Università degli Studi di Padova

Scuola di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Riconoscimento fonetico per la terapia dei disturbi del linguaggio: dalla pratica logopedica alla sperimentazione su piattaforma mobile

Laureando

Mattia Bovo

Relatore

Prof. Antonio Rodà

Correlatore

Prof. Carlo Fantozzi

Anno Accademico 2015/2016

Alla mia famiglia.

Sommario

Negli ultimi anni le tecnologie per il riconoscimento vocale hanno conosciuto un notevole sviluppo e ciò ha consentito di rendere gli assistenti vocali, ormai presenti nella maggior parte dei dispositivi mobili e non solo, sempre più sofisticati. Tuttavia, esistono contesti, come quello logopedico, in cui si rende necessario acquisire una trascrizione fonetica pura e semplice di quanto pronunciato dal paziente senza che vengano compiute correzioni o interpretazioni di ciò che viene pronunciato. In questa tesi si intende spiegare e sperimentare un meccanismo di riconoscimento ed analisi dei fonemi per la terapia dei disturbi del linguaggio. In particolare, viene illustrato un algoritmo per dispositivi mobili in grado di riconoscere gli eventuali difetti di pronuncia di un paziente a partire dalla trascrizione fonetica di quanto questi ha pronunciato. Per raggiungere questo obiettivo si è cercata una stretta collaborazione con l'ambiente logopedico al fine di acquisire le corrette e complete conoscenze preliminari, necessarie in questo campo di applicazione.
Verrà dunque presentata una panoramica sullo stato dell'arte dei servizi e delle librerie più diffuse che offrono funzionalità di *Automatic Speech Recognition* e verrà messo in luce come, in generale, sia ancora poco diffuso lo studio e la ricerca di soluzioni al problema del riconoscimento fonetico. Per valutare i risultati ottenuti con l'integrazione degli algoritmi di riconoscimento degli errori di pronuncia è stato utilizzato uno dei più famosi test di valutazione dell'articolazione utilizzato dai logopedisti: il test di Fanzago, che potrà in futuro essere integrato all'interno del progetto LogoKit. I risultati dei test effettuati hanno evidenziato infatti che la maggior parte dei fonemi viene riconosciuta in maniera affidabile dal classificatore fonetico, con percentuali d'errore nel riconoscimento che non superano il 30% per la maggior parte di essi. Questo lavoro di tesi vuole essere un punto d'incontro fra l'informatica e l'ambiente logopedico, nonché un punto di partenza per rendere sempre più smart e digitali gli strumenti a supporto del lavoro dei logopedisti.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Il contesto operativo

Durante il corso dell'ultimo decennio, visto il costante aumento di dispositivi digitali e la conseguente crescita del volume di dati ed informazioni da essi generati, si è assistito ad un rapido e costante sviluppo nel settore del *Machine Learning* e del *Data Mining*. Tuttavia, la vera rivoluzione ed innovazione che ha caratterizzato gli ultimi anni non è tanto la quantità di informazione accessibile attraverso i dispositivi smart che ci circondano, quanto piuttosto le modalità di interazione con essi per ottenere l'accesso alle informazioni. Oggi, ad esempio, è sufficiente chiedere al proprio smartphone o tablet una particolare informazione o dire di cosa si ha bisogno e subito esso è in grado di fornire indicazioni dettagliate al riguardo instaurando un vero e proprio dialogo con l'utente. Si evince, quindi, che il riconoscimento vocale, parallelamente a molte altre tecnologie, ha compiuto grandi passi avanti negli ultimi tempi e moltissime società, tra le principali *Apple*, *Google* e *Microsoft*, hanno investito e stanno tuttora investendo su questo campo al fine di spingere al massimo le potenzialità dei propri assistenti vocali e garantire ai propri utenti servizi sempre migliori. In quest'ottica, la voce umana rappresenta un tratto biometrico dal quale è possibile estrarre una miniera di informazioni sul soggetto parlante. Le situazioni quotidiane in cui solitamente si necessita e si fa uso dell'assistente vocale sono fra le più svariate ed è normale che in situazioni simili, in cui magari si è impegnati in altre attività (ad esempio si è alla guida o, più in generale, si hanno le mani occupate), si pronuncino parole scorrettamente o troppo frettolosamente. Può anche succedere che se ci si trova in ambienti particolarmente rumorosi l'audio registrato presenti imperfezioni e discontinuità, oppure può accadere che il soggetto che fa uso del riconoscitore vocale presenti difetti di pronuncia o disturbi del linguag-

gio. Il compito principale del riconoscitore vocale associato ad un assistente vocale non si limita dunque al semplice rilevamento di quanto pronunciato, ma, per garantire un funzionamento ottimale, opera una serie di analisi sulla voce, sul parlato e su tutto l'audio registrato al fine di interpretare al meglio ed eventualmente correggere quanto detto. Esistono circostanze ed ambiti in cui però si rende necessario riconoscere e trascrivere precisamente quanto pronunciato dal soggetto, indipendentemente dalla presenza di errori o difetti di pronuncia di quest'ultimo. È il caso in cui si voglia applicare questa tecnologia in ambito logopedico, a supporto dell'attività che i logopedisti svolgono durante le sedute con i pazienti, il più delle volte bambini o preadolescenti. A tal proposito, nel 2014, una collaborazione tra Logopedia del dipartimento di Neuroscienze ed Ingegneria Informatica del dipartimento di Ingegneria dell'Informazione, entrambi dell'Università degli Studi di Padova, ha dato vita ad un progetto innovativo chiamato *LogoKit* [1].

1.2 LogoKit

LogoKit prevede l'implementazione di un set di applicazioni per dispositivi mobili dotati di sistema operativo *Android*, al fine di fornire al logopedista una serie di strumenti multimediali validi e completi in grado di garantire il coinvolgimento attivo dei pazienti, prevalentemente bambini di età compresa tra i tre e i dieci anni [2]. Il progetto, nella sua idea originale, prevede lo sviluppo di quattro applicazioni utili al logopedista per svolgere determinate attività con i bambini. Ognuna di esse è rivolta a particolari patologie di carattere logopedico, come ad esempio i disturbi specifici del linguaggio (DSL) e la disprassia verbale evolutiva[1] [3]. Le quattro applicazioni mirano ad aiutare i bambini durante la terapia anche attraverso l'utilizzo di figure e suoni, ed hanno tutte lo scopo di migliorare e curare differenti disturbi specifici. Nello specifico:

- *Prime Frasi*: mira ad assistere i logopedisti nella terapia di disturbi come DSL, disprassia verbale evolutiva, disturbi dello spettro autistico e disturbi secondari del linguaggio.

[1] La disprassia verbale evolutiva è un disturbo del sistema nervoso centrale che comporta difficoltà di programmazione dei movimenti articolatori necessari alla produzione dei suoni. La difficoltà di articolare insieme tali movimenti e di ordinarli nella giusta sequenza impedisce una corretta formulazione di parole e frasi.

- *Senti la mia voce*: mira ad aiutare i logopedisti nella terapia di disturbi come DSL, disprassia verbale evolutiva, disturbi secondari del linguaggio, disfonia[2], disartria evolutiva[3] e mutismo selettivo.
- *I movimenti buffi*: mira ad assistere i logopedisti nella terapia di disturbi come DSL, disprassia verbale evolutiva, squilibrio muscolare orofacciale e deglutizione deviata, disartria evolutiva e disturbi secondari del linguaggio.
- *Pronti, foni, via!*: punta ad aiutare i logopedisti nella terapia di disturbi come DSL, disturbi di articolazione e disturbi secondari del linguaggio.

Ad oggi, soltanto una di queste quattro applicazioni è stata realizzata ed è ora in fase di aggiornamento e sperimentazione: si tratta di *Prime Frasi*, il cui sviluppo è cominciato nel febbraio del 2015. L'applicazione assiste il logopedista durante le prime sedute con un nuovo paziente e consente di monitorare nel tempo i suoi progressi, frutto della terapia somministrata. L'obiettivo principale è quello di individuare la presenza di difetti di pronuncia e problemi legati all'articolazione dei principali organi dell'apparato fonatorio. All'interno di *Prime Frasi*, infatti, il logopedista ha la possibilità di creare delle sedute personalizzate per ogni paziente, mettendo a punto, per ognuna di esse, un set di frasi differente da fargli pronunciare e ripetere, così da verificare l'eventuale presenza di disturbi specifici del linguaggio. Le frasi presenti nell'applicazione sono molto brevi e vengono sottoposte al paziente sotto forma di immagini; esse permettono di testare quanto più possibile l'inventario fonetico del paziente così da avere, fin da subito, un quadro generale della sua situazione. L'applicazione dà inoltre la possibilità di registrare quanto pronunciato durante ogni seduta così da permettere al logopedista di riascoltare il paziente anche dopo diverso tempo e poter quindi valutarne al meglio i progressi.

L'interfaccia di *Prime Frasi* è pensata appositamente per far interagire il paziente durante la seduta: infatti, come si può osservare in Figura 1.1, una parte dell'interfaccia è rivolta verso il paziente e mostra le immagini corrispondenti alla frase che egli deve pronunciare, mentre l'altra è rivolta verso il logopedista, il quale può far avanzare la seduta alla frase successiva, può far partire la registrazione e riascoltare subito quanto registrato e può inoltre contrassegnare ciascuna frase registrata come "pronunciata correttamente"

[2]La disfonia è un disturbo della voce che va dalla raucedine fino all'afonia.

[3]La disartria evolutiva è l'incapacità di articolare in modo adeguato le parole a causa di una lesione del sistema nervoso centrale o periferico.

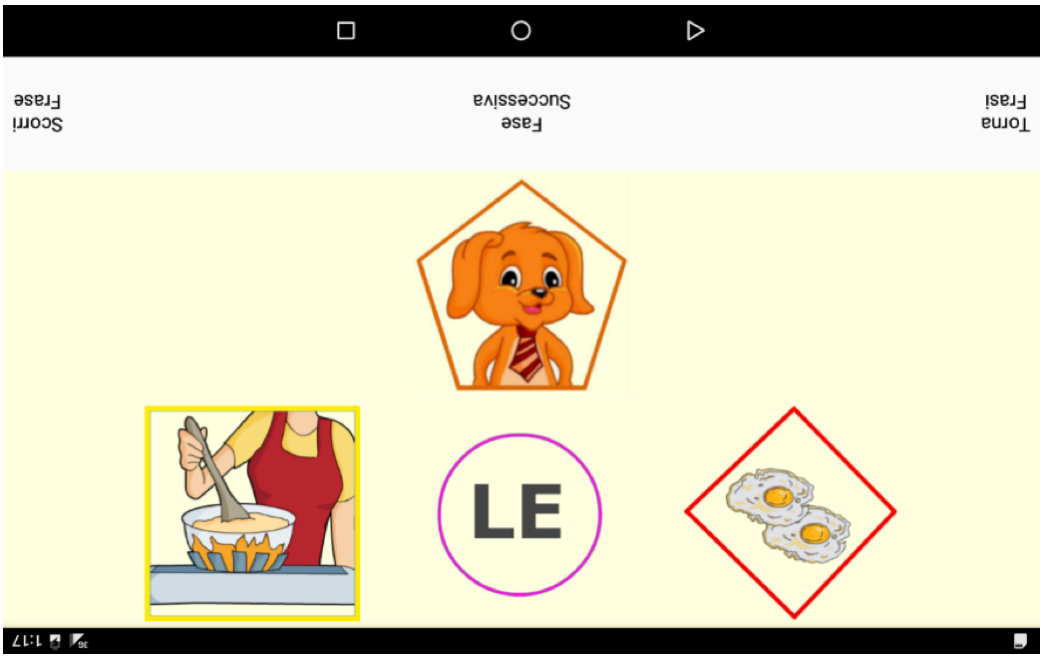

Figura 1.1: L'interfaccia di *Prime Frasi*.

oppure no. Più precisamente, una seduta tipo è costituita da tre momenti per ciascuna frase che dev'essere pronunciata: durante la prima parte, mediante un tocco sulle immagini visualizzate a schermo, il paziente ha la possibilità di ascoltare la corretta pronuncia delle parole che costituiscono la frase da pronunciare; durante la seconda parte il paziente pronuncia la frase mentre il logopedista avvia la registrazione: tale registrazione può essere immediatamente riascoltata dal logopedista e, come detto poc'anzi, può essere contrassegnata come "corretta" o "errata" in base ai difetti di pronuncia riscontrati. Durante la terza ed ultima parte la registrazione viene fatta riascoltare al paziente mentre sullo schermo scorre un'animazione che mima l'azione della frase appena pronunciata.

È evidente che per poter individuare i difetti di pronuncia del paziente e poter quindi segnare la frase appena pronunciata come "corretta" o "sbagliata", il logopedista deve scrivere su carta la trascrizione fonetica di quanto pronunciato dal paziente e, avendo in mente la corretta trascrizione fonetica di quanto avrebbe dovuto pronunciare, riscontrare "a mano" la presenza di eventuali errori. Consapevoli di questo, al fine di automatizzare e migliorare le funzionalità di *Prime Frasi*, e con lo scopo di renderne l'utilizzo meno macchinoso da parte del logopedista, si è pensato di arricchire l'applicazione con

tre nuove funzionalità, le quali vengono ora brevemente descritte, insieme ai vantaggi che esse portano alla *user experience* del logopedista:

1. Rendere automatico il riconoscimento e l'estrazione della voce del paziente bambino: in questo modo il logopedista non deve più preoccuparsi di registrare esattamente nel momento in cui parla il bambino, ma può tranquillamente interagire con lui parlando durante la registrazione; l'applicazione sarà infatti in grado di riconoscere e segmentare l'audio automaticamente, isolando la sola voce del bambino.

2. Riconoscimento automatico dei fonemi a partire dalla registrazione della voce del bambino: il logopedista non deve più nemmeno preoccuparsi di effettuare manualmente la trascrizione fonetica di quanto pronunciato dal bambino, in quanto, anche questa, viene effettuata in automatico dal software e viene successivamente visualizzata a schermo.

3. Individuazione automatica degli errori di pronuncia: anche in questo caso il logopedista non deve effettuare, manualmente o mentalmente, il confronto fra la corretta trascrizione fonetica della frase o della parola in esame e la trascrizione fonetica di quanto pronunciato dal bambino, poiché è sempre l'applicazione che, in automatico, evidenzia i fonemi pronunciati scorrettamente oppure che sono stati omessi o aggiunti.

Realizzare queste funzionalità ed automatizzare quindi lo svolgimento degli esercizi previsti da ogni seduta non è immediato e coinvolge l'approfondimento, lo studio e la conoscenza di diverse aree, a partire dalle metodologie e dalle pratiche utilizzate in ambito logopedico, passando per le tecniche di *Automatic Speech Recognition* (ASR) e riconoscimento fonetico e arrivando, infine, all'approfondimento di tecniche ed algoritmi in grado di effettuare il riconoscimento automatico dell'errore attraverso il confronto di trascrizioni fonetiche differenti.

L'obiettivo di questo lavoro di tesi è quello di realizzare, dopo aver acquisito le necessarie conoscenze nel contesto logopedico in cui si vuole intervenire, un meccanismo di riconoscimento ed analisi dei fonemi su piattaforma mobile, concentrandosi, in particolare, sull'implementazione e l'integrazione in *Prime Frasi* di un algoritmo in grado di riconoscere gli errori di pronuncia attraverso l'analisi delle trascrizioni fonetiche di quanto pronunciato dai pazienti. Per approfondire, invece, lo studio e la realizzazione di un algoritmo in grado di discriminare automaticamente la voce adulta da quella dei bambini si rimanda ai riferimenti bibliografici [1] e [4], mentre per l'approfondimento del classificatore fonetico utilizzato in *Prime Frasi* si rimanda a [5].

Di seguito, nel Capitolo 2 saranno illustrati ed approfonditi gli strumenti utilizzati e le conoscenze preliminari necessarie al raggiungimento dell'obiettivo finale; nel Capitolo 3, saranno spiegati nel dettaglio il lavoro svolto e le modalità con cui è avvenuta l'implementazione degli algoritmi sulla piattaforma mobile Android; nel Capitolo 4 saranno illustrati i test sperimentali, con i relativi risultati, atti a valutare e verificare le performance di quanto ottenuto; infine, il Capitolo 5 lascerà spazio alle conclusioni e ai possibili sviluppi futuri di quanto svolto finora.

Capitolo 2

Strumenti utilizzati e conoscenze preliminari

2.1 L'alfabeto fonetico internazionale

Fu nel 1886, quando per la prima volta un gruppo di professori di linguistica francesi ed inglesi si riunì a Parigi, che prese vita quella che sarebbe diventata la più prestigiosa associazione di fonetica al mondo: l'Associazione Fonetica Internazionale (AFI), o, in inglese, International Phonetic Association (IPA) [6] [7]. In principio, l'obiettivo di questi esperti era quello di promuovere l'uso della scrittura fonetica nelle scuole al fine di aiutare i bambini ad imparare più facilmente la pronuncia delle lingue straniere e, più in generale, per appoggiare l'insegnamento e l'alfabetizzazione dei più piccoli. Ad oggi, lo scopo dell'associazione è quello di promuovere lo studio scientifico della fonetica e le sue applicazioni pratiche. Nel corso degli anni, l'Associazione Fonetica Internazionale mise a punto e pubblicò un alfabeto fonetico per la rappresentazione fonetica di tutte le lingue, la cui notazione venne accettata dai linguisti di tutto il mondo: si tratta dell'Alfabeto Fonetico Internazionale o, in inglese, International Phonetic Alphabet (IPA), la cui primissima versione venne ufficialmente utilizzata per la prima volta nel 1888 [8].

Questo alfabeto è indipendente dalla scrittura e dall'ortografia eventualmente usate nei diversi sistemi linguistici e propone un inventario di elementi grafici e segni diacritici[1] con un'interpretazione il più possibile univoca da un punto di vista articolatorio [9]. Durante la sua storia, l'alfabeto ha subito una serie di revisioni, tra cui, una delle più importanti, è stata codificata

[1] I segni diacritici sono segni grafici che, posti sopra, sotto o accanto ad una lettera dell'alfabeto o ad un simbolo fonetico, ne indicano una particolare pronuncia.

nella *IPA Convention* di Kiel nel 1989. Dopo aver subito ulteriori revisioni nel corso degli anni novanta e nel primo decennio degli anni duemila, la versione dell'alfabeto attualmente in uso è stata revisionata e pubblicata nel 2015 (vedi [10]). Le revisioni periodiche dell'alfabeto, a cui prendono parte i maggiori studiosi di fonetica al mondo, consistono nell'integrazione di nuovi suoni eventualmente individuati in lingue precedentemente non conosciute e nella modifica della terminologia e dell'inventario degli elementi grafici che costituiscono l'alfabeto, in modo da accogliere le acquisizioni più recenti della teoria fonetica. La maggior parte dei simboli che costituiscono l'IPA sono presi dall'alfabeto latino e dall'alfabeto greco, associando, in particolare, alle minuscole di entrambi gli alfabeti i suoni principali. Altri simboli che lo costituiscono sono apparentemente non correlati a qualunque alfabeto: si tratta spesso di lettere già esistenti, girate o capovolte, alle quali talvolta vengono aggiunti simboli, come puntini o gancetti, in determinati luoghi. L'Associazione Fonetica Internazionale ha cercato di far corrispondere il più possibile ogni suono al suo rispettivo simbolo così da facilitarne la scrittura ed il riconoscimento.

La codifica di questo alfabeto è assai importante poiché esso rappresenta uno standard che consente di trascrivere in maniera univoca i suoni, o, più precisamente, i *foni*, di tutte le lingue conosciute: questo è possibile perché ad ogni simbolo IPA corrisponde un solo suono e viceversa, senza possibilità di confusione tra lingue diverse. La potenza e l'importanza di questo alfabeto risiede inoltre nel fatto che esso consente di effettuare trascrizioni fonetiche, e quindi di annotare la pronuncia, anche di lingue che non hanno una tradizione scritta e di lingue che non possiedono una grafia alfabetica, come ad esempio il cinese e il giapponese. Per le lingue che possiedono una grafia alfabetica, l'ideale sarebbe che ad ogni grafema, cioè ad ogni unità minima della lingua scritta, corrispondesse un *fonema*. Ciò, tuttavia, non è quasi mai possibile, in quanto il numero di simboli che occorrono per rappresentare tutti i suoni di una determinata lingua sono in genere in numero maggiore rispetto alle lettere che costituiscono l'alfabeto di quella lingua. Mediamente, gli inventari fonetici di una lingua si aggirano sulle 30 unità e, in generale, possono variare in un range che va da 15 a 60 fonemi. Nella lingua italiana ad esempio, l'alfabeto ortografico è composto da 21 segni, ma manca di segni specifici per rappresentare alcuni dei 30 fonemi che in totale rappresentano tutti i suoni ammessi dalla nostra lingua [11].

È evidente quindi che, anche se l'alfabeto fonetico internazionale offre oltre centosessanta simboli per trascrivere il parlato solo un sottoinsieme relativamente piccolo di questi viene impiegato per trascrivere i suoni di ciascuna

lingua. È inoltre importante sottolineare il fatto che una trascrizione in IPA di una data espressione linguistica non rappresenta una descrizione oggettiva di quest'ultima. Infatti, con il medesimo alfabeto fonetico, è possibile realizzare trascrizioni fonetiche con un diverso livello di dettaglio: in quella che è generalmente chiamata "trascrizione fonetica *larga*" si indicano soltanto le particolarità fonetiche più rilevanti, mentre un uso progressivamente più dettagliato dei segni diacritici consente trascrizioni sempre più "*strette*", cioè in grado di cogliere anche le proprietà fonetiche più particolari di una singola parola o di un determinato insieme di esse.

2.1.1 L'alfabeto fonetico della lingua italiana

I 30 simboli fonetici utilizzati per rappresentare i suoni dell'italiano sono costituiti da 7 vocali, 21 consonanti e 2 semiconsonanti. Vocali, consonanti e semiconsonanti possono essere viste come 3 classi maggiori in cui vengono suddivisi i suoni in relazione al modo in cui essi vengono prodotti. Per capire però più nel dettaglio come vengono classificati i fonemi appartenenti a queste tre classi di suoni e poter quindi comprendere il posizionamento di ciascun fonema all'interno delle tabelle IPA, è bene analizzare in modo approfondito quali sono i parametri principali che vengono utilizzati per classificare un suono e quale specifico ramo della fonetica si occupa di questo. La *fonetica* infatti, che è la scienza che studia i suoni del linguaggio articolato (detti in termini tecnici *foni*) in quanto entità fisiche, si suddivide metodologicamente in tre distinte branche ([11] [12] [9]):

- la **fonetica articolatoria**: studia il modo con cui l'espressione linguistica è prodotta dal parlante;
- la **fonetica acustica**: si occupa di studiare le proprietà fisiche del linguaggio parlato, in quanto composto di onde sonore che si propagano nell'aria;
- la **fonetica uditiva**: osserva il processo di ascolto e percezione dei suoni linguistici da parte dell'ascoltatore.

È proprio la fonetica articolatoria la disciplina che si occupa di studiare la produzione dei suoni attraverso l'apparato fonatorio. Quest'ultimo è costituito da diversi organi, alcuni fissi e altri mobili, che, con il loro movimento, o con le loro rispettive posizioni, modificano il flusso d'aria proveniente dai polmoni, dando così origine ai diversi suoni del linguaggio. Gli organi fissi sono i denti superiori, il palato e il naso, mentre gli organi mobili, o articolatori, sono le corde vocali, la mascella, le labbra, il velo palatino e la lingua.

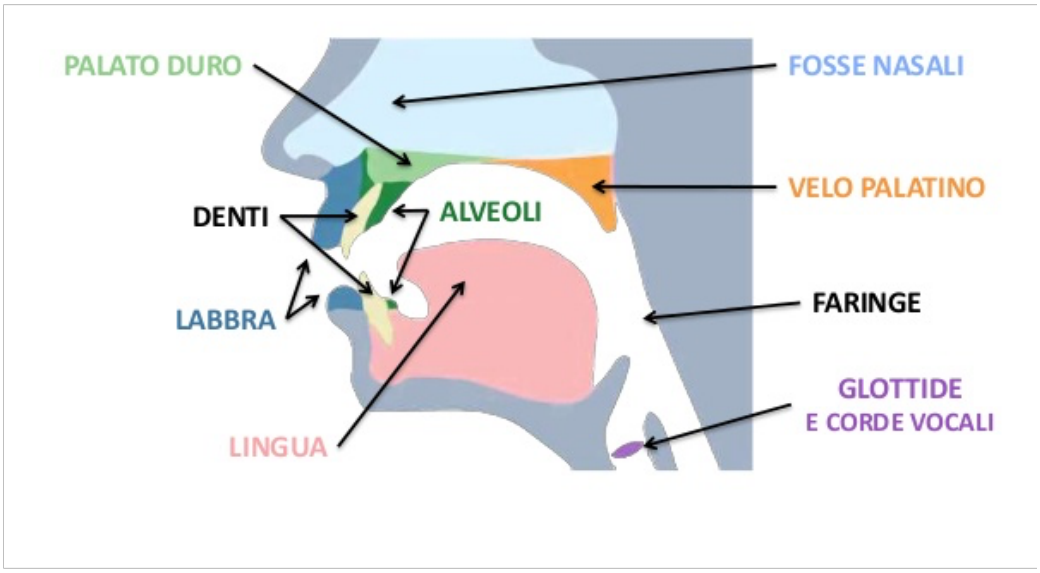

Figura 2.1: L'apparato fonatorio.

In relazione all'apparato fonatorio e ai suoi organi, di cui è riportato uno schema in Figura 2.1, esistono tre parametri principali che permettono di classificare un suono [9] [13] [14]:

- **modo di articolazione**: è legato ai vari assetti (posizioni) che i vari organi assumono nella produzione di un suono e riguarda soprattutto la natura dell'ostacolo che si frappone al passaggio dell'aria. Si distinguono grazie a questo parametro suoni:
  - **occlusivi**: un organo mobile (generalmente la lingua) tocca un organo fisso, ostruendo completamente il passaggio dell'aria: il suono è così la piccola esplosione che si ottiene rilasciando bruscamente l'ostruzione; ad esempio per il suono *t* la lingua premuta contro i denti impedisce il passaggio dell'aria: appena la posizione si rilassa, l'aria esplode con un rumore caratteristico che è appunto quello della *t*.
  - **fricativi**: gli organi articolatori sono avvicinati l'uno all'altro ma non si toccano: l'aria è così costretta a passare attraverso uno stretto canale producendo un fruscio, che è poi il suono specifico.
  - **affricati**: sono i suoni che combinano le caratteristiche degli occlusivi con quelle dei fricativi: nella *z* della parola *azione*, ad esempio, l'ostruzione non viene liberata del tutto con un'esplosio-

ne: i due organi rimangono vicini e l'aria fruscia fra essi, come se si pronunciassero una t e una s a distanza molto ravvicinata.

- **nasali**: se durante la produzione di un suono entrano in risonanza anche le cavità del naso (questo avviene se si abbassa il velo palatino e si lascia che l'aria entri nel naso) i suoni che derivano sono detti nasali.
- **laterali**: per produrre un suono laterale la lingua si posiziona contro i denti e l'aria fuoriesce dai due lati della lingua stessa.
- **vibranti**: il suono è prodotto dalla vibrazione dell'apice della lingua o dell'ugola. Questi suoni vengono generati mediante una debole occlusione intermittente del canale orale, la quale si interrompe e ripristina velocemente più volte, creando un ciclo occlusione + esplosione nel quale l'aria che proviene dai polmoni è prima costretta e poi bruscamente rilasciata producendo un suono periodico. Quando il ciclo è unico, cioè se la debole occlusione avviene una volta sola, si ha un modo di articolazione *monovibrante*; in caso contrario, si parla semplicemente di *vibrante* o *polivibrante*.
- **approssimanti**: gli organi articolatori vengono avvicinati ma senza contatto. Appartengono a questa categoria semivocali e semiconsonanti.
- **vocalici**: sono suoni nei quali l'aria passa liberamente senza incontrare alcun ostacolo; il loro timbro particolare è dato dalla posizione della lingua e delle labbra. Chiaramente, appartengono a questa categoria di suoni tutte le vocali.

- **luogo o punto di articolazione**: si riferisce ad ognuno dei vari punti del tratto vocale in cui il flusso d'aria necessario per produrre un suono può essere modificato. Per comprendere meglio la posizione dei vari punti di articolazione in relazione all'apparato fonatorio si faccia riferimento alla Figura 2.2. In base al luogo di articolazione si distinguono suoni:

  - **bilabiali**: questi suoni sono caratterizzati dall'occlusione di entrambe le labbra.
  - **labiodentali**: l'aria attraversa in questo caso una fessura che si forma appoggiando gli incisivi superiori al labbro inferiore.
  - **dentali**: nella produzione di questi suoni la parte anteriore della lingua (la lamina) tocca la parte interna degli incisivi.

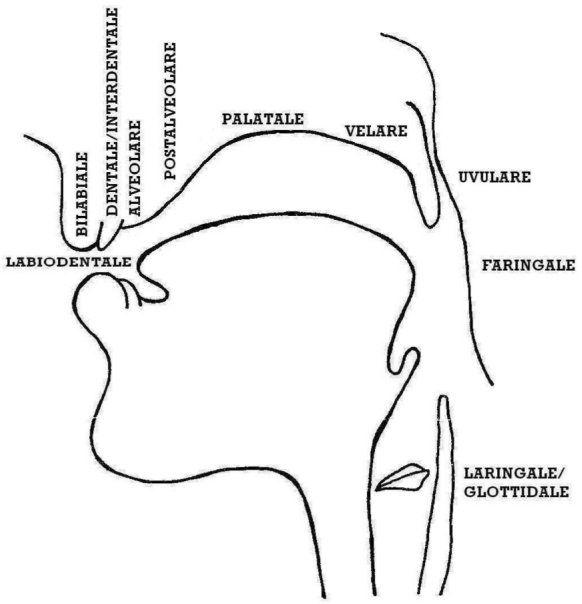

Figura 2.2: I luoghi di articolazione fonatoria.

  - **alveolari**: in questi suoni la lamina della lingua tocca o si avvicina agli alveoli.
  - **palato-alveolari**: in questo caso la lamina della lingua si avvicina sempre agli alveoli, ma assumendo una posizione arcuata rispetto a quest'ultimi.
  - **palatali (o anteriori)**: in questi suoni l'intera lingua si avvicina al palato.
  - **velari (o posteriori)**: questi suoni sono prodotti con la lingua che tocca il velo palatino.

- **sonorità**: è data dalle vibrazioni delle corde vocali; se queste vibrano si produce un suono sonoro, se non vibrano, un suono sordo. Più nello specifico, i suoni consonantici possono essere sordi o sonori: l'aria che proviene dai polmoni viene lasciata passare liberamente dalle corde vocali nelle consonanti sorde, mentre in quelle sonore l'aria mette in vibrazione le corde vocali accostate. Le vocali, invece, sono tutte sonore.

In base ai parametri appena definiti, è possibile classificare i 23 simboli dell'alfabeto fonetico italiano utilizzati per rappresentare i suoni consonantici e

LUOGO → MODO ↓	BILABIALI	LABIO-DENTALI	DENTALI	ALVEOLARI	PALATO-ALVEOL.	PALATALI	VELARI
OCCLUSIVE	p b		t d				k g
FRICATIVE		f v		s z	ʃ		
AFFRICATE				ts dz	tʃ dʒ		
NASALI	m			n		ɲ	
LATERALI				l		ʎ	
POLIVIBRANTI				r			
SEMICONSONANTI O APPROSSIMANTI						j	w

Tabella 2.1: Fonemi consonantici e semiconsonantici dell'alfabeto fonetico italiano.

semiconsonantici all'interno della Tabella 2.1, che posiziona ciascun fonema in base al proprio *modo* di articolazione e al proprio *luogo* di articolazione. Quando due consonanti sono posizionate all'interno di una stessa casella, quelle più a sinistra sono da considerarsi sorde, mentre quelle più a destra sono da considerarsi sonore.
I suoni che rappresentano le vocali sono ulteriormente classificati sulla base di quattro coefficienti [9]:

- **anteriorità-posteriorità**: questo parametro è determinato, in una rappresentazione molto schematica, dalla posizione avanzata, centrale o arretrata del dorso della lingua: il mutamento di tale posizione causa una variazione nel volume della cavità orale che è minore nelle vocali anteriori e maggiore in quelle posteriori. Si distinguono quindi vocali anteriori, vocali centrali e vocali posteriori.

- **grado di altezza**: ancora in modo molto schematico è possibile associare questo coefficiente alla posizione più o meno alta del dorso della lingua. Di grande rilevanza per il mutamento del grado di altezza è anche la posizione della radice della lingua che determina il volume della cavità faringale: questo è minore nelle vocali basse e maggiore nelle vocali alte. Per cui il rapporto tra il volume interno della cavità orale e quello della cavità faringale assume i valori maggiori nelle vocali basse. Si distinguono vocali alte, vocali medio-alte, vocali medio-basse e vocali basse.

- **labbra**: posizione delle labbra durante la produzione della vocale che possono essere arrotondate (con gli angoli della bocca accostati tra loro; si determina così un prolungamento del tratto vocale), o non arrotondate: in posizione di riposo o in posizione distesa, con gli angoli della bocca spinti verso l'esterno. Si distinguono quindi vocali arrotondate e vocali non arrotondate.

- **nasalizzazione**: il velo palatino, durante la produzione della vocale, può trovarsi in posizione alzata o abbassata. Nel primo caso l'aria, prima di uscire all'esterno, passa esclusivamente nella cavità orale. Nel secondo invece il flusso dell'aria è diviso dal velo palatino: una parte passa per la cavità orale, e una parte fuoriesce passando per le cavità nasali, provocando un alterazione del timbro della vocale. Si distinguono pertanto vocali orali e vocali nasali.

I fonemi che rappresentano le 7 vocali dell'alfabeto fonetico italiano vengono quindi, in base ad alcuni dei parametri appena descritti, posizionati all'interno del cosiddetto triangolo vocalico o sistema eptavocalico, di cui è presente una rappresentazione in Tabella 2.2.

AVANZAMENTO → ALTEZZA ↓	ANTERIORE (O PALATALE)	CENTRALE	POSTERIORE (O VELARE)
ALTE (CHIUSE)	i		u
MEDIO-ALTE (SEMICHIUSE)	e		o
MEDIO-BASSE (SEMIAPERTE)	ɛ		ɔ
BASSE (APERTE)		a	

Tabella 2.2: Fonemi vocalici dell'alfabeto fonetico italiano.

2.1.2 IPA e SAMPA: standard a confronto

Per effettuare la trascrizione fonetica di parole o frasi, i simboli fonetici più diffusi e maggiormente utilizzati sono quelli codificati nello standard dell'alfabeto fonetico internazionale (IPA), tuttavia esistono contesti e applicazioni, soprattutto informatici, nei quali tali simboli richiedono di essere codificati diversamente, affinché essi possano essere letti e processati senza errori da un computer. A tal proposito, lo standard SAMPA è stato concepito appositamente per aggirare l'impossibilità da parte di alcuni software di utilizzare codifiche di testo in grado di rappresentare correttamente tutti i simboli IPA.

Lo standard SAMPA, acronimo di *Speech Assessment Methods Phonetic Alphabet*, è costituito da un insieme di caratteri ASCII a 7 bit ed è stato messo a punto alla fine degli anni ottanta dello scorso secolo da una commissione europea nell'ambito dell'*European Strategic Program on Research in Information Technology* (ESPRIT) basandosi sull'alfabeto IPA [15].

Inizialmente, lo standard SAMPA rappresentava un sottoinsieme limitato dei simboli IPA ed era in grado di codificare i fonemi di appena sei lingue europee; ad oggi lo standard si è evoluto e può essere utilizzato per codificare i fonemi di 29 lingue diverse, compreso l'italiano. Una variante dello standard SAMPA, chiamato X-SAMPA, ovvero *Extended-SAMPA*, fu sviluppata nel 1995 da John C. Wells, professore di fonetica all'università di Londra. L'obiettivo era quello di estendere la codifica SAMPA in modo da ricoprire l'intero range dei caratteri IPA, codificandoli sempre in caratteri ASCII a 7 bit [16].

Quando nel corso dell'ultimo decennio, lo standard di codifica dei caratteri *Unicode* ha cominciato a supportare la rappresentazione della maggior parte dei simboli IPA, la necessità di avere a disposizione un sistema di codifica dei caratteri per rappresentare i simboli IPA in ASCII diminuì sensibilmente; tuttavia, i sistemi e i dispositivi di input ancora oggi utilizzati non permettono l'inserimento di una così vasta gamma di caratteri ed è per questo motivo che gli standard SAMPA e X-SAMPA sono tuttora ampiamente utilizzati, soprattutto nelle tecnologie di riconoscimento e analisi dei fonemi. Si è reso necessario, anche nel software messo a punto per l'applicazione *Prime Frasi* di LogoKit, riconoscere e processare i fonemi utilizzando la codifica SAMPA, per convertire, soltanto alla fine dell'elaborazione, ciascun fonema in codifica IPA. In questo modo è stato possibile visualizzare i fonemi nell'interfaccia dell'applicazione secondo la codifica maggiormente utilizzata dai logopedisti. Alcuni strumenti che si sono rivelati chiave per riuscire a mappare i fonemi dalla codifica SAMPA alla codifica IPA sono stati la tabella IPA multimediale e l'IPA-SAMPA Converter messi a disposizione dal Laboratorio di Fonetica Sperimentale "Arturo Genre" di Torino (vedi, rispettivamente, [17] e [18]).

2.2 I test di valutazione dell'articolazione

Per poter essere in grado di prevenire, riconoscere, classificare e successivamente curare i molteplici disturbi specifici del linguaggio che colpiscono numerosi pazienti in età prescolare e scolare, i logopedisti si avvalgono di strumenti messi a punto e studiati appositamente per lo scopo. Durante le

prime sedute con un nuovo paziente è assolutamente necessario campionare, testare e valutare l'inventario e lo sviluppo fonetico di quest'ultimo, così da poter costruire un quadro generale che metta in evidenza la pronuncia lacunosa di uno o più fonemi e poter quindi indagare più a fondo l'eventuale presenza di disturbi del linguaggio collegati all'errata pronuncia di tali fonemi. I test per valutare la cosiddetta "fonetica e fonologia in output del paziente" sono molteplici, ma tra i più famosi, efficaci ed utilizzati dai logopedisti, si trovano due test di valutazione dell'articolazione, conosciuti in letteratura logopedica con i nomi di "Test dell'articolazione di Rossi (1999)" e "Test di valutazione dell'articolazione di F. Fanzago (1983)" [19] [20].

Quest'ultimo è un test di valutazione basato sulla denominazione, spontanea o su ripetizione, di 117 parole rappresentate da immagini e raggruppate in 22 tavole. Il test, per il logoterapeuta, consiste nell'esaminare i fonemi consonantici e le vocali in posizione iniziale, intervocalica e all'interno di gruppi consonantici e stabilire quali fonemi sono articolati correttamente e quali invece vengono sostituiti, omessi e distorti durante la pronuncia di ciascuna parola da parte del paziente. Accade spesso infatti, che un bambino sia in grado di pronunciare correttamente un fonema se questo si trova in posizione iniziale, mentre emergano delle difficoltà se il fonema fa parte di un gruppo consonantico. Tutti i fonemi sono raggruppati e divisi nelle varie tavole per *modo* di articolazione: troviamo quindi tavole dedicate alle consonanti occlusive, fricative, affricate, nasali, laterali e vibranti. Inoltre, alla fine, ci sono alcune tavole dedicate alle vocali e alle semiconsonanti. La somministrazione del test consiste nel far denominare le diverse immagini al paziente e, qualora il bambino avesse qualche difficoltà, la denominazione può essere sollecitata con semplici domande da parte del logopedista oppure può avvenire su ripetizione. Come accennato, fra gli errori articolatori di un fonema, si possono rilevare attraverso questo test:

- **sostituzioni**: s'intende la realizzazione di un suono linguistico diverso da quello richiesto sempre appartenente all'alfabeto fonetico italiano;
- **omissioni**: la mancata realizzazione di uno o più fonemi;
- **distorsioni**: s'intende l'alterazione articolatoria di un fonema che risulta non appartenere all'inventario dei fonemi dell'italiano.

Ogni errore che viene rilevato, può essere annotato dal logopedista, mediante segni specifici, all'interno di un'apposita tabella di valutazione dell'articolazione, relativa esclusivamente al paziente esaminato.

Durante la somministrazione del test il logopedista tiene inoltre traccia di tutte le altre alterazioni rilevate osservando l'atteggiamento errato delle labbra, della lingua, dell'apertura del cavo orale, oltre che dalla presenza di smorfie, corrugamenti della fronte, grimace, fuga d'aria nasale, desonorizzazioni, vari tipi di stigmatismo e incoordinazione pneumo-fono-articolatoria nel paziente [21]. Nell'Appendice A è possibile consultare, oltre alla tabella di valutazione, anche le tavole relative al test di articolazione di Fanzago, le quali sono state utilizzate, in questo lavoro di tesi, per testare l'efficacia degli algoritmi implementati e dei metodi utilizzati per identificare e mettere in evidenza automaticamente la presenza di sostituzioni, omissioni ed inserimenti di fonemi rispetto alla pronuncia corretta di ogni parola.

2.3 Differenza fra stringhe: la distanza di Levenshtein

Sono molte le applicazioni, in ambito informatico e non solo, nelle quali è necessario determinare la somiglianza fra due stringhe: applicazioni di elaborazione delle informazioni, come ad esempio il controllo degli errori nelle comunicazioni di dati, la bioinformatica, il riconoscimento vocale e la correzione ortografica sono solo alcune delle aree che necessitano di algoritmi in grado di risolvere efficientemente ed efficacemente questo problema. In bioinformatica, ad esempio, un algoritmo che effettui un confronto fra stringhe che rappresentano diverse sequenze molecolari o di DNA e che sia in grado di misurare la somiglianza fra esse, permetterebbe di evidenziare sottosequenze comuni rilevanti per le ricerche biologiche che si vogliono effettuare. Ancora, nell'ambito dell'elaborazione dei testi e in strumenti di ricerca informatici, quali *grep* di UNIX, i motori di ricerca su internet, le biblioteche digitali, i giornali elettronici, gli elenchi telefonici online e anche le grandi enciclopedie online, algoritmi in grado di effettuare confronti e *matching* fra stringhe o insiemi di parole, consentono una rapida e puntuale ricerca delle informazioni [22]. Inoltre, nello specifico ambito logopedico di nostro concreto interesse, essere in grado di confrontare ed analizzare la trascrizione fonetica corrispondente a quanto pronunciato da un paziente durante un particolare esercizio di una seduta, con la corretta trascrizione fonetica di quanto avrebbe dovuto pronunciare, permetterebbe di mettere in evidenza in maniera rapida ed automatica l'eventuale presenza di errori di pronuncia e conseguenti disturbi del linguaggio.

Una metrica molto utilizzata nella teoria dell'informazione e nella teoria dei

linguaggi per rappresentare la somiglianza, o la differenza, fra due stringhe è la cosiddetta distanza di *Levenshtein*, anche chiamata distanza di *edit*. Venne introdotta nel 1965 dallo scienziato russo *Vladimir Levenshtein* ed è definita come il numero minimo di modifiche elementari che consentono di trasformare una stringa nell'altra. Per modifica elementare si intende la cancellazione, la sostituzione o l'inserimento di un carattere o di una parola, rispettivamente da una stringa o da un insieme di parole [23] [24]. Tuttavia, limitarsi semplicemente a determinare la distanza fra le due stringhe non è sufficiente per i nostri scopi: a partire dalla distanza calcolata, è necessario infatti computare un allineamento preciso fra esse in modo da risalire a quali caratteri specifici sono stati inseriti, sostituiti o cancellati. Solo in questo modo sarà possibile evidenziare ed analizzare gli eventuali difetti di pronuncia del paziente.

Nelle sottosezioni che seguono verrà definita in maniera scientifica e più dettagliata la distanza di *Levenshtein*, verrà presentato un algoritmo efficiente per il calcolo di tale distanza e, successivamente, verrà illustrata una strategia di *backtracking* necessaria per risalire ai caratteri specifici che hanno subito cancellazione, inserimento o sostituzione.

2.3.1 Definizione

Da un punto di vista intuitivo, la distanza fra due stringhe è tanto minore, quanto più queste si assomigliano. Prendiamo come esempio la differenza fra le stringhe "`sitting`" e "`kitten`": la distanza di *Levenshtein* fra queste due è pari a 3, in quanto 3 è il minimo numero di modifiche elementari necessario per trasformare "`sitting`" in "`kitten`". Infatti:

1. `sitting` → `kitting`: sostituzione della `s` con la `k`;
2. `kitting` → `kitteng`: sostituzione della `i` con la `e`;
3. `kitteng` → `kitten`: cancellazione della `g` alla fine.

Matematicamente, la distanza di *Levenshtein* fra due stringhe a e b di lunghezza rispettivamente $|a|$ e $|b|$ è data da $lev_{a,b}(|a|, b|)$, dove

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{se } min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{altrimenti.} \end{cases}$$

$1_{(a_i \neq b_j)}$ è una funzione caratteristica uguale a 0 quando $a_i = b_j$ e uguale a 1 altrimenti, mentre $lev_{a,b}(i, j)$ è la distanza fra i primi i caratteri di a e i primi j caratteri di b. Si noti come il primo caso del minimo identifichi un'operazione di cancellazione, il secondo un'operazione di inserimento, mentre il terzo identifichi sia l'operazione di sostituzione (*mismatch*), nel caso in cui $a_i \neq b_j$, sia il *matching* dei caratteri nel caso in cui $a_i = b_j$. Alle operazioni di cancellazione, inserimento e sostituzione è stato assegnato, in questo specifico caso, un peso pari ad 1, tuttavia, in alcune applicazioni è possibile assegnare alle varie operazioni pesi o funzioni di costo differenti. L'operazione di sostituzione, ad esempio, potrebbe essere vista come un'operazione di cancellazione immediatamente seguita da un'operazione di inserimento, e, sotto questa interpretazione, sarebbe possibile assegnarle un peso pari a 2 [24]. Nel nostro caso specifico però, consideriamo atomica l'operazione elementare di sostituzione in quanto non necessario attribuirne un peso differente.

2.3.2 L'algoritmo di Wagner-Fischer

A partire dalla definizione ricorsiva appena illustrata, la soluzione più immediata, semplice e diretta per calcolare la distanza di Levenshtein fra due stringhe è quella di realizzare un algoritmo ricorsivo che, ricevute in input due stringhe s e t e le loro rispettive lunghezze, ritorna la distanza di *edit* fra le due. Lo pseudocodice di un banale algoritmo che implementa questa soluzione è illustrato di seguito.

RecEditDistance($s, lengthS, t, lengthT$)

```
1  int cost;
2  // Caso base: stringhe vuote
3  if (lengthS == 0) return lengthT;
4  if (lengthT == 0) return lengthS;
5  // Verifica se gli ultimi caratteri delle due stringhe matchano o meno
6  if (s[lengthS − 1] == t[lengthT − 1])
7        cost = 0;
8  else
9        cost = 1;
10 // Chiamate ricorsive
11 return minimum(RecEditDistance(s, lengthS − 1, t, lengthT) + 1,
                  RecEditDistance(s, lengthS, t, lengthT − 1) + 1,
                  RecEditDistance(s, lengthS − 1, t, lengthT − 1) + cost)
```

È evidente che questa implementazione, seppur semplice ed immediata, è estremamente inefficiente in termini di complessità sia di spazio che tem-

porale: l'algoritmo infatti, ricalcola la distanza di *Levenshtein* delle stesse sottostringhe più e più volte rendendo di fatto esponenziale la sua complessità.

Come soluzione a questo problema e con l'obiettivo di rendere più efficiente questo algoritmo, compaiono in letteratura, tra la fine degli anni sessanta e la metà degli anni settanta dello scorso secolo, svariate scoperte indipendenti riportanti tutte il medesimo algoritmo conosciuto e ricordato oggi come l'algoritmo di Wagner-Fischer [25] [26]. Si tratta di un algoritmo che sfrutta i principi della *programmazione dinamica* per effettuare il calcolo della distanza di *edit* fra due stringhe in maniera efficiente [27] [28].

Come noto, la *programmazione dinamica* è una tecnica di progettazione degli algoritmi che si basa sulla suddivisione del problema principale in sottoproblemi e sull'uso delle cosiddette sottostrutture ottimali[2] [29]. Queste tecniche di *programmazione dinamica* vengono applicate ogniqualvolta si riscontra la presenza di problemi sovrapponibili nel contesto del problema che si vuole risolvere, proprio come nel caso dell'algoritmo ricorsivo descritto poc'anzi, dove avviene un ingenuo spreco di tempo per via del fatto che la soluzione ottimale a sottoproblemi già risolti viene continuamente ricalcolata. Per evitare che ciò accada, è necessario quindi salvare in memoria la soluzione dei problemi già risolti, così, se in seguito si necessita di risolvere lo stesso problema o sottoproblema, è possibile riprendere e riutilizzare la soluzione già calcolata. Questo approccio prende il nome di *memoizzazione*. In generale, la *programmazione dinamica* si basa su uno dei due seguenti approcci:

- *Top-down*: combina ricorsività e *memoizzazione* in quanto il problema originale viene spezzato in sottoproblemi, i quali vengono risolti e la loro soluzione viene ricordata nel caso sia necessario risolverli ancora;

- *Bottom-up*: vengono risolti per primi tutti i sottoproblemi che possono essere necessari per costruire successivamente la soluzione a problemi più ampi.

Questo ultimo approccio, nella pratica, si rivela leggermente migliore come dimensione degli stack e numero di chiamate alle funzioni rispetto all'approccio *top-down*, anche se, in generale, spesso non è semplice individuare tutti i sottopreblemi strettamente necessari alla soluzione del problema origina-

[2]In informatica, un problema possiede una sottostruttura ottimale se è possibile costruire efficientemente una soluzione ottimale di esso a partire dalle soluzioni ottimali dei suoi sottoproblemi.

le [30].

L'algoritmo di Wagner-Fischer per il calcolo della distanza di Levenshtein è un esempio di *programmazione dinamica* con approccio *bottom-up*. Infatti, esso si basa sul fatto che se si utilizza una matrice d'appoggio per salvare e tenere traccia delle distanze di *edit* fra tutti i prefissi della prima stringa e tutti i prefissi della seconda stringa, allora, anziché ricalcolare ogni volta le stesse distanze, si possono calcolare le distanze fra prefissi sempre più lunghi a partire dalle distanze già calcolate e disponibili all'interno della matrice, così da ottenere la distanza fra le due intere stringhe come ultimo valore calcolato. In altre parole, la suddetta matrice tiene traccia delle soluzioni di tutti i sottoproblemi, la cui risoluzione è necessaria per arrivare alla soluzione del problema generale. Lo pseudocodice che segue mostra un'implementazione di questo algoritmo, il quale, ricevute come input due stringhe s e t, rispettivamente di lunghezza m ed n, ritorna la distanza di *Levenshtein* fra le due.

EditDistance($s[1..m], t[1..n]$)

```
 1  Sia d un array bidimensionale di interi di dimensioni [0..m, 0..n]
 2  // Inizializzazione
 3  for i in [0..m]
 4        d[i, 0] = i
 5  for j in [0, n]
 6        d[0, j] = j
 7  // Relazione di ricorrenza
 8  for j in [1..n]
 9        for i in [1..m]
10              if s[i − 1] == t[j − 1]
11                    d[i, j] = d[i − 1, j − 1] // Nessuna operazione richiesta
12              else
13                    d[i, j] = minimum(d[i − 1, j] + 1, // una cancellazione
14                                      d[i, j − 1] + 1, // un inserimento
15                                      d[i − 1, j − 1] + 1) // una sostituzione
16  return d[m, n]
```

Dalla prima parte dell'algoritmo (righe 2-6) si evince che la distanza di una qualunque stringa non vuota da una stringa vuota è pari alla lunghezza della stringa non vuota. Infatti le operazioni elementari necessarie per trasformare l'una nell'altra consistono in una serie di cancellazioni o di inserimenti (a seconda del punto di vista), in numero pari alla lunghezza della stringa non vuota. La seconda parte dell'algoritmo (righe 7-15) invece, implementa la

relazione di ricorrenza che deriva dalla definizione di distanza di *Levenshtein* e si assume, anche in questo caso, che la sostituzione abbia un costo pari ad 1. L'invariante che viene mantenuta ed è valida durante tutta l'applicazione dell'algoritmo consiste nel fatto che è sempre possibile trasformare un qualunque prefisso di s di lunghezza i ($s[1..i]$), in un prefisso di t di lunghezza j ($t[1..j]$), usando un minimo di $d[i, j]$ operazioni elementari. Alla fine quindi, l'ultimo elemento della matrice che viene calcolato (quello in basso a destra), rappresenta il valore della distanza fra le due stringhe intere, che è ciò che volevamo calcolare.

Per capire meglio come avviene la costruzione della matrice delle distanze, o *distance matrix*, e vedere quindi l'algoritmo in azione, riprendiamo in esame le due stringhe `"kitten"` e `"sitting"`, di cui già avevamo calcolato la distanza applicando la definizione e che già sappiamo essere pari a 3. La *distance matrix* che risulta applicando l'algoritmo a tali stringhe è la seguente:

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	**3**

Tabella 2.3: *Distance matrix* fra due stringhe.

Nella matrice sono presenti le distanze fra tutte le possibili combinazioni di prefissi delle due stringhe e, l'ultimo elemento in basso basso a destra, come ci aspettiamo, rappresenta la distanza fra le due intere stringhe.

Ricordiamo che l'obiettivo finale che si intende raggiungere nel contesto logopedico dell'applicazione che dev'essere messa a punto, è quello di mettere in evidenza, date due trascrizioni fonetiche, di cui una corretta e l'altra corrispondente a quanto pronunciato da un paziente, quali specifici fonemi sono stati pronunciati correttamente e quali invece sono stati distorti, omessi, so-

stituiti o erroneamente inseriti.

L'algoritmo efficiente appena descritto fornisce, oltre alla distanza di *edit* fra le due stringhe, anche la *distance matrix* che è stata elaborata durante la computazione per facilitarne il calcolo. Mentre il valore della distanza di *Levenshtein* fra le due stringhe non ci consente di risalire direttamente ai caratteri in corrispondenza dei quali si è verificata una sostituzione piuttosto che un inserimento o un'omissione, un'analisi della *distance matrix* consentirebbe di raggiungere tale scopo.

2.3.3 Backtracking

Per poter risalire ai caratteri (o ai fonemi) interessati dalle operazioni di cancellazione, inserimento e sostituzione è necessario eseguire un allineamento fra i caratteri delle due stringhe. Per riuscire a calcolare l'allineamento *ottimo* è necessario, ogniqualvolta si calcola il valore di una cella della *distance matrix*, marcare tale cella con una freccia la cui direzione dipende dall'operazione elementare riscontrata dall'algoritmo (sostituzione, inserimento, cancellazione, matching) e, una volta raggiunta l'ultima cella, attraverso un'operazione di *backtracking* calcolare a ritroso il cammino ottimo a partire da quest'ultimo elemento [24] [28]. Per comprendere meglio questa tecnica, prendiamo come riferimento la *distance matrix* relativa alle stringhe `"kitten"` e `"sitting"`. Se ad ogni cella in cui riscontriamo un inserimento associamo una freccia che punta verso sinistra ($\leftarrow$), ad ogni cella in cui riscontriamo una cancellazione una freccia che punta verso l'alto ($\uparrow$) e ad ogni cella in cui riscontriamo una sostituzione o un matching una freccia che punta in alto a sinistra ($\nwarrow$), otteniamo la matrice raffigurata in Tabella 2.4.

È evidente che ogni cammino non crescente che, a partire dall'elemento iniziale della *distance matrix* d ($d[0,0]$), arriva all'elemento contenete la distanza ($d[m,n]$), rappresenta un possibile allineamento delle due stringhe. L'allineamento ottimo è quello costituito da sottoallineamenti ottimi. Per computare i sottoallineamenti ottimi e di conseguenza l'allineamento ottimo fra le due stringhe è necessario spostarsi, a partire dall'elemento contente la distanza fra le due stringhe, quindi dall'elemento in basso a destra della matrice, nella direzione indicata dalla freccia, e, qualora più di una freccia fosse presente nella stessa cella, spostandosi verso la cella il cui elemento ha valore minore. In questo modo, poiché si è proceduto a ritroso, si è in realtà computato l'allineamento ottimo dell'inverso delle due stringhe: basterà pertanto rovesciare l'allineamento calcolato per ottenere il reale allineamento ottimo. Nella Tabella 2.4 è stato messo in evidenza l'allineamento ottimo fra le stringhe

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	↖ 1	← 2	← 3	← 4	← 5	← 6
i	2	↑ 2	↖ 1	← 2	← 3	← 4	← 5
t	3	↑ 3	↑ 2	↖ 1	↖ 2	← 3	← 4
t	4	↑ 4	↑ 3	↖ ↑ 2	↖ 1	← 2	← 3
i	5	↑ 5	↖ ↑ 4	↑ 3	↑ 2	↖ 2	← 3
n	6	↑ 6	↑ 5	↑ 4	↑ 3	↑ 3	↖ 2
g	7	↑ 7	↑ 6	↑ 5	↑ 4	↑ 4	↑ **3**

Tabella 2.4: *Distance matrix* fra due stringhe con *backtracking*.

`"sitting"` e `"kitten"`, cioè quell'allineamento che consente di trasformare una stringa nell'altra attraverso il minor numero di operazioni elementari (tre in questo caso). Attraverso questo procedimento è quindi possibile risalire facilmente ai caratteri (o fonemi) che sono stati interessati dalle operazioni elementari di sostituzione, cancellazione, inserimento e matching: osservando infatti la direzione delle frecce presenti sul cammino ottimo, sono state messe in evidenza, col colore rosso, le operazioni di sostituzione, col colore grigio le operazioni di cancellazione e, in verde, le operazioni di matching. In altre parole, i passaggi appena descritti permettono di computare l'allineamento seguente.

s	**i**	**t**	**t**	**i**	**n**	**g**
k	**i**	**t**	**t**	**e**	**n**	*
sost.	matching	matching	matching	sost.	matching	canc.

Tabella 2.5: Il risultato dell'allineamento grazie al *backtracking*.

Quindi, se supponiamo che `"sitting"` sia la stringa corretta di riferimento e `"kitten"` invece la stringa da processare e analizzare, l'output di tutta questa procedura risulta essere `kittenn`.

Grazie all'algoritmo di Wagner-Fischer è stato possibile calcolare, oltre alla distanza di *edit* minima anche la *distance matrix* che è di fatto l'input grazie al quale le operazioni di *backtracking* permettono di risalire alle posizioni,

all'interno delle stringhe da analizzare, nelle quali si verificano le operazioni di matching, sostituzione, inserimento e cancellazione.

Da un punto di vista della complessità computazionale l'algoritmo di Wagner-Fischer così implementato ha una complessità sia in termini di tempo che in termini di spazio pari a $O(nm)$, dove n ed m sono le lunghezze delle stringhe da processare, mentre le operazioni di *backtracking* hanno una complessità pari a $O(n + m)$ [28]. Si noti come la complessità temporale e spaziale dell'algoritmo di Wagner-Fischer sia notevolmente migliorata rispetto alla complessità esponenziale dell'algoritmo ricorsivo: grazie alla *programmazione dinamica* infatti il calcolo di ogni riga della *distance matrix* dipende esclusivamente dalla riga precedente, già calcolata e *memoizzata*, e non è necessario risolvere di volta in volta sottoproblemi che già erano stati risolti.

2.4 I software

2.4.1 Kaldi

Kaldi [31] è un set di strumenti open-source per il riconoscimento vocale, scritto in *C++* e rilasciato sotto licenza *Apache v.2.0*, il cui sviluppo è cominciato nel 2009 durante un workshop dal titolo "*Low Development Cost, High Quality Speech Recognition for New Languages and Domains*" tenutosi presso la *Johns Hopkins University* nel Maryland (USA). Gli strumenti che mette a disposizione *Kaldi* sono molti e comprendono, fra i principali, modelli per l'estrazione delle *features* dai segnali audio, modelli acustici per l'*Automatic Speech Recognition* (ASR) in grado di effettuare riconoscimento fonetico attraverso l'utilizzo di alberi di decisione e modelli di Markow nascosti, strumenti per la modellazione del linguaggio e strumenti per la codifica e decodifica dei segnali audio.

Nell'ambito dello sviluppo dell'applicazione *Prime Frasi* del progetto LogoKit, *Kaldi* è stato utilizzato dal collega *Nicola Rigato*, per mettere a punto ed implementare l'accesso ad un modello di classificazione fonetico che fosse in grado di riconoscere e classificare i fonemi a partire da un segnale audio registrato dal dispositivo mobile [5]. Ai fini di questo lavoro di tesi si è reso necessario uno studio di questo modello di classificazione al fine di riconoscere e comprendere tutti i suoi possibili output, compreso lo standard con cui i fonemi sono codificati; inoltre, è stato approfondito il meccanismo di *scoring* attraverso il quale *Kaldi* effettua la stima della bontà del classificatore: l'im-

portanza di questo meccanismo è cruciale per l'analisi dei risultati dei test sperimentali, i quali verranno trattati ed illustrati nel Capitolo 4.

2.4.2 Android Studio

Annunciato ufficialmente nel corso della conferenza *Google I/O* tenutasi nel maggio del 2013, *Android Studio* è un ambiente di sviluppo integrato (IDE) esclusivamente dedicato allo sviluppo nativo di applicazioni per la piattaforma Android. La prima versione stabile, disponibile per i sistemi operativi *Windows*, *Mac OS X* e *Linux*, fu rilasciata nel dicembre del 2014 e, nello stesso periodo, *Google* annunciò ufficialmente che avrebbe discontinuato il supporto degli *Android Development Tools (ADT)* di *Eclipse*. *Android Studio* è basato sul software della *JetBrains* chiamato *IntelliJ IDEA* [32].

L'applicazione *Prime Frasi* del progetto *LogoKit*, ospitata su un repository privato di *Bitbucket* è stata sincronizzata con *Android Studio* attraverso il noto sistema di controllo versione per software chiamato *Subversion*, anche noto come *svn*. In tal modo, contemporaneamente al lavoro di alcuni colleghi [2] [4] [5], è stato possibile arricchire *Prime Frasi* di funzionalità che verranno descritte nel dettaglio nel Capitolo 3. Sempre in *Android Studio* è stata messa a punto un'applicazione di test che ha permesso di testare, non solo l'efficacia della conversione fra gli standard fonetici SAMPA ed IPA, ma anche di effettuare un'analisi circa i fonemi riconosciuti con maggior precisione grazie al modello di classificazione di *Kaldi* e agli algoritmi esposti nelle sottosezioni 2.3.2 e 2.3.3 del Capitolo 2. Tale applicazione sarà esaustivamente descritta nel Capitolo 4.

2.4.3 Audacity

Audacity è un software multipiattaforma, distribuito sotto *GNU General Public License* (GPL), che funge principalmente da editor di file audio. È interamente scritto in linguaggio *C++* e come funzionalità base permette la registrazione, la riproduzione, la modifica ed il mixaggio di file audio codificati con svariati formati, tra i quali *WAV* [33].

In questo lavoro di tesi, il software è stato impiegato per elaborare ed uniformare registrazioni audio di parlato provenienti da sorgenti differenti, così da poter essere poi utilizzate per compiere i test sperimentali che saranno illustrati nel Capitolo 4. Il software ha anche consentito, in alcuni casi, di

esaminare lo spettrogramma[3] dei file audio, così da studiare le caratteristiche più salienti di ciascuna registrazione, come ad esempio la presenza di rumore alle varie frequenze al variare della sorgente di registrazione.

[3]Lo spettrogramma è la rappresentazione grafica dell'intensità di un suono in funzione del tempo e della frequenza.

Capitolo 3

Lavoro svolto ed implementazione

3.1 Tentativi effettuati e stato dell'arte

Dal momento che l'obiettivo ultimo è quello di analizzare una trascrizione fonetica della voce registrata da un dispositivo mobile al fine di rilevare la presenza di eventuali difetti di pronuncia, il primo passo è stato quello di ricercare librerie e servizi, sia proprietari che open-source, che fornissero API[1] utili al raggiungimento di tale scopo. Poiché *Prime Frasi* è un'applicazione sviluppata per il sistema operativo Android, la ricerca e lo studio sono stati rivolti principalmente alle API ufficiali per lo *Speech Recognition* e lo *Speech-to-Text* messe a disposizione dalla piattaforma Android, ma sono stati presi in considerazione anche altri servizi e strumenti che eventualmente avrebbero potuto interfacciarsi con l'applicazione.

Visto che la maggior parte dei dispositivi mobili (e non solo) mette a disposizione sofisticati riconoscitori vocali e assistenti vocali all'interno del proprio sistema operativo, negli ultimi anni le più grandi società, come *Google* e perfino *Apple*, hanno messo a disposizione di tutti gli sviluppatori apposite API affinché le funzionalità di riconoscimento vocale e *Speech-to-Text* venissero integrate all'interno di tutte le applicazioni che ne avessero necessità. Lo *Speech-to-Text* è il processo attraverso il quale il linguaggio orale umano viene riconosciuto e successivamente tradotto in testo attraverso un apposito sistema di riconoscimento ed elaborazione della voce. Come già accennato nell'introduzione di questa tesi quando si è descritto il contesto operativo nel quale si va ad agire, le funzionalità di *Speech Recognition* e *Speech-to-Text*

[1] API è l'acronimo di *Application Programming Interface* e rappresenta un insieme di procedure o librerie software che vengono messe a disposizione del programmatore affinché rendano possibile assolvere a determinati compiti all'interno di un determinato programma.

non si limitano a rilevare e riportare una trascrizione precisa e fedele di quanto pronunciato, ma l'audio e la voce registrati vengono elaborati e processati al fine di correggere ed interpretare il più possibile quanto l'utente volesse dire. Questa capacità è un grande punto di forza per tutti gli assistenti vocali e fa si che essi risultino essere sempre più "intelligenti", ma per gli scopi e gli obiettivi che ci siamo prefissati di raggiungere nell'applicazione *Prime Frasi* a supporto dei logopedisti, questo non va affatto bene. Se qualche libreria o servizio "meno sofisticati" dei moderni riconoscitori vocali fossero in grado di fornire una trascrizione "grezza" e fedele di quanto pronunciato dall'utente, saremmo già ad un ottimo punto di partenza per raggiungere gli scopi prefissati.

Di seguito viene presentata una panoramica di librerie e di servizi, fra i più famosi e diffusi, che offrono la possibilità di effettuare riconoscimento vocale e trascrizione di quanto pronunciato. In questo modo sarà possibile, non solo escludere a priori alcune possibili strade che non porterebbero al raggiungimento del nostro scopo, ma anche offrire una visione d'insieme di quello che è lo stato dell'arte nel contesto operativo in cui andiamo ad operare.

3.1.1 Le librerie sperimentate

Prima di procedere con l'elencare le librerie che sono state sperimentate, è bene ricordare quali sono i requisiti strettamente necessari e desiderabili affinché il servizio possa essere integrato all'interno di *Prime Frasi* e sia d'aiuto al raggiungimento del nostro scopo. Per prima cosa, è necessario che la libreria o il servizio in questione siano in grado di riconoscere e trascrivere la lingua italiana; è inoltre necessario che supportino l'elaborazione di file audio e non siano soltanto in grado di elaborare in real time lo stream audio proveniente dal microfono durante la registrazione della voce; l'ideale sarebbe inoltre che permettessero un funzionamento (anche) offline e che non fossero servizi a pagamento o in abbonamento, bensì open-source.

Dopo alcune ricerche, le librerie sulle quali si è deciso di indagare sono state:

- *IBM Watson*: questa società offre molti servizi che, sfruttando l'intelligenza artificiale, sono in grado di assolvere a compiti e task che vanno dall'analisi del linguaggio naturale al riconoscimento e all'elaborazione di immagini e perfino all'analisi dei dati [34]. Fra i vari servizi offerti vi è anche lo *Speech-to-Text* di cui abbiamo bisogno, il quale supporta l'input e l'elaborazione di file di tutti i formati audio più diffusi, offre un servizio base gratuito e mette a disposizione API che consentireb-

bero di interfacciare facilmente il servizio con l'applicazione, tuttavia, oltre a non funzionare offline, non supporta il riconoscimento della lingua italiana. Per questo motivo l'impiego di questo servizio è stato scartato.

- *Wit.ai*: questo servizio possiede tutte le caratteristiche di *IBM Watson* ed inoltre è gratuito e supporta l'italiano [35]. Tuttavia, come ampiamente spiegato nella documentazione presente sul sito web, l'obiettivo principale del riconoscimento vocale è quello di interpretare quanto pronunciato dall'utente restituendo come output i risultati di una vera e propria analisi semantica delle frasi pronunciate. A causa di ciò, non si è chiaramente in grado di ottenere una trascrizione "grezza" del parlato e quindi, anche l'utilizzo di questo servizio è stato scartato.

Con funzionamento analogo a quest'ultimo servizio citiamo le API messe a disposizione da *Amazon* che consentono di usufruire delle potenzialità del neonato assistente vocale *Alexa* [36], come anche le API messe a disposizione da *Microsoft Cognitive Services* [37] che restituiscono il risultato di un'analisi semantica delle frasi pronunciate, poco utile per i nostri scopi.

Dal momento che *Prime Frasi* è un'applicazione pensata per la piattaforma *Android*, particolare attenzione è stata dedicata non solo ai servizi di *Speech recognition* e *Speech-to-Text* offerti da *Google*, ma anche alle API native per la piattaforma Android. Di seguito verranno illustrate le loro principali caratteristiche e alcuni test che sono stati effettuati per verificare se le trascrizioni del parlato restituite sono "grezze" oppure no.

- *Google: Cloud Speech API*: questo servizio mette a disposizione API in grado di effettuare *Speech recognition* e *Speech-to-Text* oltre che dallo stream audio in input dal microfono del dispositivo, anche a partire da file audio e supporta il riconoscimento di oltre ottanta lingue, fra cui l'italiano, tuttavia, è attualmente in versione beta e la sua documentazione risulta essere piuttosto scarna ed incompleta; inoltre, il servizio è a pagamento [38]. Per questi motivi si è scelto di non proseguire la sperimentazione nemmeno con questo servizio, riservandosi però la possibilità di testarlo in futuro, quando sarà rilasciata una prima release stabile delle sue API.

- *Android API*: la piattaforma Android mette a disposizione la classe `SpeechRecognizer` attraverso la quale è possibile effettuare operazioni di *Speech-to-Text* a partire dallo stream audio in input sul microfono del dispositivo [39]. Il servizio permette il riconoscimento della lingua

italiana, funziona anche offline ed è completamente gratuito, l'unica pecca è che non supporta il riconoscimento e l'elaborazione di file audio come input. Tuttavia, dal momento che le API in questione sono native per la piattaforma di nostro interesse, nell'ambito di un progetto per il corso di *Informatica musicale* tenuto dal professor *Giovanni De Poli*, sono stati effettuati, insieme ad alcuni colleghi, alcuni test con lo scopo di verificare quanto le trascrizioni restituite dai server di *Google* fossero fedeli a quanto pronunciato. Per prima cosa è stato messo a punto un dataset contenente registrazioni di frasi da noi pronunciate. Le frasi all'interno del dataset sono state suddivise per grado di difficoltà in base alla loro lunghezza, alla velocità di lettura, alla presenza di fonemi particolari ed ostici da riconoscere ed in base alla difficoltà che si ha nel pronunciarle, come ad esempio quando si tratta di scioglilingua. Nel dataset sono state volutamente inserite registrazioni contenenti evidenti errori di pronuncia, in modo tale da capire come si comporta il sistema di riconoscimento ed osservare se la loro trascrizione risulta errata oppure viene internamente analizzata e corretta dagli algoritmi di *Speech-to-Text* di *Google*. Contemporaneamente alla realizzazione del dataset, è stata messa a punto un'applicazione di test, la quale, implementando la classe `SpeechRecognizer`, aveva il compito di inviare l'audio delle registrazioni ai server di *Google* e visualizzare la trascrizione di risposta sullo schermo. Poiché con queste API è supportato soltanto l'input dell'audio dal microfono del dispositivo, le registrazioni del dataset sono state riprodotte una ad una mentre il dispositivo era in ascolto. Uno scopo secondario di questo esperimento era anche quello di capire quanto la qualità del riconoscimento dipendesse dalla qualità, dalle caratteristiche e dalla rumorosità dell'audio registrato. I risultati di questo test hanno provato e confermato le ipotesi che già avevamo: il servizio di *Speech-to-Text* di *Google* ha riconosciuto correttamente la quasi totalità delle quarantasei frasi che costituiscono il dataset per ognuna delle quattro differenti voci utilizzate. Il riconoscimento è avvenuto correttamente nonostante l'audio avesse subito due differenti compressioni, la prima in fase di registrazione e la seconda prima di essere inviato dall'applicazione al server per il riconoscimento. Inoltre, anche le frasi contenenti evidenti errori di pronuncia, sono state tutte riconosciute e trascritte correttamente, segno del fatto che il servizio offerto da *Google*, basandosi su un ampio e sempre aggiornato modello del linguaggio, è in grado di correggere le parole pronunciate in maniera errata fornendo, come risposta all'utente, la loro trascrizione corretta.

Lo studio e le ricerche effettuate sulle librerie messe a disposizione da servizi

di terze parti e i test effettuati sulle librerie native della piattaforma Android, hanno messo in luce l'impossibilità di utilizzare questi servizi per il riconoscimento e la trascrizione fedele del parlato necessaria al raggiungimento del nostro scopo. Tutti questi servizi infatti, si basano su veri e propri modelli del linguaggio, i quali sono in grado, attraverso l'utilizzo di modelli di Markow nascosti, di predire correttamente le parole pronunciate, anche se esse, foneticamente parlando, non corrispondono a quanto realmente pronunciato dall'utente. A partire da questo risultato, inoltre, la gran parte dei servizi analizzati opera sulle trascrizioni ottenute un'analisi semantica mediante tecniche di *information extraction*, fornendo come output utili informazioni circa un determinato contesto o situazione. Ciò di cui però si ha bisogno, ai fini dell'applicazione che si vuole migliorare e nell'ambito logopedico in cui si va ad operare, è di un modello fonetico del linguaggio e di uno strumento che, attraverso tale modello, sia in grado di classificare accuratamente, a partire dalla voce, i *foni* della lingua italiana, così da poter compiere successivamente un'analisi accurata della trascrizione fonetica ottenuta.

Per ottenere una trascrizione fonetica accurata, la scelta è ricaduta sul toolkit open source *Kaldi*, di cui già si è parlato nella Sezione 2.4.1 del Capitolo 2. Questo toolkit, sfruttando un modello di classificazione fonetico della voce del bambino, allenato e realizzato da un gruppo di ricerca del *CNR* di Padova[2] e messo gentilmente a disposizione per la realizzazione di questo progetto dal Dottor *Piero Cosi*, ha permesso di ottenere, all'interno dell'applicazione *Prime Frasi*, una trascrizione fonetica fedele e precisa di quanto pronunciato dall'utente e pronta per essere analizzata.

Per un maggiore approfondimento circa il funzionamento del toolkit *Kaldi* e la sua integrazione all'interno della piattaforma Android si rimanda al lavoro di tesi svolto dal collega *Nicola Rigato* [5]; in questa sede, come già accennato, ci limitiamo ad analizzare l'output del classificatore in funzione dell'analisi fonetica che si andrà a svolgere. L'analisi dell'output del modello di classificazione dei fonemi si rende necessaria non solo per capire la qualità delle trascrizioni fonetiche che sarà possibile ottenere, ma anche per il fatto che il modello di classificazione è stato fornito privo di documentazione, ed è quindi doveroso essere consapevoli di tutti i possibili output in modo da poterli gestire al meglio all'interno di *Prime Frasi*.

[2]Consiglio Nazionale delle Ricerche (CNR) - Area della Ricerca di Padova: `http://www.pd.cnr.it/index.php/it/`

3.2 Implementazione in Android

3.2.1 L'output del classificatore fonetico

Il modello di classificazione utilizzato da *Kaldi* per il riconoscimento e la classificazione dei fonemi della lingua italiana a partire dalla voce del soggetto parlante, prevede, per ciascun *fono*, 34 possibili stringhe di output, di cui, 28 corrispondono alle reali trascrizioni dei fonemi, e 6 rappresentano stringhe di controllo e marcatori vari. In altre parole, il modello di classificazione fonetica è in grado di riconoscere 28 dei 30 fonemi previsti dalla lingua italiana, infatti, è in grado di riconoscere tutti i 23 fonemi consonantici e semiconsonantici, ma rileva soltanto 5 delle 7 vocali previste dall'alfabeto fonetico italiano. Il modello non distingue i fonemi delle vocali semichiuse `/e/` ed `/o/` dalle loro rispettive versioni semiaperte `/ɛ/` ed `/ɔ/`, fornendo in output semplicemente i fonemi corrispondenti alle versioni semichiuse di queste due vocali, indipendentemente da come esse sono state pronunciate. Alla luce di questa prima analisi quindi, risulteranno identiche le trascrizioni fonetiche di parole come `[peska]` (sostantivo e voce del verbo pescare) e `[pɛska]` (frutto), oppure `[botte]` (barile) e `[bɔtte]` (percosse). Gli altri 6 possibili output sono rappresentati dalle stringhe seguenti: `<s>` e `</s>` sono dei marcatori rispettivamente di inizio e fine frase, `<#0>` rappresenta un riconoscimento fonetico ambiguo, `<eps>` corrisponde ad un carattere speciale, `@sch` corrisponde ad un suono o rumore non identificabile da alcun fonema, come ad esempio un colpo di tosse, uno starnuto, un battito di mani, il rumore provocato da un oggetto che cade a terra, ecc., ed infine `sil` rappresenta un silenzio. Durante tutti i test effettuati, il classificatore non ha mai restituito come output le stringhe `<s>`, `</s>`, `#0` ed `<eps>`, tuttavia, dal momento che *Kaldi* costruisce alberi decisionali durante il processo di riconoscimento e classificazione, la presenza di questi simboli è necessaria, in quanto essi rappresentano transizioni e stati "speciali" all'interno dei suddetti alberi di decisione.

Da queste considerazioni sulla natura dell'output del modello di classificazione dei fonemi si capisce che non sarà possibile ottenere trascrizioni fonetiche *strette* di quanto pronunciato dall'utente, in quanto il classificatore, non compiendo analisi sull'intonazione, sulla cadenza e sull'inflessione della voce in input, non è in grado di arricchire la trascrizione fonetica con segni diacritici di alcun genere. Per come è stata integrata la classificazione dei fonemi attraverso *Kaldi* all'interno di *Prime Frasi*, la trascrizione fonetica di una determinata registrazione audio che viene restituita dal classificatore, consiste in una stringa nella quale ciascun fonema o marcatore riconosciuto è

separato dall'altro da uno spazio bianco. Inoltre, i fonemi che costituiscono la trascrizione fonetica sono codificati secondo lo standard SAMPA, di cui si è già discusso nella Sezione 2.1.2 del Capitolo 2. La mancanza di una documentazione a corredo del classificatore fonetico, ha reso arduo, almeno in un primo momento, riuscire ad identificare quale fosse lo standard in cui i fonemi venivano trascritti e codificati. Ricordiamo però che lo standard di codifica dei fonemi più diffuso, nonché quello utilizzato dai logopedisti, è lo standard IPA. Per questo motivo, al fine di rendere familiare ai logopedisti la trascrizione fonetica restituita dal classificatore si è reso necessario realizzare un algoritmo che fungesse da convertitore tra gli standard di codifica SAMPA ed IPA, cosicché la trascrizione fonetica potesse essere visualizzata sull'interfaccia di *Prime Frasi* nel modo più consueto.

3.2.2 Convertitore fonetico: da SAMPA ad IPA

Per effettuare la conversione dalla codifica SAMPA alla codifica IPA è stato messo a punto un algoritmo che, ricevuta in ingresso la stringa corrispondente alla trascrizione fonetica restituita dal modello di classificazione, fornisce in output la medesima stringa nella quale tutti i fonemi sono codificati secondo lo standard IPA e non sono più separati da uno spazio bianco. Nello specifico, l'algoritmo, prima individua e separa tutti i fonemi in formato SAMPA che costituiscono la stringa in input e, successivamente, servendosi di una `HashMap` appositamente preparata, li converte uno ad uno nel formato IPA. L'`HashMap` in questione non è altro che un dizionario contenente le coppie chiave-valore formate dai fonemi in formato SAMPA e le rispettive traduzioni in formato IPA. L'implementazione di questo algoritmo è visibile nella *utility class* `PhoneticConverter` che è stata integrata in *Prime Frasi*. Nella Tabella 3.1 sono riportati alcuni fonemi scritti utilizzando lo standard SAMPA con le rispettive traduzioni in IPA. Per i fonemi che non compaiono nella tabella la codifica SAMPA e la codifica IPA coincidono ed il loro simbolo è consultabile nelle tabelle IPA dei fonemi consonantici e vocalici della lingua italiana, illustrate e spiegate nella Sezione 2.1.1 del Capitolo 2.

I marcatori `@sch` e `sil` rimangono inalterati durante la conversione così da segnalare nella trascrizione fonetica l'eventuale presenza di silenzi, rumori indesiderati, o distorsioni eventualmente prodotti dal paziente e sintomo di qualche problema di articolazione fonatoria o disturbo del linguaggio. In particolare, nel caso compaia nella trascrizione il simbolo `@sch`, sta al logopedista interpretare, in base al contesto, la natura e la sorgente di tale rumore. I simboli `<s>`, `</s>`, `#0` e `<eps>` vengono convertiti in uno spazio bianco, poiché, non solo sono poco rilevanti ai fini della trascrizione fonetica,

SAMPA	**IPA**
J	ɲ
L	ʎ
S	ʃ
dZ	dʒ
tS	tʃ

Tabella 3.1: Corrispondenza fra fonemi in codifica SAMPA ed IPA

ma inoltre, come già accennato, durante i test effettuati non sono mai stati restituiti in output dal classificatore.

Affinché tutti i caratteri della stringa codificata in IPA venissero visualizzati correttamente sulla *user interface* di *Prime Frasi*, è stato utilizzato un particolare tipo di font scelto fra quelli espressamente consigliati dall'Associazione Fonetica Internazionale. Il font in questione si chiama *DejaVuSans*, ma si sarebbero potuti impiegare equivalentemente font come *Doulos SIL*, *IPA Kiel* o diversi altri. Per i pochi fonemi da visualizzare nel contesto di questa applicazione si sarebbe potuto utilizzare con tranquillità un qualunque font di sistema messo a disposizione da Android, ma, qualora in futuro le trascrizioni fonetiche venissero arricchite di nuovi simboli e segni diacritici, o qualora il classificatore venisse esteso al riconoscimento fonetico di nuove lingue, la scelta più prudente e consigliata rimane quella di utilizzare font sempre aggiornati che garantiscano una corretta visualizzazione di tutti i fonemi.

Per fornire un esempio del funzionamento del convertitore fonetico che è stato messo a punto, supponiamo che un paziente debba pronunciare la frase "spinge gli gnomi". Se il modello di classificazione non commette errori e il paziente non presenta problemi di pronuncia o particolari disturbi del linguaggio, il classificatore dovrebbe restituire la stringa `"sil s p i n dZ e L i J o m i sil"`. L'algoritmo che effettua la conversione ne riconosce dapprima i fonemi e simboli che la compongono inserendoli in un *array*, e, successivamente, effettua la traduzione di ciascun fonema o simbolo restituendo in output e visualizzando sullo schermo la stringa `"silspindʒeʎiɲomisil"`.

A questo punto, poiché l'obiettivo finale da raggiungere è quello di mettere in evidenza la presenza di errori, omissioni e inserimenti nella trascrizione

fonetica corrispondente a quanto pronunciato dal paziente, in modo da consentire al logopedista di riscontrare la presenza di difetti di pronuncia e poter quindi intervenire di conseguenza, si è reso necessario implementare un algoritmo efficiente che effettuasse un controllo degli errori e mettesse in evidenza i risultati di questa analisi.

3.2.3 Il controllo automatico degli errori di pronuncia

Per poter effettuare un controllo circa la presenza di fonemi distorti, sostituiti o omessi sulla trascrizione fonetica restituita dal classificatore, è necessario avere a disposizione una trascrizione fonetica corretta di quanto il paziente dovrebbe pronunciare in quel preciso esercizio previsto durante la seduta, così da poter effettuare un confronto fra le due trascrizioni fonetiche. Il primo passo dunque, è stato quello di integrare il database di *Prime Frasi* con le trascrizioni fonetiche corrette, in formato SAMPA, di tutte le frasi e le parole disponibili per personalizzare gli esercizi delle sedute dei vari pazienti. Nell'effettuare tali trascrizioni si è tenuto conto delle principali regole e convenzioni per la trascrizione fonetica utilizzate dai logopedisti e dai linguisti italiani, con la speranza che il modello di classificazione fonetico realizzato attraverso *Kaldi* tenesse conto delle stesse convenzioni e fosse stato allenato prendendo in considerazione queste regole. La maggior parte delle convenzioni riguardano principalmente l'utilizzo ed il posizionamento di particolari segni diacritici e di accenti all'interno delle varie trascrizioni fonetiche ma, poiché il classificatore fornisce trascrizioni fonetiche larghe, almeno per il momento, queste regole non sono interessanti nel nostro ambito di applicazione; esistono anche convenzioni riguardanti l'utilizzo di alcuni fonemi: in particolare, nell'italiano standard, i fonemi consonantici `/ʃ/`, `/ts/`, `/dz/`, `/ɲ/` e `/ʎ/` sono sempre lunghi (o geminati), e quindi vanno raddoppiati quando compaiono in posizione intervocalica o tra una vocale e una sonorante; il fonema consonantico `/z/` e i fonemi semiconsonantici approssimanti `/w/` e `/j/` sono invece sempre brevi, in tutti i contesti, e quindi non vanno mai raddoppiati. Un esempio dell'applicazione di questa convenzione è visibile in parole come "maglione", la cui corretta trascrizione fonetica è `[maʎʎone]`, oppure "azione", la cui trascrizione diventa `[attsjone]`. In quest'ultimo caso si noti che non è l'intero fonema `/ts/` ad essere raddoppiato, ma soltanto la prima delle due consonanti che lo costituiscono, ovvero la `t`. Considerazione analoga vale anche per il fonema `/dz/`, che in parole come "azoto", la cui trascrizione fonetica è `[addzɔto]`, raddoppia soltanto la `d`. L'applicazione di tali convenzioni si riscontra anche nelle parole che costituiscono le tavole del test di articolazione di *Fanzago* di cui si è parlato nella Sezione 2.2 del

Capitolo 2 e che sono consultabili nell'Appendice A di questa tesi.

Una volta aggiornato il database di *Prime Frasi* con le trascrizioni fonetiche corrette di ciascuna frase presente al suo interno, è stato possibile far visualizzare sull'interfaccia, lato logopedista, la trascrizione fonetica della frase corrente ancor prima che essa venga pronunciata dal paziente, così da fornire al logoterapeuta una panoramica dei fonemi che dovrebbero essere pronunciati. Dopo che il paziente ha pronunciato la frase in questione, si rende necessario effettuare un confronto fra la trascrizione fonetica corretta, che già è visualizzata, e la trascrizione fonetica riconosciuta dal modello di classificazione, così da controllare l'eventuale presenza di errori di pronuncia e poter quindi metterli in evidenza. Per effettuare tale confronto si è ricorsi all'algoritmo di *Wagner-Fischer* integrato con il meccanismo di *backtracking*, il cui funzionamento è stato ampiamente descritto ed illustrato nelle Sezioni 2.3.2 e 2.3.3 del Capitolo 2. Mentre nella sua versione originaria questo algoritmo consente di calcolare la distanza fra stringhe e di effettuare un confronto fra i singoli caratteri che le compongono, in questo ambito di applicazione si rende necessario calcolare la distanza fra trascrizioni fonetiche ed effettuare un confronto tra fonemi, i quali, come si è potuto osservare, non sono sempre costituiti da un singolo carattere. Si è reso necessario quindi adattare l'algoritmo che calcola la distanza di *Levenshtein* fra le due trascrizioni fonetiche affinché fosse in grado di effettuare un confronto fra fonemi di lunghezze differenti. La sua implementazione in Android è osservabile all'interno della classe `PhoneticErrorChecking` integrata in *Prime Frasi*. Per consentire un confronto tra fonemi, le due trascrizioni fonetiche sono state trattate come due *array* contenti la sequenza ordinata dei fonemi, codificati nello standard SAMPA, di cui sono composte. L'implementazione in Java dell'algoritmo di *Wagner-Fischer* ricalca lo pseudocodice dell'algoritmo EDITDISTANCE illustrato nella Sezione 2.3.2 del Capitolo 2, e restituisce, oltre alla distanza fra le due trascrizioni fonetiche, anche la *distance matrix* utile al meccanismo di *backtracking*.

Il meccanismo di *backtracking* consente di calcolare un'`ArrayList` di *array* di interi contente gli indici delle celle della *distance matrix* che, in sequenza, costituiscono l'allineamento ottimo fra le due trascrizioni fonetiche. A partire da questo risultato e analizzando l'allineamento ottimo fra le trascrizioni fonetiche, è possibile risalire a quale fonema della trascrizione fonetica restituita dal modello di classificazione corrisponde un *matching*, a quale corrisponde una cancellazione e a quale un'omissione o un inserimento. Infatti, confrontando gli indici di ciascuna cella che costituisce l'allineamento ottimo con gli indici della casella che lo precede nell'allineamento, è possibile capire

se nel percorso dell'allineamento ottimo all'interno della *distance matrix* si sia verificato uno spostamento verso il basso (omissione), uno spostamento verso destra (inserimento), o uno spostamento in diagonale. In quest'ultimo caso, se i fonemi delle due trascrizioni fonetiche coincidono si è trattato di un *matching*, altrimenti di una sostituzione.

Per vedere all'opera questo meccanismo che consente di processare la trascrizione fonetica riconosciuta, prendiamo in esame la trascrizione fonetica corretta della frase "spinge gli gnomi", che nel database di *Prime Frasi* è archiviata come "`sil s p i n dZ e L i J o m i sil`", e supponiamo di doverla confrontare con la trascrizione restituita dal classificatore "`sil s p i z e L i J o m o sil @sch`". Da una prima lettura si contano, nella trascrizione fonetica restituita dal classificatore, l'omissione del fonema `/n/`, la sostituzione del fonema `/dZ/` con il fonema `/z/`, la sostituzione del fonema `/i/` con il fonema `/o/` e, alla fine, l'inserimento del marcatore `@sch`, per un totale di quattro tra sostituzioni, omissioni ed inserimenti. Come accennato, le due trascrizioni fonetiche vengono trattate come *array* di fonemi e marcatori e, in Tabella 3.2, è mostrata la *distance matrix* calcolata dall'algoritmo di *Wagner-Fischer* per questi ultimi.

Si noti che l'ultimo elemento in basso a destra della matrice contiene il valore 4, il quale rappresenta, come ci si aspettava, il minimo numero di sostituzioni, omissioni ed inserimenti necessari per trasformare una trascrizione fonetica nell'altra, nonché la distanza di *edit* minima tra le due. Fornendo questa matrice in input al metodo `computeBacktrace`, sempre definito ed implementato nella classe `PhoneticErrorChecking` di *Prime Frasi*, viene restituito in output la lista di *array* contenete gli indici delle celle che rappresentano l'allineamento ottimo; in questo caso viene restituita la lista di *array* `<[1, 1], [2, 2], [3, 3], [4, 4], [5, 4], [6, 5], [7, 6], [8, 7], [9, 8], [10, 9], [11, 10], [12, 11], [13, 12], [14, 13], [14, 14]>`, dove il primo indice di ciascuna coppia si riferisce alla riga della matrice, mentre il secondo alla colonna. In Tabella 3.2, è messo in evidenza, in grigio, il cammino che, partendo dall'elemento in alto a sinistra e terminando nell'elemento in basso a destra, rappresenta il cammino ottimo. Si noti che ogniqualvolta il cammino scende verso il basso rispetto alla direzione diagonale si verifica un'omissione, ogniqualvolta il cammino procede verso destra si verifica un inserimento, mentre quando scende in diagonale si verifica una sostituzione o un *matching*.

A questo punto, avendo a disposizione le posizioni dei fonemi con i loro rispettivi stati, non resta che convertire la trascrizione fonetica restituita dal

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
			sil	**s**	**p**	**i**	**z**	**e**	**L**	**i**	**J**	**o**	**m**	**o**	**sil**	**@sch**
0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**sil**	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	**s**	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12
3	**p**	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11
4	**i**	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10
5	**n**	5	4	3	2	1	1	2	3	4	5	6	7	8	9	10
6	**dZ**	6	5	4	3	2	2	2	3	4	5	6	7	8	9	10
7	**e**	7	6	5	4	3	3	2	3	4	5	6	7	8	9	10
8	**L**	8	7	6	5	4	4	3	2	3	4	5	6	7	8	9
9	**i**	9	8	7	6	5	5	4	3	2	3	4	5	6	7	8
10	**J**	10	9	8	7	6	6	5	4	3	2	3	4	5	6	7
11	**o**	11	10	9	8	7	7	6	5	4	3	2	3	4	5	6
12	**m**	12	11	10	9	8	8	7	6	5	4	3	2	3	4	5
13	**i**	13	12	11	10	9	9	8	7	6	5	4	3	3	4	5
14	**sil**	14	13	12	11	10	10	9	8	7	6	5	4	4	3	**4**

Tabella 3.2: Esempio di *distance matrix* fra due trascrizioni fonetiche con relativo *backtracking*

classificatore nello standard IPA mettendo in evidenza col colore verde i fonemi che hanno corrisposto quelli della trascrizione fonetica corretta, in rosso i fonemi che sono stati distorti o sostituiti, in grigio i fonemi che sono stati omessi e in blu quelli che sono stati inseriti. Per ottenere questo risultato la classe `PhoneticConverter` di *Prime Frasi* è stata arricchita con un metodo che provvede ad effettuare la conversione di ciascun fonema nello standard IPA, colorando ciascuno di essi del colore corrispondente attraverso gli appositi metodi delle classi `SpannableString` e `SpannableStringBuilder` di Android [40] [41]. La trascrizione fonetica così processata viene visualizzata a schermo sull'interfaccia di *Prime Frasi* insieme alla trascrizione fonetica corretta e alla trascrizione fonetica riconosciuta dal classificatore prima del rilevamento degli errori. Il logopedista può in questo modo riconoscere a colpo d'occhio gli eventuali errori di pronuncia commessi dal paziente e intervenire di conseguenza. Al logopedista viene comunque lasciata la possibilità di contrassegnare la trascrizione fonetica come "corretta" o "errata", poiché alla fine, soltanto la sua esperienza sul campo, può portare ad un'interpretazione accurata di quanto pronunciato dal paziente, confermando o smentendo in questo modo l'output del classificatore fonetico. Per concludere l'esempio che è stato analizzato poc'anzi, il risultato della conversione risulta essere "silspiizeʎiŋomosil@sch". In Figura 3.1 si può osservare il risultato del riconoscimento fonetico e del controllo automatico degli errori che viene visualizzato sull'interfaccia di *Prime Frasi* e che è stato ottenuto durante un test dell'applicazione.

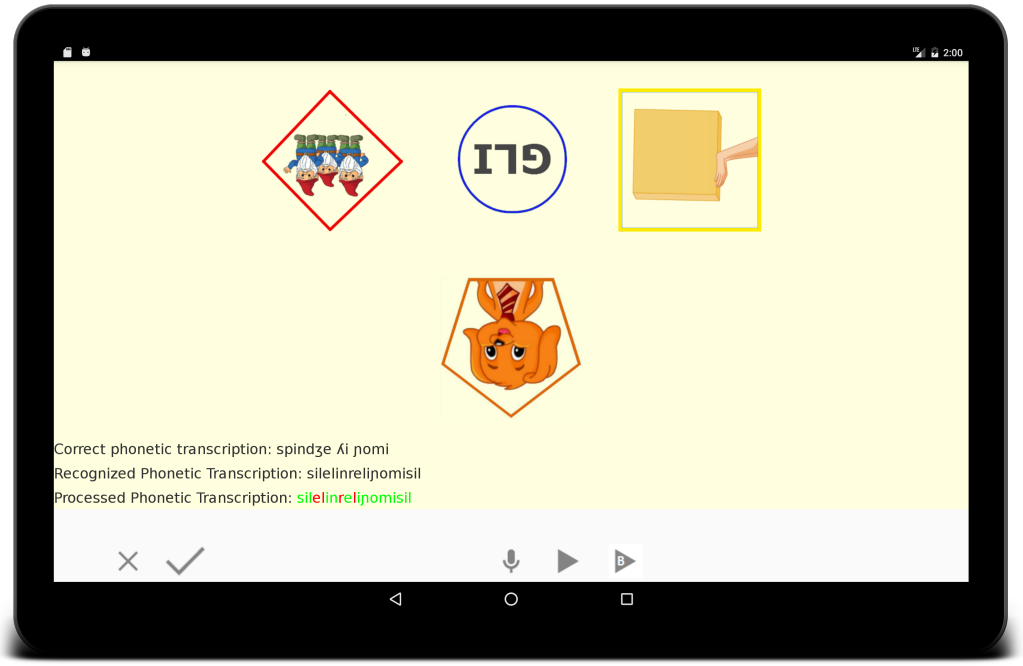

Figura 3.1: Interfaccia di *Prime Frasi* che mostra il risultato del riconoscimento fonetico e l'esito dell'algoritmo di controllo e rilevamento automatico degli errori.

Capitolo 4

Test sperimentali e risultati ottenuti

Dopo aver arricchito ed esteso *Prime Frasi* con la funzionalità di riconoscimento automatico degli errori di pronuncia, prima di avviare un vero e proprio test sul campo di tale applicazione, è bene effettuare dei test con la finalità di sperimentare la bontà del riconoscimento fonetico, in modo da stimarne l'accuratezza e la sensibilità al variare della sorgente con cui viene acquisito il parlato da analizzare e al variare dei fonemi che devono essere riconosciuti. A tal proposito, è bene ricordare che il riconoscimento automatico degli errori su di una particolare trascrizione fonetica è soltanto l'ultimo tassello di un lungo e complesso processing che subisce l'audio acquisito e risente quindi di tutti gli errori che, seppur contenuti, fin dalle prime fasi si propagano e si sommano l'un l'altro. Infatti, una volta che l'audio è stato acquisito, esso viene processato ed analizzato al fine di distinguere e separare la voce del paziente bambino da quella del logopedista adulto: questa operazione è resa possibile combinando le funzionalità della libreria *openSMILE*[1] per l'estrazione delle *features* dall'audio con appositi modelli di classificazione messi a punto con il software *Weka*; per quanto precisi siano questi strumenti, la classificazione e la distinzione della voce dell'adulto da quella del bambino non è immune da errori (per approfondimenti su questa prima fase del processing si veda [4]). La sola voce del bambino così estratta viene quindi inviata ad un server, dove entra in azione il modello di classificazione fonetico messo a punto grazie a *Kaldi* e appositamente allenato per il riconoscimento fonetico sulla voce dei bambini. È evidente che anche quest'ultima fase non è di certo priva di errori, poiché la bontà del riconoscimento è strettamente collegata alla precisione del modello di classificazione utilizzato (per

[1] `http://audeering.com/research/opensmile/`

approfondire questa seconda fase si veda [5]).

I fattori che in generale influenzano la bontà e la qualità del riconoscimento fonetico finale, nonché l'accuratezza di ciascuna fase del processing dell'audio, sono molteplici e di natura diversa. Particolare importanza è ricoperta, oltre che dall'ambiente in cui avviene la registrazione, e quindi dall'eventuale presenza di rumore, anche soprattutto dal microfono utilizzato per l'acquisizione. Infatti, dal momento che l'applicazione *Prime Frasi* è pensata e realizzata per dispositivi Android e vista l'immensa varietà di hardware che caratterizza i dispositivi di questa piattaforma mobile, è importante effettuare dei test per comprendere quanto sensibili siano i risultati del riconoscimento fonetico al variare del dispositivo mobile e della sorgente di acquisizione audio. Un altro fattore che si rende necessario indagare per comprendere l'accuratezza di ciascuna trascrizione fonetica è l'abilità da parte del classificatore fonetico utilizzato da *Kaldi* nel riuscire a riconoscere più facilmente certi fonemi piuttosto che altri: in altre parole è utile effettuare dei test per capire se parole contenti *foni* complessi vengono riconosciute con maggiore difficoltà rispetto a parole caratterizzate da fonemi più semplici. I fonemi più complicati da riconoscere sono quelli corrispondenti ai suoni che nella lingua italiana vengono rappresentati attraverso digrammi e trigrammi; inoltre, contrariamente a quanto si può pensare, anche le vocali fonetiche non sono semplici da riconoscere e trascrivere all'interno di una trascrizione fonetica, in quanto molto difficili da distinguere dalle semiconsonanti quando si tratta di analizzare complessi vocalici all'interno di frasi o singole parole.

Lo scopo di questo capitolo è quello di illustrare la sperimentazione effettuata per indagare i fattori che maggiormente influenzano la qualità della trascrizione fonetica finale che è di fondamentale importanza per l'esperto logopedista e cercare di stabilire quindi se l'applicazione *Prime Frasi* è in grado di fornire risultati affidabili durante gli eventuali test sul campo.

4.1 Setup sperimentale e dispositivi utilizzati

Per effettuare i test appena citati ed ottenere quindi una misura della bontà del riconoscimento fonetico, sono state ricreate condizioni di registrazione ideali e controllate affinché nessun agente o fattore esterno potesse influire sull'esito della sperimentazione. In particolare, per verificare quanto la qualità del dispositivo di registrazione influisse sul risultato del riconoscimento fonetico, sono state effettuate registrazioni di parlato utilizzando cinque diversi

dispositivi, aventi microfoni differenti per quanto riguarda caratteristiche e performance. I dispositivi utilizzati sono elencati di seguito:

- *Samsung Galaxy W*: è uno smartphone Android semplice, disponibile in commercio dall'aprile del 2011; durante i test effettuati il dispositivo era dotato di *Android KitKat 4.4.2*. Le registrazioni sono state eseguite utilizzando l'applicazione sviluppata da *Splend Apps* chiamata "Registratore vocale" e disponibile gratuitamente sul *Google Play Store*. Grazie a quest'ultima è stato possibile registrare l'audio in formato Pulse-code modulation (PCM) *WAVE*, con frequenza di campionamento a *16kHz* e una quantizzazione di *16bit*.

- *Samsung Galaxy Tab 3*: è un tablet Android, disponibile in commercio a partire dal secondo trimestre del 2013; durante i test effettuati il dispositivo era dotato di *Android Marshmellow 6.0*. Per effettuare le registrazioni è stata utilizzata la medesima applicazione utilizzata sul *Samsung Galaxy W*, con le medesime configurazioni di registrazione.

- *Google Nexus 7 (2012)*: è il tablet Android ideato da *Google* e prodotto da *Asus*, immesso nel mercato a partire da luglio 2012; durante le registrazioni era equipaggiato con *Android Jelly Bean 4.1*. L'applicazione utilizzata e le modalità di registrazione rimangono le medesime utilizzate per i due dispositivi sopra citati e descritti.

- *Microsoft Surface Pro 4*: si tratta del famoso personal computer di casa *Microsoft*, disponibile in commercio dall'ottobre 2015. Al momento in cui sono state effettuate le registrazioni il dispositivo era dotato del sistema operativo *Windows 10 Pro*. Per effettuare le registrazioni è stato utilizzato il microfono integrato del dispositivo ed il software *Audacity*, impostato affinché registrasse audio in formato non compresso *WAVE* con frequenza di campionamento sempre a *16kHz* e quantizzazione pari a *16bit*.

- *Zoom H1*: è un registratore digitale semiprofessionale, ideale per acquisire musica e parlato. È dotato di due capsule microfoniche disposte con configurazione X/Y[2] in grado di garantire un'immagine stereo pulita dei suoni registrati. Consente di registrare file audio nel formato *MP3* e *WAVE*: in *WAVE* consente di registrare con frequenza di campionamento di *44.1*, *48* o *96kHz* e una quantizzazione di *16* o *24bit*; in

[2]La configurazione X/Y consiste nell'orientare i due microfoni, l'uno verso l'altro, a formare un angolo di circa 90°. Tale configurazione rappresenta una tecnica di ripresa stereofonica molto diffusa.

MP3 il *bit rate* può variare da *48* a *320kbs* con una frequenza di campionamento pari a *44.1kHz*. È dotato di un filtro in grado di effettuare la soppressione automatica del rumore di fondo tagliando dal segnale acquisito alcune basse frequenze ed inoltre il dispositivo permette la regolazione automatica o manuale del volume d'ingresso del segnale. Durante le registrazioni effettuate per i test, l'audio è stato acquisito in formato *WAVE* con frequenza di campionamento pari a *44.1kHz* e quantizzazione pari a *16bit*; il filtro di riduzione del rumore è stato disattivato ed il volume di input è stato manualmente impostato e fissato ad un valore di 60 su di una scala che va da 0 a 100.

Per ricreare le condizioni di registrazione ideali e controllate di cui accennato, e per essere quindi certi che l'audio acquisito fosse il più possibile privo di rumore, mi sono recato, insieme ai colleghi *Loris Del Monaco* e *Nicola Rigato*, che hanno preso parte e collaborato alla sperimentazione, presso il *Centro di Sonologia Computazionale - Sound and Music Computing* (CSC - SMC)[3] del dipartimento di Ingegneria dell'Informazione dell'Università degli Studi di Padova. Il CSC mette a disposizione infatti una cabina silente[4] ideale per effettuare le registrazioni del parlato. All'interno della cabina silente i cinque dispositivi sono stati disposti a semicerchio di fronte al soggetto parlante e ad una distanza da questo di circa 40 centimetri. Il microfono di ciascuno di essi era rivolto verso il centro, in modo da uniformare la ricezione dei suoni. Nel corso dell'esperimento, io e i miei colleghi, abbiamo pronunciato, cercando di mantenere quanto più possibile un tono di voce costante e lineare, le 117 parole che costituiscono le 22 tavole previste dal test di articolazione di Fanzago, avviando la registrazione su tutti e cinque i dispositivi contemporaneamente. Così facendo si sono ottenute le registrazioni, effettuate con cinque diversi dispositivi, delle medesime parole, pronunciate da tre soggetti differenti.

Grazie a queste registrazioni è quindi possibile effettuare un'analisi non solo dell'incidenza che ha il dispositivo di acquisizione sulla qualità del riconoscimento, ma, avendo scelto di pronunciare le parole del test di Fanzago, anche sull'abilità del classificatore fonetico nel saper riconoscere particolari fonemi target all'interno delle parole di ciascuna tavola. Dal momento che non è stato possibile coinvolgere bambini per effettuare le suddette registrazioni,

[3] `http://smc.dei.unipd.it`

[4] Una cabina silente è un ambiente acusticamente isolato all'interno del quale il riverbero è quasi totalmente assente. Ciò è reso possibile grazie al materiale acustico fonoassorbente di forma piramidale che riveste le sue pareti interne ed appositamente progettato per ridurre le onde sonore e garantire una perfetta acustica.

il modello di classificazione fonetico utilizzato è del tutto analogo a quello addestrato per il riconoscimento della voce dei bambini, con l'unica differenza che questo è stato addestrato al riconoscimento della voce degli adulti. Il training e la messa a punto del modello è stato effettuato sempre da un un gruppo di ricerca del CNR di Padova e ci è stato gentilmente offerto dal Dottor *Piero Cosi*.

4.2 Sperimentazione

Prima di procedere ad effettuare il riconoscimento fonetico di ciascuna parola pronunciata si è reso necessario uniformare le caratteristiche audio delle registrazioni provenienti da ciascun dispositivo ed isolare ciascuna parola pronunciata in ogni registrazione. Ciascuna registrazione effettuata infatti, corrisponde ad una tavola del test di articolazione di Fanzago ed è quindi costituita da più parole. Per uniformare le caratteristiche audio delle varie registrazioni ed isolare le varie parole pronunciate è stato utilizzato il software *Audacity*, il quale ha permesso di convertire le registrazioni effettuate con il registratore *Zoom H1* da *stereo* a *mono* e di ricampionare il segnale ad una frequenza di *16kHz* portando le sue registrazioni in linea con le caratteristiche audio dei file registrati dagli altri dispositivi; in tutte le registrazioni, basandosi sull'osservazione della forma d'onda del segnale, sono stati quindi effettuati dei tagli in corrispondenza dell'inizio e della fine di ciascuna parola pronunciata, avendo premura di lasciare qualche istante di silenzio prima dell'inizio e dopo la fine di ciascuna di esse. In questo modo è stato possibile isolare tutte le parole pronunciate e predisporre quindi i file, uno per ogni parola, da far processare al classificatore fonetico di *Kaldi*. Per come è stato allenato e messo a punto quest'ultimo, per poter garantire risultati di classificazione ottimali, esso necessita di file audio con frequenza di campionamento a *16kHz* e quantizzazione di *16bit* ed è per questo che le registrazioni effettuate da tutti i dispositivi sono state uniformate a quelle specifiche caratteristiche.

A questo punto, per giungere a risultati concreti riguardo questa sperimentazione è necessario avere a disposizione uno strumento in grado di misurare la bontà di ciascuna trascrizione fonetica restituita dal modello di classificazione. Se a *Kaldi* vengono fornite le trascrizioni fonetiche corrette di ciascuna parola che di volta in volta deve essere riconosciuta, esso, attraverso un meccanismo di *scoring* è in grado di fornire il *Word Error Rate* (WER) relativo alla trascrizione fonetica restituita per ciascuna parola. Il WER, che in questa sede sarebbe più appropriato chiamare *Phoneme Error Rate* (PER), è una

misura delle performance dell'*Automatic Speech Recognition* (ASR) e rappresenta il rapporto tra la distanza di *Levenshtein* presente fra la trascrizione fonetica corretta che è stata fornita e quella restituita dal classificatore, e la lunghezza della trascrizione fonetica corretta. In altre parole, se a è la trascrizione fonetica corretta di riferimento e b è la trascrizione fonetica restituita dal classificatore si ha:

$$PhonemeErrorRate = \frac{lev_{\mathrm{a,b}}(|a|,b|)}{|a|} = \frac{S+D+I}{|a|}$$

dove $lev_{\mathrm{a,b}}(|a|, b|)$ rappresenta la distanza di *Levenshtein* fra le due trascrizioni fonetiche, S rappresenta il numero di sostituzioni di fonemi fra una trascrizione fonetica e l'altra, D il numero di cancellazioni ed I il numero di inserimenti.

Attraverso questa analisi è stato possibile indagare quale fosse il dispositivo le cui registrazioni hanno portato a valori di *Phoneme Error Rate* minori e che quindi rendesse il riconoscimento dei fonemi più preciso ed affidabile. È emerso che le registrazioni che sono state riconosciute in modo migliore appartengono al *Samsung Galaxy W* e al registratore *Zoom H1*; per commenti ed analisi più approfondite riguardo questa prima analisi e i relativi risultati ottenuti, nonché per approfondimenti sugli *script* che hanno permesso di automatizzare il meccanismo di *scoring* attraverso *Kaldi*, si rimanda alla lettura del lavoro di tesi svolto dal collega *Nicola Rigato* [5]. In questa sede si vuole approfondire l'analisi che, a partire dai risultati forniti dal modello di classificazione, mira a verificare la precisione con cui ciascun fonema viene riconosciuto, in modo da stabilire come si comporta il modello di classificazione con il riconoscimento dei fonemi più complessi e poter quindi decidere se integrare o meno il test di Fanzago all'interno dell'applicazione *Prime Frasi*.

Per effettuare questa analisi è stata realizzata un'applicazione Android che, a partire dalle trascrizioni fonetiche restituite dal modello di classificazione di *Kaldi*, calcolasse in maniera automatica, per ogni tavola del test di articolazione, il cosiddetto *Target Phoneme Error Rate* (Target-PER), che rappresenta la percentuale di parole in cui il fonema target è stato riconosciuto correttamente. Intuitivamente, partendo dal presupposto che io e i miei colleghi non presentiamo difetti di pronuncia e non siamo affetti da disturbi del linguaggio che possano invalidare la veridicità di questi test, se si riscontrano, per ognuno di noi, Target-PER molto bassi per un determinato fonema, significa che, con ogni probabilità, tale fonema viene riconosciuto senza troppe difficoltà ed in maniera affidabile dal classificatore fonetico; viceversa, se alcuni fonemi, pronunciati correttamente, vengono riconosciuti

Figura 4.1: L'interfaccia da cui è possibile scegliere la tavola ed il relativo fonema target di cui si vogliono visualizzare i risultati dell'analisi.

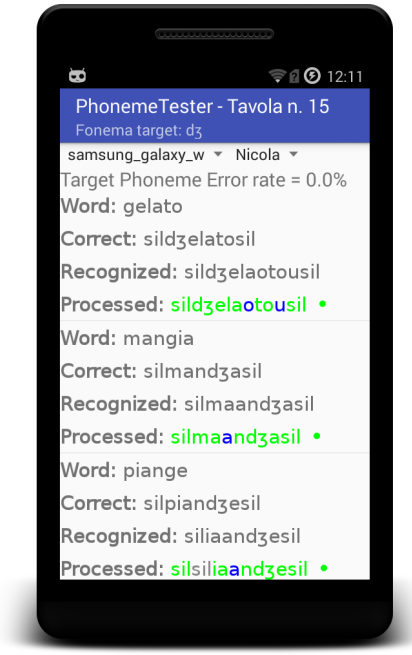

Figura 4.2: Interfaccia che visualizza i risultati dell'analisi per la tavola ed il fonema target relativi al dispositivo e alla persona scelti.

in maniera errata dal classificatore, e le relative tavole presentano quindi Target-PER piuttosto elevati, allora, per contro, si può dire che il riconoscimento di quei fonemi è poco affidabile. L'applicazione che è stata messa a punto per eseguire questa indagine si chiama `PhonemeTester` e permette di osservare, per ogni persona che ha pronunciato le parole del test, e per ogni dispositivo utilizzato, l'esito del confronto fra la trascrizione fonetica corretta e la trascrizione fonetica restituita dal modello di classificazione, insieme al Target-PER relativo a quella specifica tavola. In Figura 4.1 è possibile osservare l'interfaccia che permette di scegliere la tavola, e quindi il fonema, del quale si vogliono visualizzare i risultati dell'analisi, mentre, in Figura 4.2 è possibile osservare l'interfaccia che mostra i risultati dell'analisi per ciascuna parola della tavola scelta.

Si noti che in quest'ultima schermata è possibile scegliere il dispositivo con cui è stata effettuata la registrazione e la persona che ha pronunciato le parole della tavola, in modo da osservare come variano i risultati del riconoscimento fonetico. È inoltre visualizzato il *Target Phoneme Error Rate* relativo al fonema target della tavola selezionata. Si osservi che quest'ultimo non tiene

conto dei fonemi distorti, inseriti o omessi che siano diversi dal fonema target, ma si interessa soltanto di rilevare se il fonema target di quella tavola è stato distorto, omesso o erroneamente inserito. In questo modo, è possibile verificare se, con le parole e i relativi fonemi target che costituiscono il test di articolazione di Fanzago, il classificatore fonetico risulta essere realmente affidabile. Le trascrizioni fonetiche relative alle parole di ciascuna tavola e che vengono fornite come input all'applicazione `PhonemeTester` per l'elaborazione, sono rappresentate all'interno di file formato *CSV*, appositamente generati dallo *script* che realizza lo *scoring* del classificatore fonetico attraverso *Kaldi*; per questo motivo, qualora in futuro si volesse replicare questo test con un numero maggiore di voci, magari di bambini, o con dispositivi differenti, sarà sufficiente, con un minimo sforzo, fornire in input all'applicazione i file *CSV* relativi alle nuove registrazioni per ottenere dei risultati immediati.

4.3 Risultati ottenuti e commenti

Per formalizzare le analisi effettuate attraverso l'applicazione `PhonemeTester` e per ottenere quindi una stima della bontà della classificazione fonetica in relazione al fonema target da riconoscere, è stato calcolato il *Target-Phoneme Error Rate* medio relativo a ciascuna tavola. Una volta fissato il dispositivo di registrazione, il *Phoneme Error Rate* medio relativo ad un determinato fonema target, non è altro che la media dei *Phoneme Error Rate* relativi alle tre voci che hanno pronunciato le parole della tavola relativa a quello specifico fonema. In questo modo non si stanno più calcolando le performance del classificatore fonetico al variare del dispositivo, bensì al variare del fonema da riconoscere che è l'obiettivo ultimo di questo test. Poiché, come già accennato, i test precedenti hanno evidenziato che le trascrizioni fonetiche migliori sono quelle elaborate a partire dalle registrazioni del *Samsung Galaxy W* e del registratore *Zoom H1*, il calcolo del *Phoneme Error Rate* medio per ciascun fonema target è stato eseguito per questi due dispositivi. In Figura 4.3 è possibile osservare le percentuali relative ai *Phoneme Error Rate* medi di ciascun fonema target relativo alle consonanti presenti sulle tavole del test di articolazione di Fanzago.

Nonostante i risultati relativi al *Phoneme Error Rate* totale (o WER) per questi due dispositivi non fossero particolarmente incoraggianti (vedi [5]), i risultati ottenuti attraverso questa analisi più fine, mettono in evidenza il fatto che, in realtà, il classificatore fonetico è in grado di riconoscere correttamente la maggior parte dei fonemi target inseriti nelle parole del test di

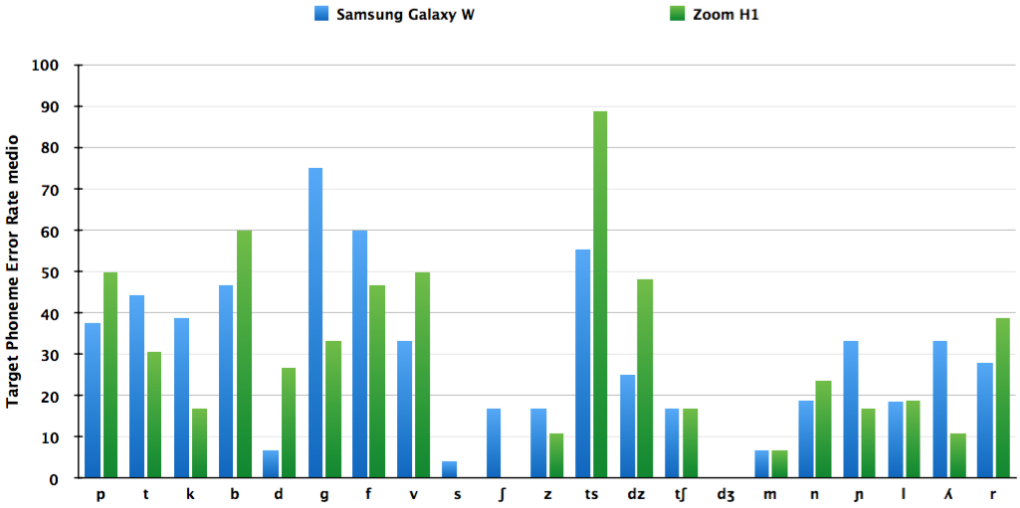

Figura 4.3: *Phoneme Error Rate* medi relativi ai fonemi consonantici target delle tavole del test di Fanzago. In blu sono evidenziati i risultati relativi alle registrazioni effettuate con il *Samsung Galaxy W*, in verde invece, i risultati relativi alle registrazioni effettuate con il registratore *Zoom H1*.

Fanzago e che la maggior parte degli errori che contribuiscono ad aumentare il WER, sono dovuti ad inserimenti ed omissioni di fonemi diversi dal fonema target. Per fare un esempio, si prendano in esame i risultati dell'analisi relativa al fonema target `/dʒ/` mostrati in Figura 4.2: il *Phoneme Error Rate* relativo a ciascuna parola della tavola n. 15, pronunciata da Nicola e registrata con il *Samsung Galaxy W*, non è pari a zero, ma per questo motivo non si può dire che il classificatore non sia in grado di riconoscere correttamente il fonema target `/dʒ/`. Gli errori di classificazione che si riscontrano in questo esempio infatti, sono tutti dovuti ad inserimenti (fonemi evidenziati in blu) ed omissioni (fonemi evidenziati in grigio), ma non a distorsioni che sarebbero il segno dell'errato riconoscimento di qualche particolare fonema. Un discorso simile vale per molti altri fonemi e situazioni analoghe si incontrano di frequente esaminando i risultati delle analisi di ciascun fonema target all'interno dell'applicazione `PhonemeTester`.

Bisogna inoltre far notare che, dal punto di vista della produzione fonatoria, molti fonemi hanno luogo e modo di articolazione molto simili e, per questo motivo, la loro pronuncia si assomiglia ed è difficile distinguerli per il classificatore. Esaminando ad esempio la tavola n. 12 pronunciata da Nicola e registrata con lo *Zoom H1*, il cui fonema target è `/ts/`, si può notare come la parola `[caltse]` (calze), venga riconosciuta dal classificatore e processata

come [caldze]. L'errore commesso dal classificatore in questo caso non è da considerarsi particolarmente grave poiché ha confuso fonemi consonantici che sono entrambi affricati alveolari, rispettivamente per modo e luogo di articolazione, e che si distinguono soltanto per la loro sonorità. Questo fatto si ripercuote sui *Phoneme Error Rate* medi relativi ai fonemi /ts/ e /dz/, che, proprio per questo motivo, risultano essere particolarmente elevati rispetto a quelli di altri fonemi. Un altro esempio di questo fenomeno è riscontrabile nella tavola n. 14, pronunciata da me, registrata sempre con lo *Zoom H1* ed il cui fonema target è /tʃ/. In questo caso la parola [pantʃa] (pancia) è stata riconosciuta come [tandʒa]: si può notare come i fonemi /tʃ/ e /dʒ/ siano stati erroneamente confusi; tuttavia, entrambi questi fonemi, appartengono alla categoria delle consonanti affricate palato-alveolari e, come per la coppia di fonemi precedenti, si distinguono soltanto per la loro sonorità che è più difficile da percepire e classificare. Sempre all'interno di questa parola, si può notare inoltre come il fonema consonantico occlusivo /p/, pur non essendo il fonema target, sia stato sostituito con la consonante occlusiva /t/.

Queste osservazioni mettono in luce, come ci si poteva aspettare, la maggiore difficoltà da parte del classificatore nel saper distinguere con precisione fonemi consonantici appartenenti alla stessa classe (vedi Tabella 2.1), tuttavia, per la maggior parte dei fonemi, i risultati ottenuti sono abbastanza soddisfacenti. Infatti, per la maggior parte delle consonanti occlusive, il *Target-Phoneme Error Rate* medio calcolato non supera il 40%, per le consonanti fricative alveolari e palato-alveolari non supera mai il 20% per entrambi i dispositivi, così come per le affricate palato-alveolari e le consonanti nasali e laterali; per le consonanti polivibranti, infine, il *Target-Phoneme Error Rate* medio non supera il 30% per le registrazioni effettuate con il *Samsung Galaxy W* e non supera il 40% per quelle effettuate con lo *Zoom H1*. *Target-Phoneme Error Rate* medi maggiori si ottengono soltanto per le fricative labio-dentali, le affricate alveolari e alcune occlusive.

Inoltre, dall'osservazione delle trascrizioni restituite dal classificatore fonetico di *Kaldi* circa le parole contenti i fonemi /ʃ/, /ts/, /dz/, /ɲ/ e /ʎ/ in posizione intervocalica, è emerso che tali fonemi non vengono mai raddoppiate dal classificatore, contravvenendo alla convenzione fonetica tale per cui questi fonemi dovrebbero essere sempre lunghi o geminati. Questa constatazione fa capire che, con ogni probabilità, il modello di classificazione che è stato fornito e che è stato integrato in *Prime Frasi*, non è stato allenato tenendo conto di tale regola. Per ovviare a ciò, si potrebbe intervenire raddoppiando in automatico tali fonemi (quando essi compaiono in posizione intervocalica) durante la fase di riconoscimento degli errori, così da migliora-

re la qualità del riconoscimento ed abbassare il *Phoneme Error Rate* generale.

Per quanto riguarda il riconoscimento dei fonemi vocalici e semiconsonantici, osservando il risultato delle analisi sulle trascrizioni fonetiche restituite per le parole della tavola n. 22 e per i dispositivi presi in esame, si può affermare che, per quanto concerne le vocali singole, il riconoscimento è stato eseguito correttamente la quasi totalità delle volte, mentre, per quanto riguarda il riconoscimento di gruppi vocalici (dittonghi e iati), talvolta il riconoscimento è stato confuso con i fonemi semiconsonantici o approssimanti. Per fornire degli esempi a supporto di queste considerazioni, si prendano in esame parole come "piede", la cui trascrizione fonetica proposta nelle tavole del test di Fanzago è [`piede`]: in diverse occasioni, tale parola è stata riconosciuta come [`pjede`], nella quale il dittongo ascendente "ie" è stato stato sostituito con la semiconsonante `/j/`. Un altro esempio risiede nella parola "auto", la cui trascrizione fonetica fornita è [`awto`]: in molte occasioni, tale parola, è stata riconosciuta come [`auto`], dove, come si può notare, il fonema semiconsonantico `/w/` è stato sostituito con il fonema vocalico `/u/`. Esempi analoghi sono riscontrabili in molte altre parole del test di articolazione, tuttavia, questi, non sono da considerarsi veri e propri errori di riconoscimento poiché le differenze tra dittonghi, iati e rispettive semiconsinanti risiedono in sfumature e dettagli che dipendono dall'evoluzione storica della lingua italiana e dai suoi dialetti [42] e che, per questi motivi, sono spesso difficili da percepire anche dall'esperto logopedista. Dal momento che l'errato riconoscimento dei dittonghi con i rispettivi fonemi approssimanti o gruppi vocalici non evidenzia la presenza di particolari difetti di pronuncia in quanto le differenze fra vocali sono più scientificamente definibili in termini acustici e percettivi anziché articolatori [12], sarebbe conveniente trascurare questo tipo di errori di riconoscimento, concentrando maggiormente l'attenzione sui riconoscimenti errati dei fonemi consonantici.

Alla luce dei risultati che sono stati messi in evidenza si potrebbe pensare, in futuro, di integrare il test di valutazione dell'articolazione di Fanzago all'interno dell'applicazione *Prime Frasi* così che i logopedisti possano testarne sul campo la reale efficienza ed affidabilità. Nonostante le performance ottenute sul riconoscimento dei fonemi siano abbastanza soddisfacenti in relazione agli obiettivi iniziali che ci eravamo proposti, è di fondamentale importanza affinare il più possibile il modello di classificazione fonetica utilizzato da *Kaldi*, soprattutto per quanto riguarda il discernimento di tutti quei fonemi consonantici che, dal punto di vista della produzione fonatoria, presentano lo stesso luogo e modo di articolazione. È proprio l'errata pronuncia di questi ultimi che evidenziano nel paziente la presenza di difetti di pronuncia

quali ad esempio il tetacismo, il cappacismo, il gammacismo ecc., e che suggeriscono al logoterapeuta gli esercizi di correzione e la corretta terapia da somministrare al paziente [43]. Inoltre, per ottenere risultati ed indicazioni più precisi circa la precisione del classificatore nel riconoscere al meglio tutti i fonemi, sarebbe opportuno ripetere questa sperimentazione, eventualmente anche in presenza di uno o più logopedisti, coinvolgendo un numero di persone maggiore, magari bambini di entrambi i sessi, che non presentino difetti di pronuncia. Sarebbe inoltre conveniente ripetere i test negli ambienti in cui solitamente vengono svolte le sedute logopediche con i pazienti, per capire quanto il riverbero e le altre variabili acustiche che caratterizzano questi luoghi impattino sulla qualità delle registrazioni e quindi sui risultati del riconoscimento. I tentativi e i test esposti in questo lavoro di tesi vogliono essere un'analisi ed una verifica preliminari sui risultati che potranno essere ottenuti affinando sempre di più il modello di classificazione ed intensificando la collaborazione con i logopedisti; vogliono inoltre offrire alcuni strumenti e metriche che siano in grado di elaborare e formalizzare esaustivamente i risultati ottenuti a partire dalle registrazioni che saranno effettuate nel corso dei futuri test.

Capitolo 5

Conclusioni e Sviluppi Futuri

L'obiettivo di questa tesi era quello di mettere a punto ed integrare in Android un sistema automatico in grado di rilevare i difetti di pronuncia a partire da trascrizioni fonetiche, ottenute attraverso un modello di classificazione e riconoscimento dei fonemi e sulla base di registrazioni audio della voce dei pazienti, acquisite dai microfoni dei dispositivi mobili. È evidente che per raggiungere un obiettivo simile, che mescola competenze in campo informatico e conoscenze in ambito logopedico, non si può prescindere dallo studio e dall'approfondimento del background fonetico e fonologico nel quale si va ad operare.

Prima di procedere all'implementazione ed al lavoro vero e proprio infatti, è stato effettuato uno studio, oltre che sulle modalità con cui avvengono le sedute logopediche con i bambini, anche sugli strumenti e sulle conoscenze necessari per l'attuazione dell'obiettivo finale. In particolare, grazie all'interazione con la dottoressa e logopedista *Manuela Susigan*, la quale ha provveduto a fornirmi materiale ed informazioni utili al riguardo, è stato possibile effettuare uno studio approfondito, sia sull'alfabeto fonetico della lingua italiana e sulle tipologie di difetti di pronuncia più diffuse, sia su alcuni strumenti utilizzati dai logopedisti per testare l'inventario fonetico dei pazienti durante le prime sedute. La dottoressa *Susigan* ha inoltre provveduto a fornirmi registrazioni audio di vere e proprie sedute con alcuni bambini e ciò mi ha permesso di capire meglio il contesto in cui operano i logopedisti; alcune delle registrazioni fornite sono state utilizzate come audio di test durante lo sviluppo di un classificatore in grado di discernere la voce dell'adulto da quella del bambino [4], il che costituisce un'operazione preliminare rispetto a quella di rilevamento automatico degli errori di pronuncia.

Dopo aver compiuto questo studio preparatorio, il quale ha consentito di

coordinarmi con l'ambiente logopedico e con i colleghi *Del Monaco* e *Rigato* che hanno preso parte allo sviluppo e all'integrazione di nuove funzionalità nell'applicazione *Prime Frasi* di LogoKit [4] [5], è stato effettuato uno studio circa lo stato dell'arte delle librerie e dei servizi di terze parti che permettessero operazioni di *Automatic Speech Recognition* e riconoscimento automatico dei fonemi, ma, dopo aver appurato l'inadeguatezza di tali librerie e servizi per il raggiungimento del nostro scopo, si è proceduti con l'utilizzo del toolkit open-source *Kaldi*. Una volta compreso l'output del classificatore fonetico, grazie anche ai preziosi incontri avuti con il dottor *Piero Cosi* del CNR di Padova, ho realizzato un algoritmo in grado di effettuare una conversione fra lo standard di codifica dei fonemi SAMPA e quello IPA, così da rendere le trascrizioni fonetiche in linea con lo standard dei logopedisti. Infine, ho messo a punto ed implementato un algoritmo in grado di riconoscere gli errori di pronuncia attraverso un'analisi della trascrizione fonetica restituita dal classificatore. Per testare la bontà del riconoscimento di ciascun fonema ho dunque realizzato un'applicazione di test che effettuasse il rilevamento degli errori sulle parole delle tavole del test di articolazione di Fanzago pronunciate da me e i miei colleghi e registrate con dispositivi differenti.

I test effettuati hanno evidenziato la possibilità di poter integrare, in un prossimo futuro, il test di articolazione di Fanzago all'interno di *Prime Frasi*, a meno del fatto che siano prima effettuati nuovi test ed analisi con voci differenti. È inoltre auspicabile, arrivati a questo punto, intensificare la collaborazione con i logopedisti, così da potenziare l'algoritmo di rilevamento dei difetti di pronuncia presente in *Prime Frasi*, rendendolo in grado di far corrispondere ai diversi errori riscontrati sulla trascrizione fonetica, il tipo, o i tipi, di difetti di pronuncia e di disturbi del linguaggio che interessano il paziente. Inoltre, visto il fatto che a determinati difetti riscontrati corrispondono determinate terapie e specifici esercizi correttivi, si potrebbe pensare di integrare tali esercizi all'interno di *Prime Frasi*, così che l'applicazione sia in grado di sottoporre automaticamente il paziente agli esercizi più utili per risolvere il suo problema in base ai difetti di pronuncia precedentemente rilevati; in questo modo il paziente diventerebbe, sotto un certo punto di vista, "autonomo" nell'utilizzo dell'applicazione ed il logopedista potrebbe, grazie alla possibilità di riascoltare le registrazioni, correggere le diagnosi proposte dall'applicazione e monitorare i suoi progressi anche a distanza. Raggiungere questo obiettivo richiede ancora molteplici test e sperimentazioni sul campo, ma sarebbe un ottimo risultato nel tentativo e nel processo di snellire e digitalizzare un contesto nel quale l'informatica ancora non aveva messo piede.

Appendice A

Il test di articolazione di Fanzago (1983)

Di seguito sono riportate le 117 parole, suddivise in 22 tavole, che costituiscono il test di articolazione di *Fanzago*. Per ogni tavola è indicato il fonema target che vuole essere indagato dal logopedista quando sottopone il paziente alla lettura delle parole che la costituiscono. Fra parentesi, per facilitarne la lettura, sono riportate le trascrizioni di tutte le parole che contengono uno o più fonemi che non hanno una corrispondenza grafemica biunivoca con le lettere dell'alfabeto ortografico italiano. In Figura A.1 è visibile la tabella di valutazione che attualmente viene compilata dai logopedisti durante la somministrazione del test per ciascun paziente. A partire dall'analisi di tale tabella viene formulata una diagnosi circa la presenza di disturbi del linguaggio nel paziente.

Tavola n. 1 - /p/
palla
pipa
prato
kapra (capra)

Tavola n. 2 - /t/
topo
letto
tromba
kwattro (quattro)

Tavola n. 3 - /k/
kaza (casa)
kane (cane)
oka (oca)
fwoko (fuoco)
krotʃe (croce)
kra-kra (cra-cra)

Tavola n. 4 - /b/
baffi
banana
brattʃo (braccio)
libro
brutto

Tavola n. 5 - /d/
dito
dado
drin
kwadro (quadro)
ladro

Tavola n. 6 - /g/
gatto
gabbia
ago
grande
grasso
magro

Tavola n. 7 - /f/
fiore
fumo
kaffe (caffé)
freddo
frutta

Tavola n. 8 - /v/
vazo (vaso)
vino
uva
avro (avrò)

Tavola n. 9 - /s/
sole
sasso
stella
posta
skopa (scopa)
taska (tasca)
spago
rospo

Tavola n. 10 - /ʃ/
ʃi (sci)
ʃimmia (scimmia)
ʃiarpa (sciarpa)
kuʃino (cuscino)
peʃʃe (pesce)

Tavola n. 11 - /z/
kaza (casa)
roza (rosa)
vizo (viso)
zlitta (slitta)
zbatte (sbatte)
zberla (sberla)

Tavola n. 12 - /ts/
pottso (pozzo)
tattsa (tazza)
kaltse (calze)

Tavola n. 13 - /dz/
dzaino (zaino)
dzebra (zebra)
prandzo (pranzo)
dzampa (zampa)

Tavola n. 14 - /tʃ/
tʃao (ciao)
tʃip-tʃip-uttʃello (cip-cip-uccello)
kaltʃo (calcio)
pantʃa (pancia)

Tavola n. 15 - /dʒ/
dʒelato (gelato)
pioddʒa (pioggia)
mandʒa (mangia)
piandʒe (piange)

Tavola n. 16 - /m/
mano
mela
womo (uomo)
kampana (campana)
gamba

tavola n. 17 - /n/
nave
nonno
pane
banko (banco)
ponte
lingwa (lingua)
imverno (inverno)

tavola n. 18 - /ɲ/
ɲomo (gnomo)
raɲɲo (ragno)
baɲɲo (bagno)
spuɲɲa (spugna)

Tavola n. 19 - /l/
lana
luna
palla
ala
male
talko (talco)
alto
doltʃe (dolce)
kaldo (caldo)

Tavola n. 20 - /ʎ/
foʎʎɑ (foglia)
maʎʎa (maglia)
zveʎʎa (sveglia)

Tavola n. 21 - /r/
rana
karro (carro)
porta
erba
dorme
karne (carne)

Tavola n. 22 - vocali
ape
awto (auto)
oka (oca)
osso
orso
uova
wovo (uovo)
erba
piede
io

Figura A.1: La tabella di valutazione compilata dai logopedisti per ciascun paziente durante la somministrazione del test.

Bibliografia e Sitografia

[1] Enrico Massarente. *Classificazione automatica della voce in ambito logopedico: training e testing di un algoritmo per discriminare la voce adulta da quella dei bambini.* Tesi magistrale, Università degli Studi di Padova, 2015.

[2] Diego Vescovi. *Progetto e realizzazione di un'applicazione mobile per la terapia dei disturbi del linguaggio nei bambini.* Tesi triennale, Università degli Studi di Padova, 2015.

[3] P. Zago. *Lo sviluppo fonologico nel bambino: valutazione e terapia logopedica del disordine fonologico.* Università degli Studi di Padova - Facoltà di Medicina e Chirurgia - Corso di Laurea in Logopedia, A.A. 2011/2012.

[4] Loris Del Monaco. *Classificazione automatica della voce in ambito logopedico: un algoritmo per dispositivi mobili per discriminare la voce adulta da quella dei bambini.* Tesi magistrale, Università degli Studi di Padova, dicembre 2016.

[5] Nicola Rigato. *Classificazione automatica della voce in ambito logopedico: sperimentazione di un riconoscitore fonetico per dispositivi mobili.* Tesi magistrale, Università degli Studi di Padova, dicembre 2016.

[6] Associazione Fonetica Internazionale - Wikipedia. `https://it.wikipedia.org/wiki/Associazione_fonetica_internazionale`, Online accessed: novembre 2016.

[7] International Phonetic Association. `https://www.internationalphoneticassociation.org`, Online accessed: novembre 2016.

[8] Alfabeto fonetico internazionale - Wikipedia. `https://it.wikipedia.org/wiki/Alfabeto_fonetico_internazionale`, Online accessed: novembre 2016.

[9] Giancarlo Schirru. *Dispensa di fonetica.* Università degli Studi di Cassino - Facoltà di Lettere e filosofia - Corso di Glottologia e linguistica, A.A. 2005/2006.

[10] International Phonetic Association. Tabella IPA.
`https://www.internationalphoneticassociation.org/sites/default/files/IPA_Deja_2015.pdf`, 2015.

[11] Claudio Zmarich. *Problemi di trascrizione dei foni dell'Italiano: grafemi vs fonemi. Necessità e importanza della trascrizione fonetica.* Insegnamento di fonetica e fonologia - Corso di Laurea in Logopedia - Facoltà di Medicina e Chirurgia - Università degli Studi di Padova, A.A. 2008/2009.

[12] Claudio Zmarich. *Fonetica articolatoria tradizionale.* Insegnamento di fonetica e fonologia - Corso di Laurea in Logopedia - Facoltà di Medicina e Chirurgia - Università degli Studi di Padova.

[13] Claudio Zmarich. *Il sistema fonologico dell'italiano: norma e variabili sociolinguistiche e regionali.* Insegnamento di fonetica e fonologia - Corso di Laurea in Logopedia - Facoltà di Medicina e Chirurgia - Università degli Studi di Padova, A.A. 2007/2008.

[14] Clara Ferranti. *Fonetica articolatoria: schemi ed eserciziario.* Pubblicazione digitale gratuita - Università degli Studi di Macerata, Macerata, novembre 2013.

[15] UCL Psychology & Language Sciences. SAMPA - Computer readable phonetic alphabet.
`http://www.phon.ucl.ac.uk/home/sampa/`, Online accessed: novembre 2016.

[16] X-SAMPA - Wikipedia.
`https://it.wikipedia.org/wiki/X-SAMPA`, Online accessed: novembre 2016.

[17] Laboratorio di Fonetica Sperimentale "Arturo Genre" di Torino. Tabella IPA multimediale.
`http://www.lfsag.unito.it/ipa/index.html`, Online accessed: novembre 2016.

[18] Laboratorio di Fonetica Sperimentale "Arturo Genre" di Torino. Phonverter.
`http://www.lfsag.unito.it/ipa/converter.html`, Online accessed: novembre 2016.

[19] La valutazione del linguaggio nei bambini: i test. Parte I: fonologia, articolazione e primo linguaggio.
`http://www.trainingcognitivo.it/la-valutazione-del-linguaggio\-nei-bambini-i-test/`, Online accessed: novembre 2016.

[20] Paola Zanchi. *La valutazione del linguaggio*. Facoltà di Psicologia - Università degli Studi di Milano-Bicocca, A.A. 2010/2011. Laboratorio Strumenti di valutazione dello sviluppo linguistico, mnestico e delle funzioni educative.

[21] F. Fanzago. *Test di valutazione dell'articolazione*. Centro Stampa Palazzo Maldura, Padova, 1986.

[22] Flavio Pietrelli. *Il problema dello string-matching*. Tesi magistrale, Università degli Studi di Roma "La Sapienza", A.A. 2010/2011.

[23] Levenshtein distance - Wikipedia.
`https://en.wikipedia.org/wiki/Levenshtein_distance`, Online accessed: novembre 2016.

[24] Daniel Jurafsky & James H. Martin. *Speech and Language Processing*, chapter 2, pages 17–21. Prentice Hall, 3rd edition, 2016.

[25] Wagner-Fischer algorithm - Wikipedia.
`https://en.wikipedia.org/wiki/Wagner--Fischer_algorithm`, Online accessed: novembre 2016.

[26] Robert A. Wagner & Michael J. Fischer. The string-to-string correlation problem. *Journal of the Association for Computing Machinery*, Vol. 21(1), 1974.

[27] Umesh V. Vazirani Sanjoy Dasgupta, Christos H. Papadimitriou. *Algorithms*, chapter 6, pages 174–177. McGraw-Hill Education, 1st edition, september 2006.

[28] Benjamin Langmead. *Dynamic programming and edit distance* (lecture slides). Course of Algorithms for DNA Sequencing - Johns Hopkins Whiting School of Engeneering - Department of Computer Science - Baltimore (Maryland), 2014.

[29] Ronald L. Rivest Clifford Stein Thomas H. Cormen, Charles E. Leiserson. *Introduction to Algorithms*, chapter 15. The MIT Press, 2009.

[30] Dynamic programming - Wikipedia.
`https://en.wikipedia.org/wiki/Dynamic_programming`, Online accessed: novembre 2016.

[31] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, dec 2011.

[32] Android Studio - Wikipedia.
`https://it.wikipedia.org/wiki/Android_Studio`, Online accessed: novembre 2016.

[33] Audacity.
`http://www.audacityteam.org/about/`, Online accessed: novembre 2016.

[34] Watson Services.
`https://www.ibm.com/watson/developercloud/services-catalog.html`, Online accessed: novembre 2016.

[35] wit.ai - Natural Language for Developers.
`https://wit.ai`, Online accessed: novembre 2016.

[36] Amazon Developer - Alexa.
`https://developer.amazon.com/alexa`, Online accessed: novembre 2016.

[37] Microsoft Cognitive Services - Bing Speech API.
`https://www.microsoft.com/cognitive-services/en-us/speech-api`, Online accessed: novembre 2016.

[38] Google: Cloud Speech API.
`https://cloud.google.com/speech/`, Online accessed: novembre 2016.

[39] Android Developers - `SpeechRecognizer`.
`https://developer.android.com/reference/android/speech/SpeechRecognizer.html`, Online accessed: novembre 2016.

[40] Android Developers - `SpannableString`.
`https://developer.android.com/reference/android/text/SpannableString.html`, Online accessed: novembre 2016.

[41] Android Developers - `SpannableStringBuilder`. `https://developer.android.com/reference/android/text/SpannableStringBuilder.html`, Online accessed: novembre 2016.

[42] Dittongo - Treccani - Enciclopedia dell'italiano (2010). `http://www.treccani.it/enciclopedia/dittongo_(Enciclopedia-dell'Italiano)/`, Online accessed: novembre 2016.

[43] E. Perrotta & P. Rustici. *Correggere i difetti di pronuncia.* Edizioni Centro Studi Erickson - Trento, 2006.

Ringraziamenti

Al termine di questa tesi, vorrei ringraziare il mio relatore professor *Antonio Rodà* e correlatore professor *Carlo Fantozzi*, non solo per avermi dato la possibilità di prendere parte a questo progetto che spero possa un giorno diventare di aiuto concreto per l'ambiente logopedico, ma anche per avermi seguito e consigliato durante tutto questo percorso con costanza e disponibilità.

Un doveroso grazie va anche alla dottoressa *Manuela Susigan* che, durante le prime fasi di questo lavoro, ha gentilmente provveduto a fornirmi materiale ed informazioni utili sulle pratiche in uso dai logopedisti, aiutandomi a comprendere meglio questo settore.

Un ringraziamento particolare va, infine, al dottor *Piero Cosi* del CNR di Padova, non solo per il prezioso e numeroso materiale fornito a me e ai miei colleghi, ma anche per i consigli e i molteplici suggerimenti che ci ha dato durante tutti gli incontri svolti presso il CNR.

Grazie di cuore a tutti coloro che, direttamente o indirettamente, mi hanno supportato durante questo interessante lavoro di tesi.