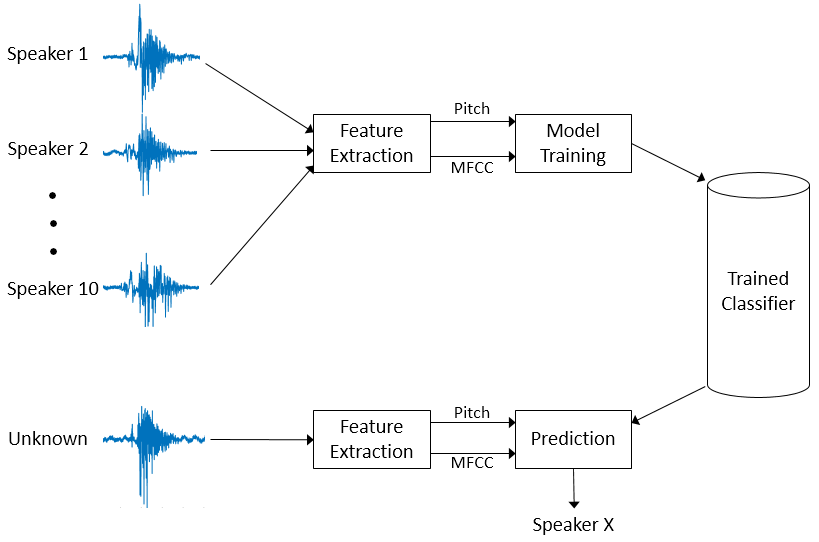
Speaker Recognition

Durante il processo di analisi e riconoscimento della voce di un utente sono stati seguiti i seguenti step:

1. Il dato è ottenuto in un formato ottimale per le funzioni utilizzate negli step successivi
2. Si ottiene il MFCC (Mel-frequency cepstral coefficient) dal dato vocale
3. Il dato ottenuto si confronta con i dati presenti nel data set
4. Una volta trovato il dato più simile a quello posto in input viene mostrata la “distanza” tra i due elementi
5. La distanza viene convertita in percentuale per mostrare all’utilizzatore quanto il sistema sia sicuro della corrispondenza tra i due tratti vocali

Di seguito sono analizzati i punti uno per uno.

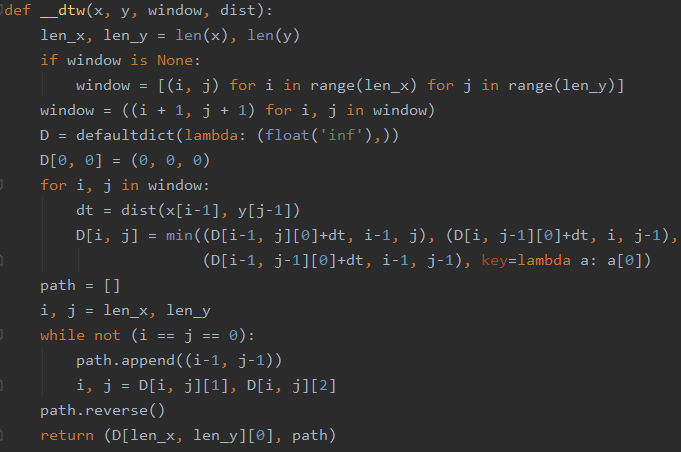
1. La voce viene registrata per un numero di secondi, frequenza di campionamento e numero di canali fissato in precedenza. La registrazione è stata effettuata attraverso la funzione *rec* appartenente alla libreria *python-sounddevice* dalla quale si ottiene un dato del tipo *numpy.ndarray* salvato poi secondo il formato *.*wav (WAVEform audio file format) che si presta perfettamente alle operazioni presenti nelle fasi successive
2. La funzione che permette l’ottenimento del MFCC dal file in formato wav prende il nome di *extract\_mfcc\_from\_wav\_file* appartenente al file *tools* alla libreria speaker-verification-toolkit. La funzione richiede il percorso del file con estensione wav ed eventuali ulteriori parametri aggiuntivi quali la frequenza di campionamento, la lunghezza della finestra di analisi e un eventuale salto, in termini di tempo, tra finestre successive. All’interno viene utilizzata la funzione *load* della libreria librosa, la quale ha il compito di ottenere, a partire da un file audio, una serie con elementi di tipo floating point che lo rappresentino e la funzione *mfcc* appartenente al file base della libreria *python\_speech\_features*, la quale utilizza la serie pocanzi menzionata per calcolare l’MFCC e mostrarlo come dato del tipo numpy.array avente lunghezza pari al numero di frames del dato analizzato. Nello specifico, la funzione si serve di altrettante:

* Fbank: calcola le caratteristiche dell’energia di Mel-Filterbank, cioè del banco di filtri che separa il segnale in input in più componenti, ognuno con una singola frequenza di sub-banda del segnale originale. Utilizza funzioni appartenenti al file sigproc che contiene funzioni utili per processare il segnale. Sono usate funzioni per dividere un frame in frame sovrapposti, attuare una pre-enfasi e calcolare lo spettro di potenza (Lo spettro di potenza di una serie temporale descrive la distribuzione di potenza in componenti di frequenza che compongono quel segnale) di ogni frame
* Dct: funzione di scipy che ritorna la trasformata coseno discreta di una sequenza x
* Lifter: applica un cepstral lifter (un filtro che opera nel dominio delle frequenze) alla matrice dei cepstra (plurale di cepstrum: risultato di una trasformazione matematica nel campo dell’analisi di Fourier)

Gli MFCC sono coefficienti che collettivamente costituiscono un MFC (Cepstrum mel-frequency)

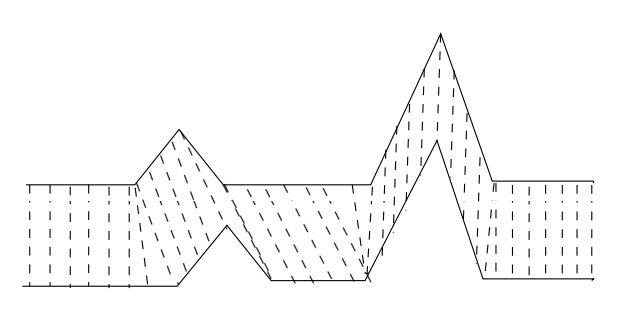
1. Il confronto tra l’MFCC trovato e quelli dei dati immagazzinati nel dataset avviene secondo la funzione *find\_nearest\_voice\_data* appartenente alla libreria *speaker\_verification\_toolkit* la quale, attraverso la chiamata alla funzione *compute\_distance* della medesima libreria, confronta gli MFCC di due segnali audio in modo da ottenere un dato di tipo float che rappresenti la distanza minima tra i due campioni. Il calcolo della distanza viene effettuato attraverso l’utilizzo della funzione *fastdtw*, parte dell’omonima libreria. La funzione si applica su dati aventi complessità tempo-spaziale del tipo O(n), calcolabile come il rapporto tra il tempo utilizzato e la quantità di memoria occupata per eseguire una funzione di una determinata lunghezza. Tuttavia, la funzione *fastdtw* non fa che da “intermediario” per il calcolo della distanza tra i due campioni poiché prepara l’input e chiama a sua volta la funzione *\_\_fastdtw* che, ricorsivamente, riduce la lunghezza dei due campioni ed aumenta la grandezza della “finestra”, cioè della matrice su cui poi verrà chiamato l’algoritmo. Il calcolo vero è proprio è effettuato nella funzione *\_\_dtw* presente nello stesso file .py di tutte le altre.

La sigla dtw presente nella funzione si riferisce al Dynamic Time Warping: un algoritmo che permette l'allineamento tra due sequenze, e che può portare ad una misura di distanza tra le due sequenze allineate. Particolarmente utile per trattare sequenze in cui singole componenti hanno caratteristiche che variano nel tempo, nell’implementazione utilizzata si presenta come:



Il parametro dist è una funzione, un intero>0, o il valore None: se un intero indica il numero di norma che sarà usato; se una funzione sarà usata al posto del calcolo della norma; se None sarà usato il valore assoluto della differenza tra x[i] e y[j]

1. Si sottolinea come il fatto che venga mostrata una distanza non sia sinonimo di riconoscimento da parte del sistema. In caso di distanza ritenuta grande il sistema non riconoscerà l’utente.
2. Per una corretta e più semplice visione del risultato, la distanza è mostrata sotto forma di percentuale corrispondente al grado di certezza che il sistema ha nell’identificare il parlante con uno degli utenti presenti nel data set. Tuttavia, il Dynamic Time Warping non è in grado di assicurare una certezza pari allo 0% che due audio siano diversi: non esiste una distanza massima calcolabile con questo algoritmo. Per questo motivo, durante le fasi di test del sistema, sono state dichiarate sia una soglia massima che definisca una distanza massima raggiungibile sia una percentuale oltre la quale si differenziano le due aree di certezza ed incertezza, in modo che il sistema sappia sempre come comportarsi davanti al dato numerico della distanza ottenuta dal Dynamic Time Warping



Raccolta dei dati

Il dataset è creato a partire da raccolte di audio appartenenti a vari utenti. Per ogni utente è stato salvato più di un campione. È stato richiesto di utilizzare timbri vocali diversi, ambienti con rumori diversi e parole di riconoscimenti diverse. Ogni audio possiede un id incrementale ed univoco. Vengono richiesti all’utente 3 audio.





Viene ottenuto il percorso del file di registrazione poiché da esso si potrà estrarre l’MFCC che sarà poi salvato nel dataset.

Addestramento

Durante la fase di addestramento i campioni vengono aggiunti al dataset. Da ogni audio ottenuto nella fase precedente viene calcolato l’MFCC secondo le modalità spiegate pocanzi.

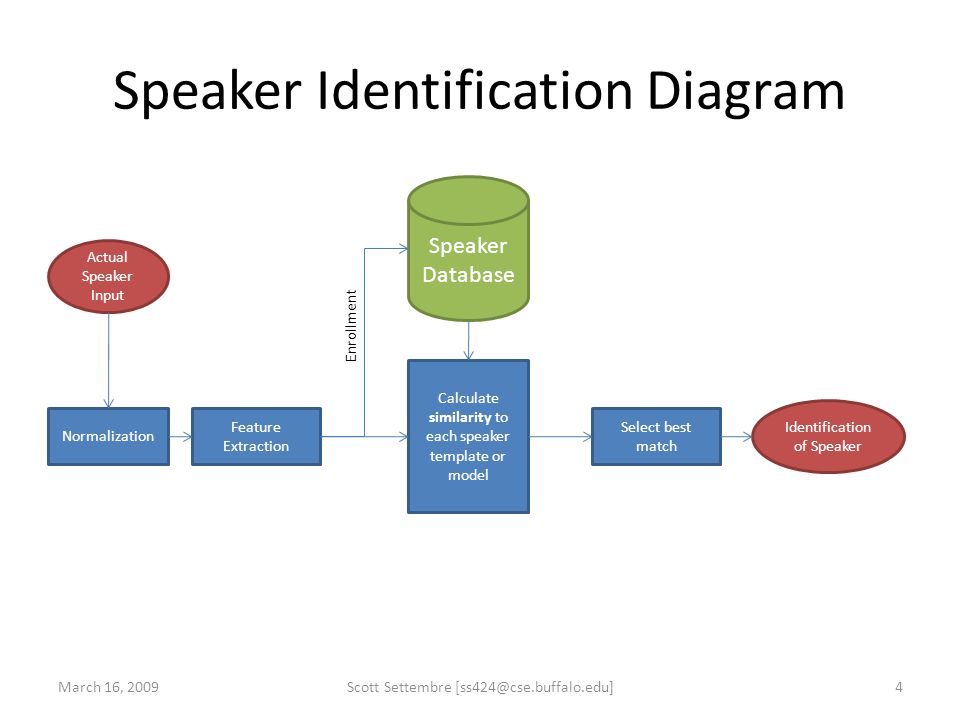
Riconoscimento

Nella fase di riconoscimento verrà acquisito un ulteriore audio, dal quale verrà estratto il MVCC per poi essere confrontato con quelli presenti nel dataset. Sarà mostrata la distanza minima tra due MVCC trovata e l’elemento del dataset dal quale confronto è risultato il valore. L’algoritmo normalizza la distanza trovata per farla rientare in un range e dare una immediata idea di quanto i due audio siano simili tra loro. In base alla distanza ottenuta saremo in grado di dire la probabilità che la voce registrata dall’utente in questa fase sia simile alla voce, secondo l’algoritmo, più simile del dataset

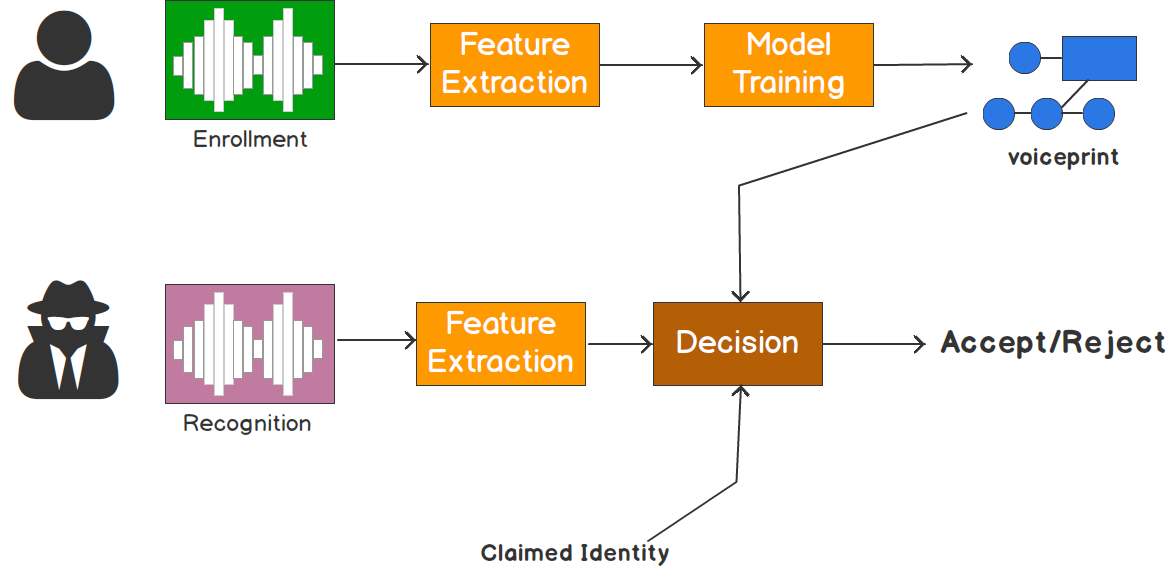
VERSIONE 2

Lo speaker recognition è il processo di riconoscimento automatico del parlatore sulla base di informazioni individuali presenti nel segnale vocale. Possiamo parlare di Speaker identification se otteniamo quale parlatore presente nel data set ha la maggiore probabilità di essere il parlante; speaker verification se si accetta o rigetta l’affermazione implicita, fatta dal parlate, di essere uno degli utenti registrati.

Speaker identification: per essere attuata dati di addestramento e di test devono essere diversi. Essenzialmente si tratta di un sistema di riconoscimento del tipo 1:N nel quale l’identità di un parlatore è confrontata con N campioni posti durante la fase di addestramento



Speaker verification: per essere attuata, si deve lavorare sugli stessi dati sia per il test che per l’addestramento. Essenzialmente si tratta di un sistema di riconoscimento 1:1 nel quale l’identità di un parlatore è confrontata e verificata con l’identità dichiarata di un parlatore specifico presente tra i campioni.



Ciò che viene fatto nel nostro algoritmo sta nell’attuare dapprima una fase di speaker identification per ottenere una possibile identità dichiarata e, una volta fatto ciò attuare una fase speaker verification per controllare che l’utente soddisfi i requisiti per essere ritenuto davvero chi la fase precedente abbia comunicato.

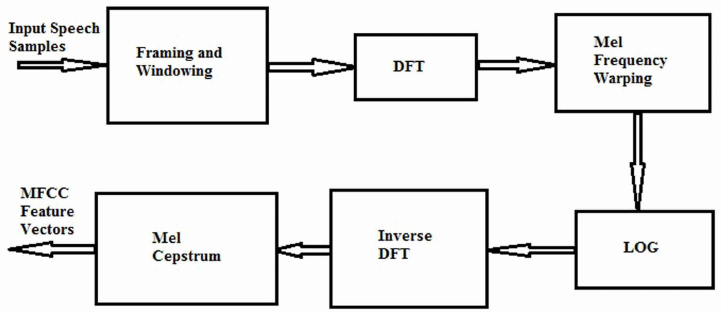
Tutto ciò può essere riassunto in cinque fasi:

1. Acquisizione
2. Pre-processing
3. Estrazione delle feature
4. Addestramento
5. Testing

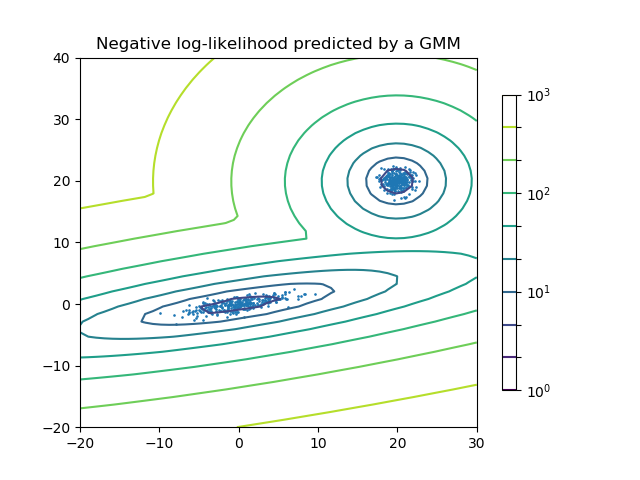
1: l’acquisizione dei dati è avvenuta sia attraverso la ricerca e l’utilizzo di dataset online quale ad esempio Common Voice di Mozilla sia attraverso la realizzazione di un dataset personale basato su un numero di audio tale da poter costruire un modello ottenuti da un numero discreto di individui. A questi ultimi è stato richiesto di muoversi in ambienti sia privi sia con rumore per riuscire a testare al meglio l’algoritmo. Al fine di ottenere una qualità migliore nei modelli, agli utenti è stato richiesto di inserire un numero minimo di tre audio. Una volta registrati questi ultimi, all’utente sarà richiesto se voglia inserirne altri per migliorare la precisione del modello. Una volta generato un modello i file audio saranno automaticamente cancellati al fine di rispettare i dati sensibili dell’utente

2: pre-processing: ad essere applicati sui campioni sono stati rimozione del silenzio e riduzione del rumore. La rimozione del silenzio avviene attraverso l’utilizzo della funzione rms\_silence\_filter appartenente alla libreria speaker\_verification\_toolkit la quale adatta la soglia di treshold sulla base dei primi frame di audio. La funzione di noise removal è stata implementata attraverso l’utilizzo di un butterworh filter applicato attraverso la libreria scipy.signal. Il filtro si applica ottenendo un grado di smoothing del grafico rappresentante l’audio, nel nostro caso è 1 per permettere uno smooth lento, e una frequenza di cutt-off normalizzata ottenuta scegliendo una frequenza di cut-off e dividendola per la frequenza di Nyquist. Prima di applicare queste due tecniche il dato viene letto attraverso la funzione load appartenente alla libreria librosa che si occupa di convertire l’audio in un vettore di floating point. Dalla suddetta funzione otteniamo anche un intero maggiore di zero che indica il tasso di campionamento del nostro file wav.

3: in questa fase ci concentriamo su due caratteristiche fondamentali per il nostro algoritmo: Mel frequency Cepstral coefficients (MFCC) e la sua derivata, chiamata Delta-MFCC: il calcolo degli MFCC avviene grazie alla funzione extract\_mfcc appartenente alla libreria speaker\_verification\_toolkit; il calcolo della derivata avviene per mezzo della funzione delta appartenente alla libreria librosa. Si sottolinea l’utilizzo della funzione scale appartenente alla libreria sklearn, la quale attua una standardizzazione del dataset lungo ogni asse, requisito comune per molti stimatori in ambito machine-learning. Se le feature non fossero standardizzate gli stimatori potrebbero dare risultati inaspettati. I coefficienti vengono infine combinati e salvati in un’apposita cartella. I MFCC si concentrano su una serie di calcoli che usano il Cepstrum con un asse di frequenza non lineare che segue la scala Mel. Per ottenere melcepstrum, il segnale audio è diviso in finestre usando dapprima una finestra di analisi e poi attraverso l’utilizzo della trasformata di Fourier. Il calcolo degli MFCC si suddivide in 6 fasi:



4: Nella Fase di addestramento ciò che viene fatto è generare un Gaussian Mixture Model a partire dagli MFCC memorizzati per ogni singolo utente. Ciò è attuato attraverso l’utilizzo della funzione GaussianMixture appartenente alla libreria sklearn, la quale opera su una matrice contenente tutti i coefficienti appartenenti ad un singolo utente, e della funzione fit che immette i dati nella matrice. Infine, il file viene salvato per mezzo della funzione dump appartenente alla libreria pickle. La funzione fit stima i parametri del modello grazie all’algoritmo EM (expectation maximization) il quale stima la massima somiglianza statistica attraverso un procedimento iterativo. Sklearn.mixture è un package che permette di stimare Gaussian Mixture Models a partire da dati. Un GMM è un modello probabilistico che assume che tutti i data points sono generati da una combinazione di un numero finito di distribuzione gaussiane con parametri sconosciuti.



5: Durante la fase di test i GMM sono letti e, per ognuno di essi, è ottenuta una somma di score mostrata su scala logaritmica e basata sul confronto tra i modelli presenti ed un audio immesso in input da un utente talvolta registrato talvolta sconosciuto al sistema. Lo score è ottenuto grazie alla funzione omonima appartenente al Modello. Lo score maggiore appartiene a chi, secondo il sistema, appartiene con più probabilità la voce immessa.