4.1: EER in funzione del numero di distribuzioni Gaussiane utilizzate per la generazione dei modelli.

mente l'insieme di feature estratto dalle registrazioni e in sostanza si perdono informazioni, ottenendo un modello poco accurato. Un numero di distribuzioni troppo elevato è ugualmente un problema nel caso in cui non si abbiano abbastanza dati in ingresso per riuscire a stimare correttamente i parametri della GMM, ottenendo quindi un modello nel quale alcune componenti delle misture non sono state allenate e presentano ancora i valori di inizializzazione, solitamente casuali.
Sempre rimanendo nella fase di training, un altro parametro rilevante è il **numero di iterazioni per il training dell'UBM**. Questo è meno dipendente dall'input rispetto al numero di distribuzioni, ma va ugualmente impostato a seconda del caso sulla base di alcuni test sperimentali. In Figura 4.2 si può vedere come varia l'EER a seconda del numero di iterazioni eseguite per la generazione del modello di background. Si può notare che anche qui è presente un valore che permette di ottenere prestazioni nettamente migliori rispetto agli altri, che equivale a 6 iterazioni. In questo caso un valore troppo basso porta a problemi di *under-training*, generando un modello poco accurato, mentre un valore troppo alto fa perdere generalità al modello, in quanto più è alto il numero di iterazioni e più il modello sarà specifico per quel particolare insieme di feature ricevute in ingresso. Dal momento che l'UBM ha lo scopo di modellare la voce in generale, e non uno specifico insieme di voci, un numero troppo alto di iterazioni non va bene e porta ad un

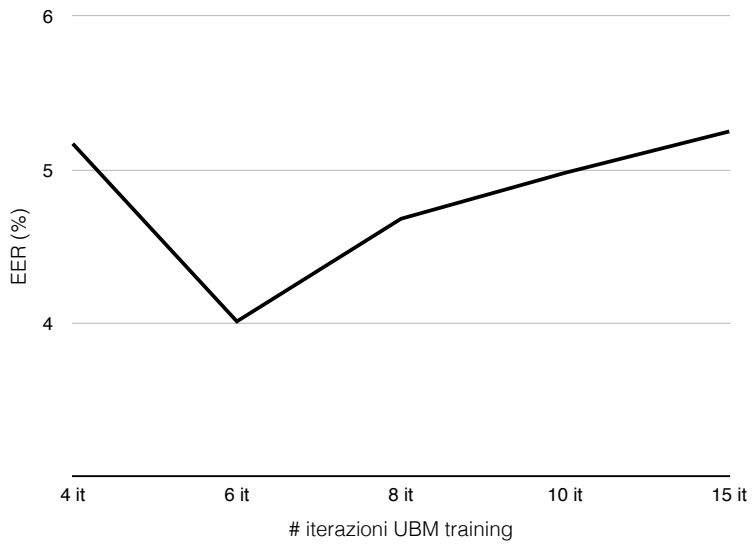

Figura 4.2: EER in funzione del numero di iterazioni eseguite nella fase di training del modello di background.

peggioramento delle prestazioni.

E' possibile inoltre variare il numero di iterazioni anche per la fase di training dei modelli target. Tuttavia i test effettuati non hanno portato a significative ripercussioni sulle prestazioni del sistema, indicando che il parametro, almeno per il dataset utilizzato, non è particolarmente significativo.

Infine si può analizzare un parametro su cui è possibile agire nella fase di testing. Questo parametro è chiamato **top-k** e fornisce la possibilità di considerare solamente le k distribuzioni più significative (determinate calcolando la log-verosimiglianza verso il modello di background e prendendo le k distribuzioni che ottengono lo score più alto) del modello target e valutare le feature in ingresso solamente su quelle specifiche distribuzioni. Questo permette di ridurre notevolmente i costi computazionali e renderli indipendenti dal numero di distribuzioni utilizzate. Il parametro è impostato di default a 10, e i test hanno dimostrato che la sua variazione non influisce particolarmente sulle prestazioni generali del sistema. Aumentando il valore infatti le prestazioni rimangono invariate, fino a che non si arriva a considerare quasi tutte le distribuzioni. Arrivati a quel punto le prestazioni iniziano a calare, fatto dovuto probabilmente ad un utilizzo di distribuzioni marginali che risultano poco allenate. In definitiva si è visto che le top-10 distribuzioni permettono di rappresentare in modo adeguato il modello, e la loro sola valutazione in

fase di testing permette di ridurre notevolmente i tempi di esecuzione senza sacrificare le prestazioni.

4.5 Risultati ottenuti

In questa sezione vengono presentati i risultati ottenuti, sulla base delle metriche di valutazione descritte nella Sezione 1.2. I parametri utilizzati per la configurazione del sistema sono quelli che hanno permesso di ottenere i risultati migliori nei test specifici presentati nella precedente sezione.
Sono stati effettuati dei test per entrambi i casi di verifica ed identificazione, le cui prestazioni vengono analizzate nelle due sezioni 4.5.1 e 4.5.2, rispettivamente. Questo ha permesso di valutare il comportamento del sistema in oggetto per entrambi gli utilizzi, evidenziandone punti forti e punti deboli.

4.5.1 Verifica del parlatore

I test effettuati hanno portato a dei buoni risultati per il caso di verifica su voci maschili e femminili pulite, mentre l'aggiunta di rumore di sottofondo ha peggiorato non di poco le prestazioni del sistema. Utilizzando la porzione di database comprendente solamente gli uomini e la normalizzazione Z è stato ottenuto un EER del 4.01%. Le prestazioni peggiorano utilizzando soggetti di sesso femminile, con un EER che scende al 6.24%. Questa differenza nelle prestazioni del sistema in base al sesso dei parlatori è un fenomeno noto in letteratura, ed è dovuto principalmente alle diverse caratteristiche anatomiche del tratto vocale di uomini e donne. La voce femminile (come anche quella dei bambini) è generalmente più acuta rispetto a quella maschile, e le formanti della voce sono posizionate ad una frequenza più alta rispetto a quelle maschili. Di conseguenza la scala Mel si adatta bene agli uomini in quanto i filtri sono concentrati al di sotto della frequenza di 1 kHz, regione nella quale sono presenti le principali formanti della voce maschile. Le formanti della voce femminile, avendo una frequenza più alta rispetto a quelle maschili, trovano nella scala Mel una minor concentrazione di filtri; questo fatto si traduce in un minor dettaglio e quindi una perdita di informazioni nella fase di estrazione delle feature, peggiorando di conseguenza le prestazioni del sistema.
L'utilizzo dei file rumorosi ha portato ad un peggioramento delle prestazioni che raggiunge il 340%, ottenendo un EER di 17.62%.

	No norm	T-norm	Z-norm	ZT-norm
Uomini	4.12	4.06	4.01	4.04
Donne	6.38	6.59	6.24	6.77
Uomini (+ rumore)	18.13	18.08	17.8	17.62

Tabella 4.3: Valori di EER (%) nel caso di parlatori maschi (con audio pulito e rumoroso) e femmine, utilizzando diversi tipi di normalizzazione.

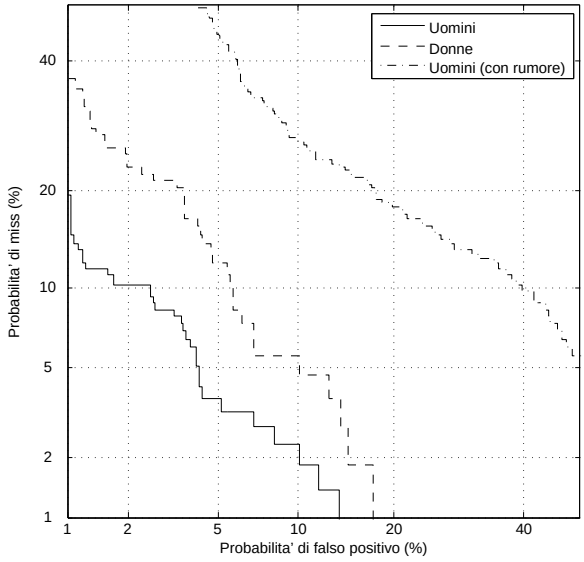

Figura 4.3: Curve DET relative ai sottoinsiemi testati: uomini (con e senza rumore) e donne. Le curve sono relative al caso di utilizzo della normalizzazione Z.

Si vede quindi come una bassa qualità audio (si ricorda infatti che è stato eseguito un downsampling a 8 kHz su tutti i file) in aggiunta ad un considerevole rumore di fondo, mettano in serie difficoltà il sistema di riconoscimento. Tutti i risultati dei test sono riportati nella Tabella 4.3. In Figura 4.3 sono invece riportate le curve DET per i tre casi esaminati, relative all'utilizzo della normalizzazione Z.
Si nota infine che le diverse tecniche di normalizzazione degli score utilizzate non portano a miglioramenti degni di nota rispetto all'utilizzo degli score di base, con casi in cui addirittura le prestazioni peggiorano (T-norm e ZT-norm sul database donne). Inoltre le normalizzazioni sono molto simili tra di loro

in termini di prestazioni, con nessuna delle tre che spicca rispetto alle altre. Solitamente le normalizzazioni permettono di migliorare sensibilmente le prestazioni, quindi i risultati ottenuti sono probabilmente da attribuire ad una particolare conformazione del database TIMIT utilizzato, che non beneficia in modo particolare da queste tecniche.

4.5.2 Identificazione del parlatore

Il problema di identificazione ha dimostrato un comportamento simile al caso precedente, con delle buone prestazioni utilizzando un segnale pulito e un drastico calo di prestazioni nel caso di file rumorosi.
La Tabella 4.4 mostra i risultati ottenuti. L'utilizzo di file rumorosi ha dimostrato anche qui di mettere in seria difficoltà il sistema, con una precisione raggiunta di appena 34.72%. La precisione raggiunge invece l'82.87% nel caso di audio pulito e voci maschili, mentre peggiora leggermente nella controparte femminile, con una precisione del 79.63%. La differenza delle prestazioni rispetto al sesso dei soggetti è da imputare agli stessi fenomeni esposti nella precedente sezione, sebbene in questo caso la differenza sia meno marcata.
Il rank medio è pressoché identico nei due casi che coinvolgono uomini e donne, con un valore di 1.449 e 1.444 rispettivamente. Anche l'indice MRR risulta molto simile nei due casi. Questi ultimi due risultati si possono tradurre nel fatto che il parlatore corretto ha sempre rank inferiore o uguale a 10. Questo fenomeno è riportato graficamente nelle Figure 4.4 e 4.5 dove non è stata riportata la percentuale di parlatori corretti aventi rank 1 (che corrisponde alla Precisione riportata nella Tabella 4.4) in quanto avrebbe reso il grafico poco leggibile dato il valore molto più grande degli altri. Da queste figure si vede non solo che il parlatore corretto è sempre nelle prime 10 posizioni, ma che i casi in cui si trova oltre la quinta posizione sono molto rari. Infatti, nel 97.21% dei casi per gli uomini e nel 98.14% per le donne, il parlatore corretto si trova tra i primi cinque candidati restituiti dal sistema.

	Precisione	MeanRank	MRR
Uomini	82.87	1.449	0.889
Donne	79.63	1.444	0.873
Uomini (+ rumore)	34.72	8.175	0.477

Tabella 4.4: Risultati ottenuti per il caso di identificazione.

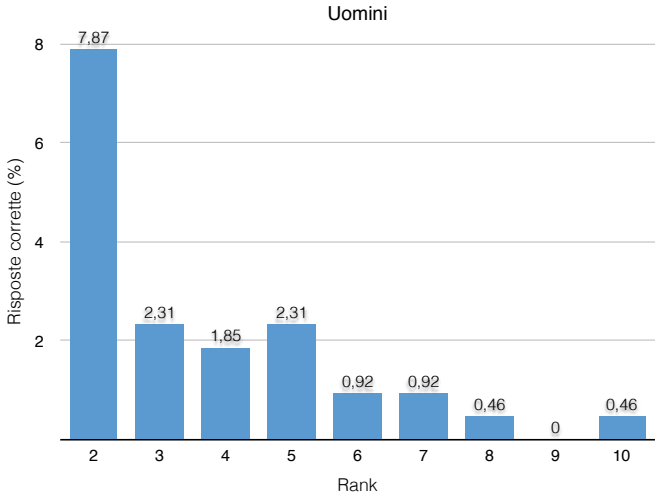

Figura 4.4: Percentuale di parlatori corretti con rank 2-10 nel caso maschile. La rimanente porzione ha rank 1.

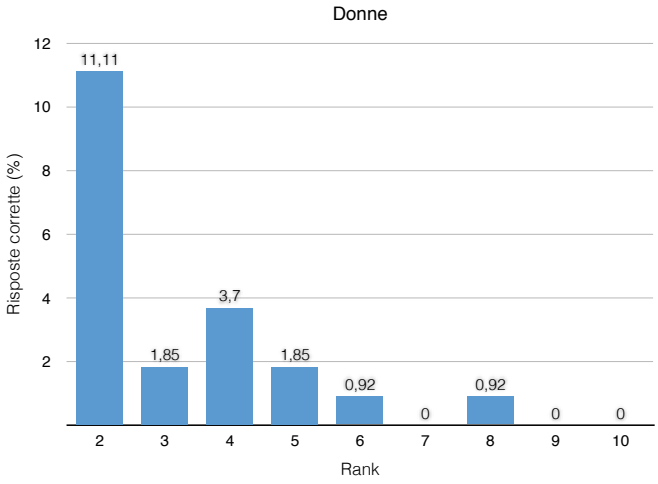

Figura 4.5: Percentuale di parlatori corretti con rank 2-10 nel caso femminile. La rimanente porzione ha rank 1.

Capitolo 5

Conclusioni

Il sistema realizzato in questa tesi ha permesso di valutare le prestazioni di un sistema di riconoscimento del parlatore utilizzando la tecnologia attualmente disponibile. La libreria Alize mette a disposizione degli sviluppatori una buona implementazione dei principali algoritmi necessari per la realizzazione di un sistema di questo tipo, sebbene la documentazione sia scarsa e poco curata. I risultati ottenuti sono stati soddisfacenti nonostante la bassa qualità dei file audio utilizzati che aumenta considerevolmente il livello di sfida, ma simula situazioni di utilizzo non ideali.
Come spiegato nel Capitolo 1, un sistema di riconoscimento del parlatore può essere adibito ad un utilizzo finalizzato ad eseguire un task di verifica (verificare l'identità di una persona) oppure identificazione (individuare l'identità della persona che sta parlando). I test hanno permesso di valutare le prestazioni del sistema per entrambi questi due casi.
Nel caso di verifica del parlatore, il sistema ha permesso di ottenere un EER di 4.01% nel caso di voci maschili, e di 6.24% nel caso femminile. Considerando la qualità dell'audio questi valori sono da considerarsi un buon risultato, in linea con l'attuale stato dell'arte. L'utilizzo di file rumorosi ha messo invece in serie difficoltà il sistema, dimostrando che il problema più grande per questa tecnologia risiede proprio nella qualità delle registrazioni, non solo riguardante il modo in cui le registrazioni sono state eseguite, come la scelta della frequenza di campionamento, ma soprattutto per quanto riguarda la pulizia del suono. Va comunque notato che i test eseguiti non hanno previsto nessuna fase di preprocessing delle registrazioni audio presenti nel database, se non per il downsampling che ne degrada ulteriormente la qualità. Sono presenti tecniche in letteratura che permettono di "pulire" il segnale audio, andando a ridurre il rumore di fondo ed enfatizzare la voce. L'utilizzo di queste tecniche permette molto probabilmente di migliorare sensibilmente le prestazioni del sistema nel caso di registrazioni rumorose.

Un fatto interessante emerso dai test riguarda i valori assunti dagli score nella risposta del sistema. Questi si sono infatti rivelati essere particolarmente dipendenti dal parlatore. Gli score relativi ai test di un particolare parlatore possono raggiungere valori massimi intorno a 1-1.2, ma arrivare solamente a 0.5-0.6 per altri. La normalizzazione degli score dovrebbe mitigare proprio questo problema ma, nel caso del dataset utilizzato, non ha portato a significativi miglioramenti, con gli score che rimangono nella maggior parte dei casi strettamente dipendenti dal singolo parlatore. Questo significa che esistono casi in cui una soglia globale, come quella utilizzata nei test, non assicura il risultato migliore in termini di prestazioni. In casi come questo, l'utilizzo di una soglia adattiva e determinata appositamente per ogni modello potrebbe portare ad un significativo incremento delle prestazioni.
Per il caso di identificazione del parlatore sono state raggiunte precisioni di 82.87% e 79.63% per uomini e donne, rispettivamente. Valgono invece le stesse considerazioni fatte per il caso precedente per quanto riguarda l'utilizzo di file rumorosi. Il fatto interessante emerso da questi test è che il parlatore corretto si trova in circa il 98% dei casi nelle prime cinque posizioni tra i possibili parlatori ricevuti in risposta dal sistema, su un totale di 72 nel caso degli uomini e 36 per il caso femminile. Questo fatto apre molte strade su possibili modi alternativi di affrontare il problema: potrebbe infatti essere eseguita una prima scrematura in modo da ridurre notevolmente lo spazio delle alternative, concentrandosi sui primi cinque risultati, per poi eseguire dei test aggiuntivi e specifici su questi parlatori, cercando di individuare con precisione quello corretto.
Questi potrebbero essere dei possibili sviluppi futuri legati al lavoro svolto, insieme ad una validazione più approfondita eseguita utilizzando diversi dataset.

Bibliografia

[1] J. Benesty, M. Sondhi, and Y. Huang, *Springer Handbook of Speech Processing*. Springer Berlin Heidelberg, 2008.

[2] J. N. Holmes, *Speech Synthesis and Recognition*. Van Nostrand Reinhold (UK), 1988.

[3] F. Bimbot, J. françois Bonastre, C. Fredouille, G. Gravier, I. Magrin-chagnolleau, S. Meignier, T. Merlin, J. Ortega-garcía, D. Petrovska-delacrétaz, and D. A. Reynolds, "A tutorial on text-independent speaker verification," Oct. 04 2014.

[4] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, 2010.

[5] D. A. Reynolds, "Automatic speaker recognition using gaussian mixture speaker models," *The Lincoln Laboratory Journal*, vol. 8, no. 2, pp. 173–192, 1995.

[6] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics* (S. Z. Li and A. K. Jain, eds.), pp. 659–663, Springer US, 2009.

[7] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[8] D. E. Sturim, D. A. Reynolds, R. B. Dunn, and T. F. Quatieri, "Speaker verification using text-constrained gaussian mixture models," in *ICASSP*, pp. 677–680, IEEE, 2002.

[9] J. S. Mason and J. Thompson, "Gender effects in speaker recognition," Sept. 26 1995.

[10] N. C. Ward and D. R. Dersch, "Text-independent speaker identification and verification using the TIMIT database," 1998.

[11] J.-F. Bonastre, N. Scheffer, D. Matrouf, C. Fredouille, A. Larcher, A. Preti, G. Pouchoulin, N. W. D. Evans, B. G. B. Fauve, and J. S. D. Mason, "ALIZE/spkdet: a state-of-the-art open source software for speaker recognition," in *Odyssey*, p. 20, ISCA, 2008.

[12] A. Larcher, J.-F. Bonastre, B. G. B. Fauve, K.-A. Lee, C. Lévy, H. Li, J. S. D. Mason, and J.-Y. Parfait, "ALIZE 3.0 - open source toolkit for state-of-the-art speaker recognition," in *Interspeech*, pp. 2768–2772, ISCA, 2013.

[13] D. A. van Leeuwen and N. Brümmer, "An introduction to application-independent evaluation of speaker recognition systems," in *Speaker Classification (1)* (C. A. Müller, ed.), vol. 4343 of *Lecture Notes in Computer Science*, pp. 330–353, Springer, 2007.