UNIVERSITÀ DI PISA

Corso di Laurea Magistrale in Informatica Umanistica

SubItalian.

Un'applicazione Web per l'apprendimento dell'italiano tramite sottotitolazione e annotazione di audiovisivi.

***Candidato**: Sara Pupi*

***Relatore**: Mirko Tavosanis*

Anno Accademico 2014/2015

A voi che mi avete sostenuta durante questo importante percorso.

A mamma, babbo e Amore.

Contents

1 Introduzione

Il fascino di uno strumento informatico è strettamente dipendente dalle finalità per cui è stato concepito; se dunque il suo scopo è quello di facilitare l'apprendimento della nostra lingua, tale strumento si rivela estremamente importante. SubItalian è un progetto realizzato con lo scopo di dimostrare come le tecnologie informatiche sviluppate negli ultimi anni possano facilitare e incentivare l'apprendimento di una lingua (in questo caso l'italiano) in modo diverso dal solito e, con tutta probabilità, stimolante.

1.1 Apprendimento di una lingua: l'approccio comunicativo e l'utilizzo della TV

L'idea che sta alla base dello sviluppo di questo progetto nasce dal fatto che la costruzione di una strategia di apprendimento coinvolgente ed emotivamente stimolante è un requisito fondamentale in qualsiasi contesto didattico. L'integrazione nelle attività didattiche di un elemento strettamente connesso alle attività del piacere e del relax quotidiano può rivelarsi una buona tattica per predisporre lo studente ad un migliore percorso di apprendimento e stimolare la sua volontà di *imparare*[16].

La televisione è senz'altro una delle fonti di svago più diffuse tra la popolazione e pensare di sfruttarla in modo intelligente utilizzandola, oltre che come mezzo di informazione e intrattenimento, come strumento che abbia uno scopo istruttivo quale quello dell'apprendimento di una lingua straniera, è una tattica tutt'altro che inattuabile. Sembra infatti che guardare la televisione nella *lingua target* sia il secondo modo più popolare utilizzato dagli europei, e il quarto utilizzato dai parlanti cinesi, per imparare una lingua straniera [21].

Già nel 1963, Tullio de Mauro[15] aveva identificato nella televisione una *scuola di lingua* per gli italiani, riferendosi all'apprendimento dell'italiano standard da parte di dialettofoni. Negli anni successivi la sua ipotesi si è rivelata estremamente veritiera, non solo per quanto riguarda la diffusione dell'italiano standard ma, in generale, come strategia di apprendimento di una lingua straniera. Willing[45] ha evidenziato quattro comportamenti propri di un discente che stia effettivamente cercando di apprendere la lingua target: comunicativo, analitico, guidato, concreto. Il primo di questi, quello comunicativo, vede l'apprendente desideroso di ascoltare parlanti nativi apprendendo nuove parole dalle loro conversazioni, cerca di parlare con i propri amici e parenti utilizzando la lingua target e guarda programmi televisivi nella lingua target.

Non sono pochi i casi che dimostrano come la televisione possa rappresentare un importante mezzo di diffusione di una lingua e numerosi sono gli studi che, già a partire dagli anni ’80, trattano tale argomento. Di seguito vengono riportati una serie di esempi tratti da pubblicazioni che riguardano perlopiù l’apprendimento della lingua inglese; come si vedrà più avanti, i concetti e le conclusioni cui giungono gli autori sono perfettamente applicabili anche al caso della lingua italiana.

Nel 1987 è stato condotto uno studio[24] su un campione di 60 studenti di un liceo linguistico cinese. Gli studenti sono stati sottoposti ad una conversazione orale seguita dalla compilazione di un questionario orientato a capire quali fossero le loro strategie di apprendimento extra scolastico dell’inglese orale. I risultati hanno dimostrato che i soggetti che hanno esibito una maggiore capacità di comunicazione orale durante il colloquio sono perlopiù coloro che dichiarano di ricorrere a strategie di apprendimento che prevedono l'uso pratico della lingua. Tra queste strategie troviamo: la comunicazione orale in lingua inglese con altri studenti (o, ove possibile, con parlanti nativi), il pensare o parlare tra sé e sé in lingua inglese, la lettura di giornali in lingua inglese e la fruizione di film e programmi televisivi in lingua inglese.

Nel 1990 è stata condotta un'indagine su otto studenti tra i 24 e i 30 anni iscritti all'IEP (Intensive English Program)[1] provenienti da tre diversi paesi asiatici: Corea, Giappone e Cina. Gli otto studenti intervistati erano tutti laureati e manifestavano un forte desiderio di migliorare le loro competenze nella lingua inglese. I ragazzi sono stati sottoposti ad una serie di domande mirate a comprendere quale fosse il loro rapporto con la lingua target al di fuori dell'ambito didattico. La maggior parte ha dichiarato che l'attività praticata più frequentemente nel tempo libero per migliorare le proprie competenze di inglese era guardare la televisione in lingua originale o addirittura prendere in prestito film. Oltre ad essere l’attività più praticata risultava essere anche quella più piacevole e accessibile; uno dei soggetti ha spiegato: “*I don't have to wait for other people. I can watch it by myself”.*

Pickard[37] ha condotto nel 1996 un'indagine per capire quali fossero le attività atte a migliorare l'apprendimento della lingua target (l'inglese) svolte nel tempo libero dagli studenti di madre lingua tedesca. La ricerca è stata effettuata su un campione di 20 studenti iscritti al primo anno del corso *European Business* alla Lincoln University

[1] http://iep.indiana.edu/

Appendix 2
Out-of-class listening activities

Names	Radio	TV	Theatre	Talks	Lectures	Tapes	Cinema
C.B.	4	0	1	3	0	0	2
J.B.	4	3	1	1	1	0	1
A.F.	4	3	1	0	0	1	0
E.G.	4	4	2	1	0	0	1
K.He.	2	0	2	3	2	0	2
J.H.	2	0	0	0	0	0	1
T.H.	2	4	2	1	0	0	1
S.H.	5	0	1	3	1	1	2
K.Hu.	1	1	2	5	0	1	1
U.K.	3	2	1	0	0	0	1
A.Ka.	4	0	0	0	0	0	1
V.K.	3	5	1	0	0	2	2
A.Ko.	4	1	1	2	1	2	2
A.L.	2	1	1	1	0	0	1
M.O.	0	0	1	0	0	0	1
B.R.	4	2	1	1	1	0	2
H.R.	4	3	1	0	0	0	0
C.S.	5	1	0	1	0	0	1
P.T.	5	2	1	1	0	0	0
G.W.	4	3	1	0	1	0	1
Median	4	1.5	1	1	0	0	1

Figura 1 Attività di apprendimento orale in ambito extra scolastico degli studenti tedeschi iscritti al primo anno del corso European Business alla Lincoln University.

(Regno Unito) aventi studiato l'inglese dagli 11 ai 18 anni e con una buona padronanza di tale lingua. Al campione in indagine è stato sottoposto un questionario mediante il quale hanno potuto dichiarare la frequenza con cui erano usi svolgere attività di lettura, scrittura e ascolto in orario extra scolastico. Per quanto riguarda l'ascolto, l'attività più praticata è la fruizione di programmi radiofonici in inglese con una media di ben 4 ore al giorno. Segue la visione di programmi televisivi (trasmessi sia da canali televisivi inglesi ricevibili mediante satellite, sia da alcuni canali tedeschi che abitualmente trasmettono film in lingua originale) con una media di un'ora e mezzo al giorno. Come mostra la Figura 1, a concorrere a questa media giornaliera sono dati molto disomogenei: sebbene infatti vi siano studenti che non guardano mai la televisione (o al massimo un'ora al giorno), ve ne sono alcuni che ne fanno ampio uso; addirittura un soggetto dichiara di guardarla per 5 ore al giorno;

Un altro studio atto a capire quali siano le attività extra-scolastiche praticate dagli studenti per migliorare il loro inglese è stato condotto da Hyland[25] nel 2004. L'indagine è stata effettuata mediante un questionario sottoposto ad un campione di 238 studenti di un' Università di Hong Kong. I risultati del questionario hanno mostrato che la maggior parte dei soggetti predilige attività extra scolastiche individuali piuttosto che quelle che implicano un'interazione faccia a faccia. Le motivazioni date dagli studenti sono di

diversa natura e la tendenza è quella di mascherare ciò che, ad un'analisi più approfondita, è risultata la causa principale per questo tipo di scelta, ovvero il timore di poter essere giudicati. Avere dunque la possibilità di svolgere su attività individuali quali la fruizione di trasmissioni televisive può aiutare a familiarizzare con la lingua target anche nel caso in cui si preferisca non fare uso di tale lingua in pubblico, almeno nella prima fase dell'apprendimento. Hyland aggiunge poi che le attività individuali non implicano un minor livello di competenze acquisite, anzi; questi studenti hanno mostrato una grande padronanza della lingua e si sono dimostrati ottimi candidati come futuri insegnanti di inglese LS. Hyland conclude la sua indagine chiedendosi se la società non sia troppo focalizzata sull'importanza dell'utilizzo *pubblico* della lingua durante la fase di apprendimento.

Cavaliere[10], in *Motivazione e apprendimento dell'Italiano LS,* testimonia indirettamente un altro caso di apprendimento di una LS per mezzo della tv. Poiché gli studenti serbi sembrano essere particolarmente portati all'apprendimento delle lingue straniere, in questo saggio Cavaliere analizza quali possono essere gli aspetti e le motivazioni che portano a questa predisposizione. Lo studio è stato condotto su un campione di studenti dell'Università di Banja Luka (Bosnia ed Erzegovina). L'autore, nel corso della sua analisi, dichiara che tale predisposizione sarebbe favorita da alcuni fattori tra i quali rientra la messa in onda, da parte delle emittenti serbe, di film, serie tv e documentari in lingua originale (soprattutto in inglese). Rapportandosi con la televisione sin da piccoli, i discenti serbi acquisiscono facilmente la padronanza di una lingua straniera e la facilità e naturalezza di questo apprendimento li predispone emotivamente anche all'apprendimento di altre lingue (tra cui l'italiano). Secondo alcuni studi infatti sembra che chi abbia già appreso una prima LS si trovi avvantaggiato nell'apprendimento di una seconda.

Uno studio più recente è stato condotto da Wong e Nunan[46] su un campione di studenti di Hong Kong non ancora diplomati, aventi sviluppato diversi livelli di competenza dell'inglese (alcuni dimostrano un effettivo processo di acquisizione in atto, altri no). Facendo riferimento agli stili di apprendimento definiti da Willing nel 1994 [58], gli

Table 1.
Learning style preferences of more and less effective students (n = 110).

	Communicative	Authority-oriented	Analytical	Concrete
More effective	41	10	13	1
Less effective	11	12	4	3

Figura 2 Preferenze sugli stili di apprendimento

autori hanno deciso di sottoporre agli studenti un questionario tramite il quale poter esprimere quale stile di apprendimento preferissero. Nel 50% dei casi gli studenti che stavano effettivamente imparando la lingua hanno dichiarato di scegliere proprio l'approccio comunicativo (Figura 2). Wong e Nunan hanno ulteriormente approfondito l'indagine per cercare di evidenziare, all'interno delle strategie di apprendimento comunicativo, quali fossero le più significative. Queste (Figura 3), per quanto riguarda l'approccio comunicativo, sono risultate essere: osservare e ascoltare parlanti nativi, parlare in inglese con i compagni, guardare la televisione in inglese. Alla fine di questa indagine gli autori giungono alla conclusione che al fine di imparare in modo effettivo una lingua è importante assumere un atteggiamento interattivo, guardando film e leggendo giornali in lingua originale e osservando attentamente parlanti nativi. Aggiungono che non è necessario essere sistematici e, anzi, è preferibile cercare di assorbire la lingua mediante canali differenti.

Item	Strategy	Style
6	In English class, I like to learn by reading.	Authority-oriented
13	I like the teacher to explain everything to us.	Authority-oriented
18	I like to study English by myself (alone).	Analytical
24	I like to learn many new words.	XXX
29	At home, I like to learn by reading newspapers, etc.	Analytical
30	At home, I like to learn by watching TV in English.	Communicative
33	I like to learn by talking to friends in English.	Communicative
34	I like to learn by watching, listening to native speakers.	Communicative
35	I like to learn by using English outside class in stores etc.	Concrete

Figura 3 Attività di auto apprendimento più efficaci

Grazie a questi e a molti altri studi si può giungere alla conclusione che guardare la televisione in lingua originale sembra essere una delle favorite, e probabilmente una delle più efficaci, strategie di apprendimento spontaneo della lingua target.

1.2 La TV e l'apprendimento dell'italiano

"*Come tutti i ragazzi della sua età, Erman Rapushi aveva imparato l'italiano guardando la televisione*". È così che Alessandro Leogrande nel suo racconto *Il naufragio: morte nel Mediterraneo,* inizia il capitolo che tratta la storia di Erman, uno dei sopravvissuti al naufragio della motovedetta albanese del Marzo 1997, tema centrale di questo racconto che racchiude appunto le storie dei sopravvissuti intervistati dall'autore. Prosegue scrivendo: "*Nei caotici anni successivi alla caduta del regime comunista, i primi anni*

novanta, quella scatola magica posizionata al centro del piccolo salotto della loro casa aveva avuto il potere di rivelargli la più pura essenza della modernità. […]"[31].

Nel raccontare un triste avvenimento, fin troppo frequente anche ai giorni nostri, Leogrande fa riferimento anche ad uno dei fenomeni più significativi per quanto riguarda l'apprendimento spontaneo della lingua italiana tramite la televisione in alcuni paesi del Mediterraneo.

A partire dalla metà degli anni '70, con la liberalizzazione delle trasmissioni televisive via cavo, nacquero molti canali italiani privati e la loro ricezione raggiunse (e in parte raggiunge tutt'oggi) molti paesi del Mediterraneo. Anche se oggi gli ascolti dei canali italiani sono in calo, per alcuni di questi paesi la televisione ha rappresentato un forte veicolo di diffusione della lingua e i casi più esemplari sono quello dell'isola di Malta e quello dell'Albania.

1.2.1 Il caso di Malta

A partire dalla prima metà del 500 l'italiano ha giocato un ruolo molto importante nel panorama linguistico maltese. Ai tempi dell'Ordine di San Giovanni infatti (tra il 1530 e il 1798), dopo che l'italiano fu scelto come lingua ufficiale dei Cavalieri, iniziò a prendere piede lo studio delle grammatiche del Cinquecento. Conseguentemente l'italiano scritto utilizzato dai Cavalieri dell'Ordine si avvicinò notevolmente a quello prodotto in Italia e divenne la lingua della classe elevata. Successivamente alcuni eventi portarono ad un graduale abbandono della lingua italiana in favore di quella inglese. In primo luogo, con la colonizzazione britannica, molti maltesi iniziarono a introdurre nel loro repertorio linguistico la lingua inglese che lentamente rimpiazzò l'italiano come lingua della classe elevata. Altro fattore che ostacolò la sopravvivenza della nostra lingua fu il ruolo che Malta assunse durante la seconda guerra mondiale: l'isola fu impegnata a contribuire alla difesa dei paesi del Mediterraneo contro il regime fascista. Per contrastare quindi un regime quale quello irredentista che rivendicava l'italianità di Malta, l'uso dell'italiano scemò ulteriormente[44][52].

Negli ultimi decenni però, grazie ad alcuni fattori, si è verificata un'inversione di tendenza e tra questi fattori rientra l'influenza della televisione. La ricezione di canali televisivi italiani a Malta inizia negli anni '50[23] e a partire da questo momento la fruizione di programmi televisivi in lingua italiana divenne un fenomeno sempre più diffuso che riversò importanti conseguenze sulla diffusione della lingua italiana a Malta.

Il tema dell'acquisizione dell'italiano L2 a Malta è stato oggetto di vari studi negli ultimi anni e gli esiti di queste ricerche mostrano che la televisione, soprattutto verso la fine del '900, ha avuto un ruolo importantissimo nella diffusione della lingua. Bambini dai 5 agli 11 anni che non erano mai stati sottoposti all'insegnamento dell'italiano in contesti formali erano capaci di tradurre un buon numero di parole e alcune frasi dall'italiano al maltese o all'inglese[8]. Tali studi hanno inoltre mostrato che non si sono verificate differenze notevoli tra coloro che alla scuola media hanno scelto l'italiano come LS e coloro che hanno scelto un'altra lingua (quale il francese o il tedesco) e hanno invece iniziato ad apprendere l'italiano spontaneamente attraverso la fruizione di programmi televisivi.

Caruana[9] ha condotto uno studio su un campione di 61 liceali tra i 14 e i 15 anni, 27 dei quali studiavano italiano come seconda lingua e i restanti 34 la stavano apprendendo spontaneamente (attraverso diversi mezzi, tra cui la televisione). Il campione è stato sottoposto alla visione di un filmato di Charlie Chaplin lungo circa 20 minuti. Alla fine del filmato è stato chiesto ai ragazzi di raccontare in italiano quanto avvenuto. I risultati hanno mostrato che dei 34 soggetti apprendenti l'italiano spontaneamente, 26 sono riusciti ad elaborare una descrizione piuttosto dettagliata. Lo stesso risultato è stato raggiunto da 26 dei 27 studenti di italiano L2. Riguardo alla padronanza mostrata nell'utilizzo del sistema verbale, come mostra la Figura 4, non emergono differenze significative tra i due gruppi ma, guardando la Figura 5, è possibile notare che maggiore è la fruizione di programmi televisivi in italiano (indifferentemente da parte di apprendenti spontanei e non), maggiore è la padronanza nell'utilizzo dei verbi italiani durante la descrizione del filmato. Gli 8 dei 34 studenti

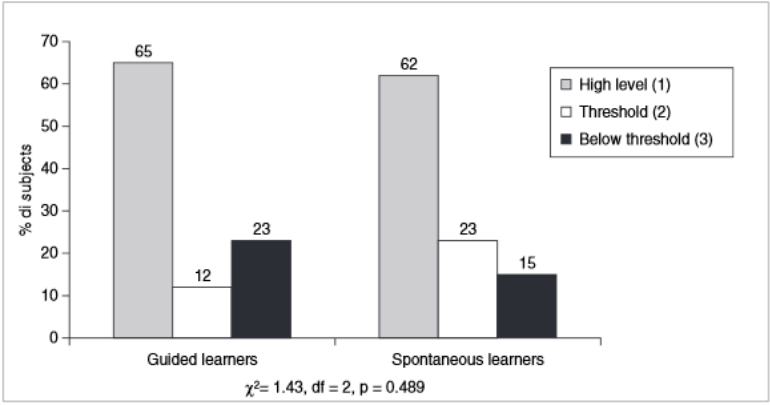

Figura 4 Padronanza del sistema verbale da parte dei due gruppi di apprendenti

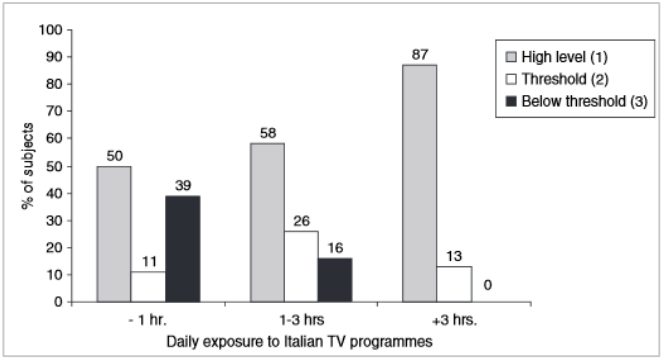

Figura 5 Competenza del sistema verbale in relazione alla fruizione di programmi televisivi italiani

apprendenti l'italiano in modo spontaneo che non sono riusciti a descrivere il filmato hanno infatti avuto un minimo input linguistico da parte della televisione.

A partire dalla fine degli anni ’90, sono entrati a far parte del palinsesto maltese molti canali locali trasmettenti nuovi programmi in lingua maltese e inglese. La logica conseguenza è stata una riduzione dell’audience per i canali televisivi italiani; si stima infatti che, se nel 1996 gli spettatori che seguivano abitualmente trasmissioni italiane erano circa il 40%, nel 2006 tale valore è sceso al 15%[9].

1.2.2 Il caso dell’Albania

In Albania, in modo particolare negli anni '90, i programmi televisivi italiani entravano nelle case di moltissimi cittadini. Si stima che oggi l’italiano venga compreso da circa l’80% della popolazione albanese e parlato, con più o meno padronanza, da circa il 50%. La richiesta di italiano a livello scolastico è però piuttosto bassa (solo il 5% lo sceglie come prima lingua straniera), indice del fatto che l’italiano si è diffuso attraverso qualche altro canale. Sembra infatti che la maggior parte di questo 50% sia costituita perlopiù da soggetti giovani, venuti a contatto con la lingua italiana in seguito alla caduta del regime comunista. Secondo l’intervista condotta da Longo nel 2007, su 133 giovani albanesi parlanti italiano (abitanti in centro città e dunque aventi una buona potenza del segnale televisivo), ben 124 hanno dichiarato di essere venuti a contatto con la lingua italiana proprio attraverso la televisione e solo 9 di essi attraverso corsi di lingua universitari[32].

Agli inizi degli anni 2000 però, così come Malta, anche l’Albania ha visto nascere nuovi canali e trasmissioni televisive locali, tra cui rifacimenti di vari programmi italiani (ad esempio in Albania vengono trasmesse le versioni albanesi di *Striscia la Notizia* e *Amici*). Inoltre l’accesso a molti canali italiani è divenuto possibile soltanto attraverso l’utilizzo di decoder o parabola. La più vasta offerta locale e il costo richiesto per la fruizione di canali televisivi italiani hanno ovviamente posto un freno agli ascolti.

1.2.3 Altri esempi

I casi di Malta e dell’Albania sono senz’altro i più lampanti. Vi sono però diversi altri casi, soprattutto nel Mediterraneo, che testimoniano la diffusione dell’italiano attraverso la televisione. Ad esempio, sembrerebbe che un buon ascolto di programmi televisivi in lingua italiana sia riscontrabile anche in altri tre paesi che si affacciano sul Mediterraneo: Tunisia, Marocco[52] e Grecia.

In Grecia tale fenomeno è testimoniato da Athanasia Drakouli e Georgia Milioni-Bertinelli, due insegnanti dell'Istituto Statale di avviamento professionale della scuola media statale di Iraklio (Creta). In una pubblicazione che tratta il tema dell'insegnamento della lingua italiana in Grecia, le due insegnanti si mostrano favorevoli ad un ripensamento dell'insegnamento della lingua italiana (introdotta come LS nel sistema didattico della scuola media a partire dall'anno accademico 2008/2009) nelle scuole greche. Tale ristrutturazione dovrebbe prevedere l'introduzione della tv satellitare nella prassi didattica dell'insegnamento scolastico dell'italiano LS. Le due insegnanti sostengono che gli apprendenti di lingua italiana che abbiano avuto occasione di fruire film in lingua originale trasmessi da un'emittente greca, ne parlino come un'esperienza estremamente piacevole[34]. Aggiungono che l'utilizzo di tale strumento renderebbe la didattica più avvincente ed emotivamente coinvolgente, predisponendo il discente ad imparare più volentieri, e forse più efficacemente, la lingua target. Tale strategia incrementerebbe inoltre la naturalezza del processo di apprendimento, rendendolo più simile a quello dell'acquisizione spontanea di una L2. Si avrebbe inoltre la possibilità di apprendere mediante un modello linguistico estremamente attuale, senza limitarsi al più rigido modello previsto dalla didattica tradizionale[16][56].

1.3 Acquisizione spontanea e risultati

In un approccio audiovisivo è possibile che il discente parta da una condizione di completa incompetenza nella lingua target e che dunque l'input ricevuto sia incomprensibile. Secondo Krashen[28] un input incomprensibile è sinonimo di rumore e, di conseguenza, l'acquisizione della lingua risulta essere nulla; se dunque nella prima fase dell'apprendimento non si possiede minima conoscenza della L2, per sperare di ottenere risultati è necessario che tale lingua sia molto simile alla propria lingua madre. Karshen riporta infatti un esempio citato da Larsen-Freeman[18] in cui un parlante tedesco ha imparato l'olandese interamente tramite la tv; data l'appartenenza delle due lingue al ceppo germanico, l'affinità è senz'altro abbastanza alta. Anche per il caso dell'italiano a Malta che si è trattato nella Sezione 1.2.1 è effettivamente vero che il maltese, anche se appartenente al ceppo semitico, presenta elementi propri delle lingue romanze (soprattutto per quanto riguarda il lessico, ma anche dal punto di vista della struttura sintattica[9]), acquisiti in particolar modo nel periodo dell'Ordine di San Giovanni; la

facilità con cui molti maltesi hanno appreso la lingua italiana potrebbe dunque essere spiegata anche da questa peculiarità[60].

Per poter intraprendere un processo di apprendimento spontaneo è però sufficiente una minima conoscenza della lingua target; l'uomo riesce infatti a comprendere il significato di una parola sconosciuta grazie al contesto entro il quale essa si inserisce (Contextual Vocabulary Acquisition -CVA-[39]). Il significato di tale parola viene presagito grazie alla sua contestualizzazione, escludendo qualsiasi altra fonte di informazione esterna come dizionari o persone. Vero è, che qui entra in gioco la conoscenza pregressa della lingua; ma è altresì vero che in un contesto audiovisivo le espressioni facciali, i gesti, il tono della voce e la prosodia degli speaker sono tutti elementi che possono agevolare l'apprendente nell'intuire il senso delle parole.

Per facilitare ulteriormente questo processo di CVA, un ruolo fondamentale è giocato dalla sottotitolazione.

1.4 L'importanza della sottotitolazione

La sottotitolazione è il processo mediante cui uno spettatore può leggere i dialoghi sullo schermo contemporaneamente al loro ascolto e alla visione del filmato. In un contenuto audiovisivo sottotitolato l'apprendente ha a disposizione dunque tre differenti canali di input: lo scritto, l'audio e le immagini. Il discente, nonostante debba porre maggior impegno per sentire, vedere e leggere contemporaneamente, può fare affidamento su tre fonti di informazione diverse per cercare di comprendere il contenuto (CVA) [51].

Si potrebbe pensare che la sottotitolazione distolga l'attenzione dalla lingua parlata e che impigrisca l'apprendente. Vi sono però vari studi che attestano che la sottotitolazione, al contrario, favorisce l'apprendimento della lingua [11][12][35].

Innanzi tutto è necessario distinguere tra due tipologie di sottotitolazione [36]:

- *Interlinguistica*: quando i sottotitoli sono tradotti in un'altra lingua. Questa tipologia di sottotitolazione, che richiede un complesso processo di traduzione (oltre che di trascrizione), è ovviamente utile in tutti quei casi in cui ci sia bisogno che il messaggio venga recepito da un interlocutore che non comprende la lingua originale. Può dunque essere utile anche per quanto riguarda l'aspetto dell'apprendimento, seguendo ad esempio il metodo detto *reversed subtitling*. In questo tipo di approccio i dialoghi sono doppiati nella lingua madre del discente, mentre i sottotitoli vengono mantenuti in lingua originale. Questa metodologia

risulta essere particolarmente utile nelle prime fasi di apprendimento di una lingua straniera[19].

- *Intra-linguistica*: quando i sottotitoli sono nella stessa lingua della traccia audio. In questo caso i sottotitoli sono utili come:
    - strumento per tutti coloro che abbiano deficit uditivo; in questo caso il sottotitolo si presenta solitamente ridotto rispetto a ciò che viene effettivamente detto nell'audio ed è arricchito da alcune informazioni quali "musica di sottofondo", "ride" e così via.
    - supporto didattico che favorisce l'apprendimento spontaneo di una lingua straniera; in questo caso il sottotitolo rispecchia fedelmente la traccia audio.

È possibile poi fare un'ulteriore distinzione tra sottotitoli:

- *verbatim*: la sottotitolazione rispecchia parola per parola la traccia audio (comprese interiezioni, ripetizioni ecc.);
- *nonverbatim*: la sottotitolazione non rispecchia fedelmente la traccia audio di cui vengono fatte parafrasi, rimosse le ripetizioni e così via.

Secondo Bravo[6] sia i sottotitoli inter-linguistici che quelli intra-linguistici favoriscono l'apprendimento di una lingua straniera. Gli apprendenti che non conoscono minimamente la lingua target saranno favoriti dall'approccio inter-linguistico. Coloro invece che abbiano anche un minimo di competenza della lingua target beneficeranno dall'approccio intra-linguistico. L'ausilio della sottotitolazione intra-linguistica infatti aiuterà il discente a riconoscere la forma grafica delle parole della lingua target e ad associare loro la corretta pronuncia. Caimi[7] aggiunge che questo tipo di sottotitolazione è estremamente utile per la memorizzazione del lessico.

Riguardo alla sottotitolazione inter-linguistica Ina[26] ha condotto un'indagine per capire quali siano gli effetti a breve termine della fruizione di programmi italiani sottotitolati e non, da parte di bambini greci tra i 9 e i 12 anni di età. L'esperimento è stato condotto suddividendo un campione di 93 studenti in tre gruppi (di 31 soggetti) che sono stati sottoposti per due volte alla visione di materiale audiovisivo della durata di 15 minuti. Il primo gruppo ha guardato il filmato in italiano con l'ausilio della sottotitolazione in greco, il secondo gruppo lo ha guardato senza la sottotitolazione e il terzo gruppo ha guardato lo stesso filmato doppiato in greco. Il materiale audiovisivo italiano presentato ai bambini

è stato selezionato affinché fosse piuttosto elementare sia dal punto di vista del contenuto, sia dal punto di vista linguistico (il parlante è una donna madrelingua che fa uso di un italiano pulito e ben scandito). La valutazione è stata condotta chiedendo agli studenti (confortati a priori sul fatto che il test non avrebbe influito sui loro voti scolastici!) di tradurre in greco 35 parole italiane (occorse nel video almeno quattro volte). L'esperimento ha dimostrato che il primo gruppo ha acquisito un maggior numero di vocaboli rispetto al secondo e, ovviamente rispetto al terzo.

Anche Koolstra e Beentjes[27] hanno condotto un test simile su 828 bambini della scuola elementare tedesca. L'indagine è stata condotta per a capire come la televisione e la sottotitolazione influissero sulle capacità di lettura degli studenti. Il test è durato tre anni, durante i quali i bambini sono stati suddivisi in tre gruppi e sottoposti alla visione rispettivamente di programmi televisivi in lingua inglese non sottotitolati, programmi in lingua inglese sottotitolati in tedesco e programmi doppiati in tedesco. Trascorsi i tre anni i risultati hanno mostrato che i bambini che avevano acquisito un migliore vocabolario sono stati quelli del secondo gruppo, che hanno guardato filmati inglesi sottotitolati in tedesco.

Dunque guardare filmati con sottotitolazione inter-linguistica, rispetto a guardare filmati non sottotitolati, favorisce in tali circostanze l'apprendimento della lingua.

Riguardo alla sottotitolazione intra-linguistica, Baltova[3] ha condotto un esperimento su 93 studenti canadesi che studiano il francese come LS. Gli studenti sono stati suddivisi in tre gruppi e sono stati sottoposti alla visione di un filmato di sette minuti e mezzo. Il primo gruppo ha guardato il filmato in lingua inglese con sottotitoli in francese, il secondo gruppo lo ha guardato in lingua francese con sottotitoli in francese e il terzo gruppo ha guardato il filmato in francese senza sottotitoli. I soggetti sono poi stati sottoposti ad alcune domande per verificare l'acquisizione del vocabolario: i risultati dell'esperimento indicano che il gruppo in cui si è riscontrato un maggior apprendimento è il secondo.

Zarei[48] ha condotto un esperimento su 120 studenti iraniani aventi una conoscenza base dell'inglese (i soggetti sono stati preventivamente selezionati in modo che avessero tutti lo stesso livello di competenza). I soggetti sono stati sottoposti alla visione di un filmato della durata di 70 minuti (*She's the man*). I partecipanti sono stati suddivisi casualmente in quattro gruppi di 30 soggetti e ogni gruppo ha guardato il filmato in modalità diverse:

- gruppo 1: sottotitolazione inter-linguistica verbatim;

- gruppo2: sottotitolazione inter-linguistica nonverbatim;
- gruppo3: sottotitolazione intra-linguistica verbatim;
- gruppo 4: sottotitolazione intra-linguistica nonverbatim.

Dopo la visione del filmati i soggetti sono stati sottoposti a due test di verifica: il primo, atto a verificare la comprensione del vocabolario, il secondo mirato ad esaminare la competenza espositiva. I risultati di entrambi i test hanno dimostrato che il gruppo che ha ottenuto un punteggio medio maggiore è il terzo, ovvero quello sottoposto alla visione del filmato con sottotitolazione intra-linguistica verbatim.

Il tipo di sottotitolazione utilizzato in SubItalian, poiché realizzato completamente in modo automatico, è proprio quello **intra-linguistico verbatim**.

2 L’e-learning, l’approccio CALL e SubItalian

Nel capitolo precedente si è dimostrato come un approccio audiovisivo possa effettivamente essere determinante per l’apprendimento spontaneo della lingua italiana. A partire da questa osservazione si potrebbe pensare di spostare gli ascoltatori su un altro *canale*, perlopiù gratuito, potenzialmente accessibile a tutti[2] e, così come la televisione, fonte –anche- di svago: il web.

Il principale scopo di SubItalian è quindi quello di trasportare in un mondo 2.0 la fruizione dei contenuti audiovisivi sottotitolati e le loro potenzialità in termini di apprendimento di una lingua, in questo caso la lingua italiana, fornendo così un innovativo strumento di Computer Assisted Language Learning.

2.1 L’e-learning

Con il termine *e-learning* si intende l’integrazione della rete e delle tecnologie multimediali all’interno dei processi di apprendimento al fine di migliorarne qualità e tempistiche, di facilitare l’accesso alle risorse e favorire l’interazione e la collaborazione da remoto. Scopo dell’e-learning è quello di offrire contenuti via web svincolati dal tempo e dallo spazio rispetto alla formazione tradizionale; l'utente dunque sceglie come e quando approcciarsi all'apprendimento. I requisiti di una generica applicazione di e-learning sono [20]:

a. utilizzo di una connessione internet;
b. nessuna restrizione spaziale e poche restrizioni temporali;
c. monitoraggio del livello di apprendimento;
d. utilizzo della multimedialità;
e. interattività con docenti, tutor e altri apprendenti;
f. valorizzazione della dimensione sociale.

Tali strategie si rivelano di estrema importanza sia come supporto all’insegnamento in presenza (*Blended learning*), sia per la Formazione a Distanza (FAD). Per quanto riguarda il primo aspetto, è ormai appurato come strumenti tecnologici possano costituire un valido aiuto in vari ambiti didattici, tant’è che i Piani d’Azione *eEurope 2002* e *eEurope 2005*, deliberati dalla Commissione Europea, sostengono azioni a favore dell’innovazione tecnologica nelle scuole.

[2] In realtà come si è visto nella sezione X sono ancora molti gli ostacoli che impediscono una fruizione trasversale di internet.

Per quanto riguarda il secondo aspetto, è altrettanto ovvio che la tecnologia rappresenti un enorme passo a favore della Formazione a Distanza. Questo approccio didattico permette infatti di accedere all'*apprendimento* quando:

- la fonte di informazione e l'apprendente sono separati nel tempo e/o nello spazio;
- l'apprendente, o la sua famiglia, non può sostenerne la spesa di un tradizionale corso universitario;
- l'apprendente non può presenziare ai corsi a causa di disabilità o altre malattie che impediscano un'agevole mobilità.

La FAD vede le sue origini alla fine del 1800 quando si sviluppa quella che viene chiamata la prima generazione della formazione a distanza. Questa era basata sulla corrispondenza postale che in quegli anni divenne più accessibile grazie alle nuove tecniche di stampa e allo sviluppo del trasporto ferroviario che agevolò lo scambio di materiale. I docenti inoltravano al discente materiale didattico cartaceo accompagnato da istruzioni e consigli sulla modalità di studio da attuare. Seguiva una fase di verifica che solitamente consisteva nello svolgimento di un test scritto che l'apprendente inviava al tutor da cui riceveva la correzione. In questa prima generazione la comunicazione è di tipo *uno a uno* e non vi è alcun tipo di interazione tra discenti.

La FAD vede una prima svolta con l'invenzione del primo mezzo di comunicazione di massa: la radio. Si entra così in quella che è la seconda generazione della formazione a distanza. L'approccio della didattica a distanza si trasforma e passa da essere *uno ad uno* a *uno a molti*. Il discente può ricorrere all'assistenza telefonica ma permane comunque l'utilizzo dello scritto per quanto riguarda la fase della verifica che continua ad essere svolta tramite corrispondenza.

Ulteriore e decisivo passo avanti nella FAD è rappresentato dall'introduzione della televisione negli anni '50. Il nuovo mass media ha rappresentato un importante mezzo per la diffusione della cultura (come si è mostrato nel Capitolo 1 riguardo alla diffusione della lingua), facilmente accessibile e comprensibile anche ai meno scolarizzati. Come nella prima generazione permane comunque l'assenza di un'interazione tra discenti e dunque l'aspetto sociale.

La terza generazione, infine, è segnata dalla nascita del web e dall'utilizzo di tutte le tecnologie a esso connesse; si può quindi iniziare a parlare di *e-learning*. In questa fase

la figura del discente si trasforma, assumendo un ruolo decisamente meno passivo e incentrato sulla dinamicità e interattività tra pari[42].

Grazie all'utilizzo di Internet la formazione a distanza è stata indubbiamente resa più accessibile, veloce ed efficace e l'e-learning, inteso appunto come FAD, ha apportato numerosi vantaggi, tra cui[29]:

- riduzione dei costi di formazione (tasse, trasporti, libri di testo);
- controllo sulle fasi di apprendimento: il discente può scegliere quando e su cosa focalizzarsi maggiormente, soffermandosi sugli aspetti su cui è meno preparato e tralasciando quelli su cui è più ferrato;
- accesso al materiale in qualsiasi momento (nel caso di un apprendimento asincrono).

2.2 CALL (Computer Assisted Language Learning)

Un concetto affine all'e-learning, ma specifico per l'apprendimento di una lingua, è quello di Computer Assisted Language Learning (CALL). Per CALL si intende una strategia di apprendimento che prevede l'utilizzo di tecnologie e strumenti informatici che facilitino il processo di apprendimento di una lingua. Questo approccio sostituisce il precedente CALI (Computer Assisted Language Instruction), introdotto per la prima volta negli anni '60 all'Università dell'Illinois. Questa prima elaborazione, come il nome suggerisce, vedeva primario il ruolo dell'insegnante; l'etichetta CALL è stata introdotta (negli anni '80) proprio per evidenziare che, invece, tale filosofia si basa su un approccio comunicativo (v. 1.1) *student-based,* che pone dunque il focus sull'apprendimento *learning*) piuttosto che sull'insegnamento (*instruction*).

Due fondamentali aspetti di questa strategia sono quindi:

- apprendimento individuale;
- interattività.

Sebbene le possibilità offerte dai primi computer fossero alquanto limitate (le macchine non erano ad esempio in grado di riprodurre file audio né video), con l'evoluzione delle tecnologie e, in particolar modo, con la nascita e la diffusione dei personal computer, si sono aperti nuovi orizzonti che hanno permesso di giungere a quello che oggi viene chiamato *multimedia CALL*. A partire dagli anni '80, ad esempio, sono nati i primi CD-

ROM (sostituiti poi dai DVD) per l'apprendimento linguistico[40][14] orientato all'interattività e alla partecipazione attiva dell'apprendente.

I primi anni '90 hanno poi visto la nascita di ciò che avrebbe rappresentato, anche per la didattica, una svolta decisamente significativa: il web. Con lo sviluppo e la diffusione del web le potenzialità del CALL sono aumentate esponenzialmente. Il concetto dell'ipertesto informatico, per quanto agli occhi dei navigatori di oggi possa sembrare scontato, ha rappresentato una vera e propria rivoluzione; anche in ambito didattico. Nel web sono organizzate informazioni espresse in linguaggi di natura estremamente varia (testi, immagini, video, simboli e così via) e tale varietà coinvolge l'apprendente stimolandone diverse strategie di pensiero.

Il continuo evolversi delle tecnologie informatiche ha accresciuto sempre più l'introduzione dell'interattività all'interno dei processi di apprendimento di una lingua, che oggi sicuramente sconfinano da quelli che sono i classici modelli didattici. Tra la molteplicità di applicazioni CALL si possono citare:

- podcast: i siti dedicati alla creazione e alla fruizione di podcast permettono la diffusione di contenuti mediante feed RSS che notificano l'utente registrato ogni qualvolta vengano caricati nuovi contenuti;
- giochi online: tutti i giochi in cui ci sia bisogno di comunicare con il computer o con gli altri giocatori (sia attraverso input testuale che tramite input vocale) utilizzando la lingua target rappresentano uno strumento ricco di potenzialità in termini di apprendimento;
- blog, chat online e social network: un blog, una chat o qualsiasi social network che permetta di comunicare con parlanti nella lingua target, può senz'altro favorire lo sviluppo delle competenze di lettura e scrittura;
- rivisitazione di esercizi standard: gli strumenti informatici possono essere utilizzati per realizzare facilmente alcune tipologie di esercizio standard come ad esempio il cloze (v. a.7.1) o i cruciverba;
- contenuti audio nei dizionari di lingua: l'introduzione del supporto uditivo nei dizionari di lingua online (http://www.wordreference.com/enit/hello) permette di ascoltare la corretta pronuncia della parola straniera enunciata da un parlante nativo;
- sistemi ASR: sistemi di trascrizione automatica possono essere utilizzati per realizzare esercitazioni mirate al miglioramento della pronuncia dell'apprendente.

2.3 SubItalian

SubItalian consiste in una piattaforma web ispirata ai concetti di e-learning e di CALL. Tale strumento è pensato per tutti coloro che vogliano apprendere l'italiano, o migliorare le proprie competenze, guardando filmati in lingua italiana. SubItalian sfrutta le potenzialità del web per agevolare il processo di apprendimento favorendo non solo la familiarizzazione con la forma orale, ma anche con quella scritta. Grazie alla trascrizione e sottotitolazione automatica infatti, SubItalian offre un supporto testuale utile per due ragioni:

- agevola la comprensione della lingua parlata;
- permette di sviluppare competenze che riguardano aspetti della forma scritta.

Un ulteriore vantaggio di SubItalian consiste nella possibilità di approfondire ulteriormente la scoperta della lingua italiana fornendo l'annotazione linguistica (automatica) dei sottotitoli e della trascrizione dei video. Affinché tale funzionalità possa essere ritenuta effettivamente utile sono però necessarie due condizioni:

1. il discente deve aver già acquisito una buona padronanza della lingua italiana;
2. la lingua del discente deve prevedere una struttura grammaticale simile a quella dell'italiano.

In SubItalian la sottotitolazione viene realizzata a partire dalla trascrizione, generata processando i video con un motore di trascrizione automatica (v. 3.2.1); la trascrizione viene poi annotata automaticamente da un motore di annotazione linguistica (v. 3.2.2) e processata da un programma Java (v. 3.3) che permette di ottenere il sottotitolo arricchito di metadati linguistici.

Benché, come si è visto nel capitolo precedente, la fruizione di contenuti audiovisivi sia un approccio che rientra perlopiù in una modalità di apprendimento individuale, SubItalian prevede la collaborazione di utenti madrelingua che favoriscano il corretto apprendimento. SubItalian gestisce dunque due tipologie di utenza: gli utenti *learner* (v. 4.1.1) e gli utenti *tutor* (v. 4.1.2). Gli utenti *tutor* possono ovviare agli inevitabili limiti dei motori di trascrizione e annotazione automatica (v. 9) tramite una funzionalità di revisione che permette loro di modificare in modo semplice e intuitivo i risultati del processing automatico (v. 4.6).

Mentre nei tradizionali metodi di insegnamento è solitamente prevista un'attività di ascolto mirata all'esercitazione (scelte multiple, griglie e così via), l'attività di ascolto

(accompagnata dal supporto testuale) di SubItalian è invece prevalentemente mirata all'apprendimento. Nonostante questo per un apprendente è importante, e magari motivante, poter mettere alla prova le proprie competenze; per questo motivo è stata sviluppata una sezione dedicata alle esercitazioni. Agli utenti *tutor* viene messo a disposizione un sistema tramite cui possono facilmente preparare delle esercitazioni sulla base delle trascrizioni e annotazioni automatiche (v. 4.7.3); gli utenti *learner* hanno invece a disposizione una sezione per lo svolgimento di tali esercitazioni (v. 4.7.4).

3 Descrizione dell'architettura e delle tecnologie utilizzate

Come è possibile constatare dallo schema mostrato in Figura 6, raffigurante l'intera architettura dei SubItalian, vi sono quattro macro componenti individuabili:

- l'applicazione web (Figura 6.a);
- il database (Figura 6.b) ;
- il software TASEe (Trascrizione, Annotazione, Sottotitolazione, and Editing engine) (Figura 6.c) ;
- i componenti Synthema (v. 3.2) che forniscono i servizi di trascrizione e annotazione (Figura 6.d) .

Nelle sezioni che seguono, tali componenti vengono descritti in modo dettagliato.

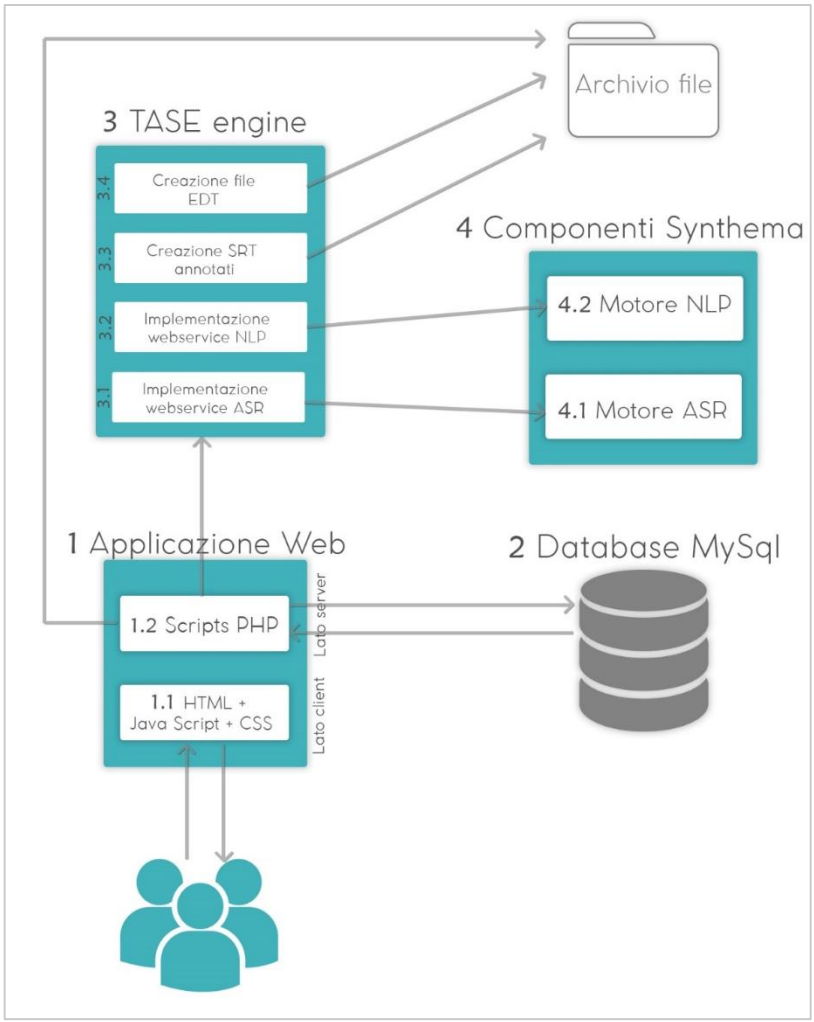

Figura 6 Architettura SubItalian

3.1 L'applicazione Web e il database

L'applicazione web è un'architettura client-server che da una parte si interfaccia con l'utente finale permettendogli di interagire con l'applicazione (programmazione lato

client), dall'altra gestisce lo scambio di dati (programmazione lato server) con due degli altri tre macro componenti: il database e il software TASEe. L'applicazione risiede su una macchina Windows messa a disposizione da Synthema (v. 3.2), accessibile dalla rete esterna. Su tale macchina è installata la piattaforma Wamp che include tre componenti fondamentali per lo sviluppo web: Apache HTTP Server, che contiene la root dell'applicazione e permette di gestire la richiesta di pagine web da parte del client, MySQL e PHP. È da questi tre componenti che infatti deriva il nome della piattaforma: (**W**indows) **A**pache **M**ySQL **P**HP.

L'applicazione segue il paradigma del *web dinamico*, secondo cui le informazioni da presentare all'utente vengono processate mediante linguaggi di scripting lato server il cui comportamento è dettato dalle azioni dell'utente. Tale concetto si differenzia da quello di *web statico* che invece prevede una comunicazione quasi esclusivamente unilaterale per cui l'utente si limita a visualizzare i contenuti generati dal sito. Proprio perché basata su questo paradigma, l'applicazione web è strutturata secondo un'architettura *three-tier* (Figura 7) che consiste nella stratificazione logica dell'applicazione in tre diversi livelli, chiamati: *presentation tier* (o logica di presentazione), *application tier* (o logica funzionale) e *data tier* (o gestione dei dati persistenti). Si vedano nel dettaglio questi tre livelli e le tecnologie utilizzate all'interno di ognuno di essi.

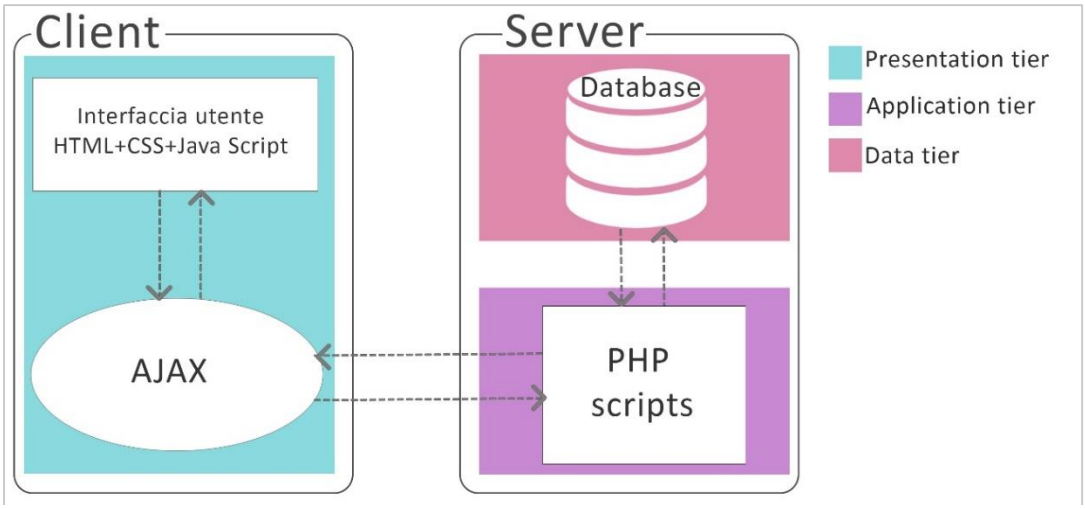

Figura 7 Architettura three-tier

3.1.1 *Presentation tier*

Il *presentation layer* è il livello dedicato al *front-end* dell'applicazione ovvero l'interfaccia utente. Questo livello comprende quindi tutte le tecnologie utilizzate per la visualizzazione dei contenuti elaborati dal browser del client (e dunque visualizzati dall'utente), le tecnologie per la gestione della manipolazione dei dati da parte dell'utente

e le tecnologie utilizzate per inviare i dati generati dal presentation tier all'application tier (v. 3.1.2). I linguaggi di programmazione lato client utilizzati all'interno di questo livello sono quindi:

- HTML per la definizione della struttura delle pagine e la realizzazione dei contenuti statici (ad esempio il menu di navigazione, uguale per tutte la pagine);
- Java Script per la gestione dell'interattività, degli eventi e dei contenuti dinamici;
- il framework JQuery per la semplificazione di alcune funzioni di Java Script come, ad esempio, le selezione di elementi DOM[3] delle pagine HTML e la gestione degli eventi a loro agganciati;
- la tecnologia AJAX (Asynchronous JavaScript and XML) per lo scambio di dati tra il browser e il server tramite richieste HTTP di tipo GET o POST;
- le librerie HighCharts e HighStock per la realizzazione dei grafici (v. 4.8);
- la libreria JWPlayer per l'inserimento del player all'interno delle pagine HTML (v. 4.5, 4.6, 4.7.4.1);
- CSS per la definizione della formattazione degli elementi HTML.

3.1.2 *Application tier*

Questo livello è dedicato all'elaborazione dei dati passati dal livello precedente (v. 3.1.1) e rappresenta dunque il *back-end* dell'applicazione. Per poter gestire ed elaborare i dati c'è bisogno di un linguaggio di programmazione lato server (ASP.NET, PHP, JSP o un CMS) che possa comunicare con una base di dati. Il linguaggio di programmazione lato server utilizzato dall'applicazione di SubItalian è PHP v5.5.12. Sono stati implementati diversi script PHP ognuno dei quali è incaricato di comunicare con uno o più dei seguenti componenti:

  - il browser (front-end) da cui riceve richieste HTTP di tipo GET o POST generate dagli script lato client mediante AJAX, a cui risponde passando i dati opportuni;
  - TASEe (v. 7); alcuni script si occupano di invocare il motore TASEe e controllarne lo stato di avanzamento;

- il database (v. 6), che compone il *data layer* (v. 3.1.3), a cui formula query mediante l'utilizzo della libreria MySQLi, disponibile a partire dalla versione 5.0.0 di PHP;

Porzioni di codice PHP sono poi incluse all'interno delle pagine HTML per la generazione di contenuti dinamici da integrare nel DOM; per questo tutte le pagine che compongono il portale hanno in realtà estensione *.php*. Per non far confusione con gli script PHP che costituiscono questo livello (application tier), ci riferiremo comunque a queste pagine come *pagine HTML*. Ad esempio, come è possibile osservare dalla Figura 8, ogni pagina del portale contiene del codice PHP che recupera lo username dell'utente loggato da una variabile di sessione (v. 5.3) e lo stampa all'interno di un elemento *span*.

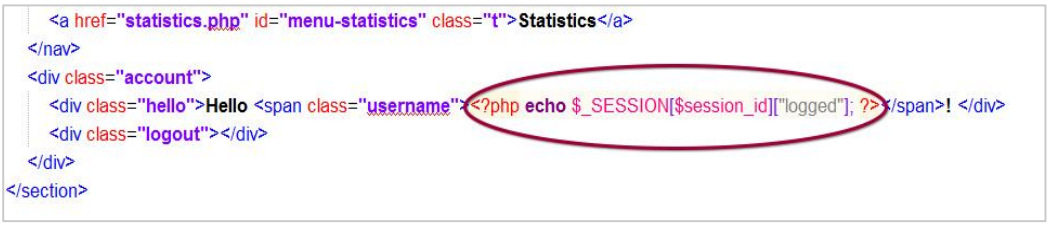

Figura 8 Esempio di codice PHP iniettato in una pagina HTML

Un ulteriore componente che viene utilizzato in questo livello è *Sendmail*[4]. Si tratta di un *mail transfer agent* incluso nella piattaforma Wamp che permette al PHP di inviare messaggi di posta tramite il protocollo SMTP. Come si vedrà più avanti il suo utilizzo sarà necessario, ad esempio, per effettuare la registrazione degli utenti che dovranno confermare l'iscrizione mediante un link mandato, appunto, via mail.

Inoltre, viene utilizzata l'applicazione *https://www.translate.com/* per permettere agli utenti learner di tradurre nella loro lingua madre parole italiane di cui non riescono a comprendere il significato (v. 4.5).

3.1.3 *Data tier*

Questo livello consiste in un DBMS (Database Management System). Il database utilizzato in questa applicazione è MySQL che segue il modello relazione. Questo livello è dunque destinato alla memorizzazione dei dati persistenti e può essere interrogato dall'*application tier* per leggere, scrivere o modificare dati presenti nelle tabelle. La struttura del database verrà trattata nel Capitolo 6.

[4] http://www.sendmail.com/sm/open_source/

3.2 Componenti Synthema

All'interno del workflow di SubItalian, sono stati integrati due componenti a cui ho avuto modo di accedere e, in parte, ho avuto modo di collaborare, grazie all'attuale impiego presso l'azienda Synthema.

L'azienda, situata a Pisa e fondata nel 1994 da alcuni informatici precedentemente impiegati all'IBM, è una SME (Small Medium Enterprise) che si occupa di sviluppare soluzioni informatiche e servizi di NLP (Natural Language Processing). A partire dal 1999, Synthema è coinvolta in un'attività di Ricerca e Sviluppo che l'ha resa partecipe di molteplici progetti commissionati dalla Comunità Europea all'interno dei quali ha avuto modo di collaborare con altri importanti centri e compagnie di ricerca (tra cui l'IBM, Nuance, VoiceInteraction e così via). All'interno di due di questi progetti sono stati rilasciati i due strumenti integrati in questo progetto: il servizio di trascrizione *SpeechScribe* e il servizio di annotazione *OpeNER*.

3.2.1 SpeechScribe ASR

Per poter trascrivere automaticamente l'audio dei file caricati dagli utenti, è stato integrato un componente di ASR (Automatic Speech Recognition) messo a punto nel corso di SAVAS (Sharing AudioVisual language resources for Automatic Subtitling), progetto commissionato dalla comunità Europea, iniziato nel 2007 e concluso nel 2012, di cui Synthema è stata partner. SpeehScribe[5], questo è il nome della soluzione ASR, è un prodotto/servizio che può essere utilizzato mediante un'apposita interfaccia web oppure può essere integrato nel proprio workflow mediante dei web service che, dato un file audio/video in input, ne restituiscono la trascrizione.

3.2.1.1 SpeechScribe e altri ASR

Uno degli aspetti che rende notevole questo motore di trascrizione è il fatto che sia *speaker independent.* Molti ASR (ViaVoice[6] dell'IBM, Microsoft Windows Speech Recognition[7] - oggi non più disponibile sul mercato-, Nuance Dragon Naturally Speaking[8]) sono stati pensati come sistemi di dettatura e respeaking[9] e sono dipendenti

[5] http://www.synthema.it/index.php/Prodotti/speechscribe/SpeechScribe.HTML

[6] http://www-01.ibm.com/software/pervasive/viavoice.html
[7] http://windows.microsoft.com/en-us/windows-10/getstarted-use-speech-recognition
[8] http://www.nuance.com/index.htm
[9] *Respeaking*: tecnica di speech recognition che presuppone la presenza di un operatore che detta al sistema ciò che viene pronunciato nell'audio originale ed è spesso affiancato da un secondo operatore che corregge in tempo reale l'output del sistema.

dallo *speaker*. Tali sistemi dunque, prima di poter essere utilizzati da un certo parlante, necessitano di una fase di addestramento in cui lo stesso parlante, per farsi *conoscere*, deve dettare al sistema un piccolo testo. Le performance di questi motori nel parlato spontaneo e/o in un contesto che presenti rumori o musica di sottofondo sono quindi piuttosto basse; ovviamente, il materiale con cui viene addestrato il sistema pone dei forti vincoli sul suo futuro utilizzo, sia in termini di dominio linguistico, sia in termini di dizione che in termini di condizioni acustiche. Per questo SpeechScribe è stato addestrato su registrazioni di notizie di telegiornali, contenenti una grande varietà di speaker e di condizioni acustiche. Altri motori ASR *speaker independent* sono Koemei[10], SailLabs[11], Vecsys[12], Verbio[13] e il motore di trascrizione di Google, recentemente integrato nella piattaforma YouTube per la sottotitolazione dei video caricati dagli utenti; quest'ultimo è l'unico tra i motori citati a produrre anche sottotitolazione. A differenza di SpeechScribe però, tale motore non produce punteggiatura né *capitalization*. In Figura 10 è mostrato il confronto tra i risultati, in termini di Word Error Rate (WER), di SpeechScribe e dell'ASR di Youtube, valutati per spagnolo, italiano, francese e tedesco su 2 ore (2 ore e mezzo per l'italiano) di registrazioni di telegiornali. È possibile constatare dalla tabella che i risultati di SpeechScribe sono migliori e per l'italiano si riscontra un WER del 17,81% contro il 27,80% dell'ASR di YouTube. Si deve però tenere in considerazione che il dominio su cui è stato eseguito il test è lo stesso con cui SpeechScribe è stato addestrato[2].

Language	Duration	Youtube	S.Scribe!
Spanish	2 H	24.91 %	16.50 %
Italian	2.5 H	27.80 %	17.81 %
French	2 H	37.61 %	25.81 %
German	2 H	31.10 %	26.92 %

Figura 9 WER YouTube VS SpeechScribe

3.2.1.2 I componenti dell'ASR

Il sistema è costituito da una pipeline di componenti:

[10] https://koemei.com/
[11] http://www.sail-labs.com/
[12] http://www.vecsys-technologies.fr/en
[13] https://www.verbio.com/

1. ***Audio Pre-Processing block***: individua le sezioni di parlato e di silenzio, identifica i cambi di speaker e il loro sesso e classifica le condizioni acustiche di sottofondo;
2. ***LVCSR block*** (Large Vocabulary Continuous Speech Recognition): trascrive le porzioni di parlato inviategli dal componente precedente facendo uso di:

   a) Un modello acustico

   Si tratta di un *file* contenente la rappresentazione statistica della combinazione dei foni all'interno di una parola per una certa lingua. Il modello acustico dell'italiano è stato creato a partire da un corpus di 200 ore di registrazioni di telegiornali RAI trascritte manualmente da annotatori aventi un certo grado di esperienza. A partire da questo corpus sono state estratte, attraverso algoritmi di training, le rappresentazioni statistiche di ogni fonema della lingua italiana. Il modello acustico viene consultato dal motore ASR al momento della trascrizione: per ogni fono che il motore riconosce all'interno di una sequenza di suoni, viene cercato il corrispondente fonema all'interno del modello. Il motore procede fintanto che non recepisce una pausa nel parlato; a questo punto recupera la serie di fonemi che ha trovato prima della pausa e cerca una corrispondenza nel dizionario.

   b) Un modello della lingua

   SpeechScribe fa uso di un modello linguistico che contiene informazioni riguardo al modo in cui le parole si possono combinare all'interno di una certa frase per una certa lingua. Più precisamente, ad una certa sequenza di parole è associato un valore che indica la probabilità che tale sequenza di parole occorra all'interno di una produzione orale nella lingua italiana. Il modello linguistico è fondamentale per disambiguare parole che suonino in modo simile. Se infatti ad una certa sequenza di fonemi è associata una parola facente parte di una produzione di parole a cui nel modello è associata una probabilità molto bassa o, a maggior ragione, tale produzione non fa parte del modello, tale sequenza di fonemi è probabilmente attribuibile ad un'altra parola che,

collocata all'interno di tale produzione ha una più alta probabilità nel modello del linguaggio. Il modello della lingua di SpeechScribe è stato creato a partire da un corpus di 1 miliardo di parole realizzato crawlando giornalmente una vasta quantità di materiale da internet. Prima di poter essere processati, i testi raccolti sono stati opportunamente normalizzati; per il motore di ASR era necessario ad esempio che date, numeri, valori e abbreviazioni fossero scritte per esteso.

c) ***Un dizionario***

A partire dallo stesso corpus utilizzato per la creazione del modello del linguaggio, è stato creato il vocabolario. Si tratta di un file contenente la lista di parole che il sistema è in grado di riconoscere (grafemi) con la relativa trascrizione fonetica.

3. ***Output Normalization block***: rende maiuscoli gli opportuni caratteri (*capitaliszation*) e aggiunge i segni di punteggiatura.
4. ***Subtitling generation block***: crea i sottotitoli (tale componente non è incluso in SubItalian poiché i sottotitoli necessitano di essere creati in seguito all'annotazione).

3.2.1.3 Il webservice

Per poter utilizzare l'ASR all'interno del workflow di SubItalian, SpeechScribe è stato integrato mediante i web service. Le operazioni che i web service permettono di effettuare per ottenere la trascrizione di file audio/video sono:

- specificare il file che il motore dovrà processare;
- specificare la lingua di tale file (le lingue attualmente supportate dal motore sono: inglese, italiano, francese, tedesco, italiano svizzero, francese svizzero, tedesco svizzero, portoghese, basco e spagnolo);
- specificare il modello acustico con cui si desidera effettuare la trascrizione (attualmente per l'italiano non è stato realizzato alcun modello acustico oltre a quello base ma per il portoghese, ad esempio, è stato realizzato un modello acustico per il telefonico);
- specificare il modello linguistico con cui si desidera effettuare la trascrizione (attualmente, oltre al modello linguistico di base, è stato realizzato un modello linguistico per l'italiano parlamentare);

- specificare una soglia di confidenza di riconoscimento minima sotto la quale le parole riconosciute non vengono aggiunte alla trascrizione;
- specificare il formato di output che si desidera; i formati previsti per la trascrizione sono:
    - TXT
    - DOCX
    - SSNTXML
    - PDF
    - SRT[14]

3.2.2 OpeNER NLP

La soluzione integrata in SubItalian per l'annotazione linguistica delle trascrizioni è il servizio rilasciato in OpeNER (Open Polarity Enhanced Named Entity Recognition), progetto commissionato dalla Comunità Europea, partito nel 2012 e conclusosi nel 2014, di cui Synthema è stata partner. L'obiettivo di OpeNER è stato proprio quello di fornire un servizio gratuito di NLP (Natural Language Processing) costituito da una serie di moduli linguistici, basati su Apache OpenNLP. Questi moduli sono: Language Detector, Tokenizator, Part-of-speech Tagger, Constituent parser, Named Entity Resolution, Polarity Tagger, Opinion Detector. Vi sono due moduli di base, tokenizer e POS tagger (e language-identifier se non si conosce a priori la lingua dal testo in input) che sono obbligatori per poter chiamare qualsiasi altro componente successivo; tutti gli altri moduli possono essere concatenati a piacimento (v. 7.1.3). OpeNER è disponibile in sei lingue europee: italiano, olandese, inglese, francese, tedesco e spagnolo.

3.2.2.1 Moduli di annotazione in SubItalian

In SubItalian i moduli integrati nel processo di annotazione sono proprio i due componenti di base: *tokenizer* e *POS tagger*.

- ***Tokenizer***

  La tokenizzazione è il primo passo di qualsiasi analisi computazionale ed è il processo mediante il quale uno stream di parole viene suddiviso in unità costituite da caratteri indivisibili; queste unità sono chiamate token. Un token può essere rappresentato da

[14] L'SRT (SubRip) è il formato di sottotitolazione utilizzato in SubItalian, tuttavia, poiché nel nostro caso prima di poter mostrare i sottotitoli è necessario formattarli con l'apposito markup al fine di attribuire colori diversi alle parole annotate, il formato di output che si richiederà al web service è SSNTXML. L'SRT verrà costruito a posteriori in seguito all'analisi linguistica (v. 7.1.5).

una parola, una data, un numero o un segno di punteggiatura. A differenza delle lingue a sistema ortografico continuo (cinese mandarino, giapponese), per le lingue che utilizzano gli spazi per delimitare le parole, la tokenizzazione è un task piuttosto semplice. Nonostante questo è necessario che il tokenizzatore gestisca delle eccezioni come ad esempio i segni di punteggiatura che non prevedono spazi dopo la parola che li precede[53]. Una corretta tokenizzazione è fondamentale per il corretto funzionamento delle analisi successive. In OpeNER questo modulo è stato addestrato per ogni lingua a seconda delle rispettive caratteristiche linguistiche[50].

- ***Il POS tagger***

Questo modulo si occupa di assegnare ad ogni token la sua categoria grammaticale ed è basato sul modello probabilistico di OpeNLP e addestrato per ogni lingua con il relativo modello del linguaggio. Questo modulo include inoltre un processo di lemmatizzazione che si basa su un dizionario.
Il tagset del POS tagger di OpeNER si basa sul tagset di annotazione TUT-Penn[15].

TUT è un corpus italiano sviluppato dal gruppo di NLP dell'Università di Torino. Dal TUT nasce il formato TUT-Penn, un adattamento dello standard Penn Treebank per l'italiano. La struttura sintattica è la stessa del Penn ma, essendo la lingua italiana grammaticalmente più ricca di quella inglese, prevede un più vasto repertorio di tag (68 contro i 36 del Penn Treebank). Ogni tag è composto da una parte base, ad esempio *ADJ* per aggettivo, e un'addizionale informazione morfologica, ad esempio *DE* per dimostrativo. Ogni tipologia di informazione morfologica può essere associata soltanto a un numero limitato di tag di base; *DET* ad esempio, oltre che a *ADJ*, potrebbe legarsi soltanto a *PRO* (pronome)[18][13].

Il tagset adottato dal POS tagger di OpeNER prevede quindi i seguenti tag:

- *ADJ*: aggettivo . Può legarsi alle seguenti informazioni:
  - *QU*: qualificativo
  - *IN*: indefinito
  - *DI*: dimostrativo
  - *PO*: possessivo
  - *IR*: interrogativo
- *ADV*: avverbio; non ha ulteriori informazioni.

[15] http://www.di.unito.it/~tutreeb/

- *ART*: articolo; può legarsi alle seguenti informazioni:
    - *DE*: determinativo
    - *IN*: indeterminativo
- *CONJ*: congiunzione; non ha ulteriori informazioni.
- *NOUN*: sostantivo; può legarsi alle seguenti informazioni:
    - *PR*: proprio
    - *CS*: cosa singolare
    - *CP*: cosa plurale
- *NUMR*: numero; non ha ulteriori informazioni.
- *PREP*: preposizione; non ha ulteriori informazioni.
- *PRON*: pronome; può legarsi alle seguenti informazioni:
    - *ID*: indefinito
    - *PE*: personale
    - *DE*: dimostrativo
    - *RE*: relativo
    - *PO*: possessivo
    - *IN*: interrogativo
- *VMA*: forma base del verbo (verb *main*); VMO: verbo modale; VAU: verbo ausiliare; queste tre forme del verbo possono legarsi alle seguenti informazioni:
    - *IN*: infinito
    - *IM*: imperfetto
    - *IP*: indicativo presente
    - *RA*: passato remoto
    - *GE*: gerundio
    - *CO*: condizionale
    - *PA*: participio
    - *CG*: congiuntivo
    - *FU*: futuro

3.2.2.2 *Il web service*

Il servizio analisi computazionale offerto da OpeNER è stato progettato in modo da essere customizzabile e facilmente integrabile in un workflow. Ogni modulo è infatti integrato all'interno di un web service; ogni web service può essere invocato singolarmente oppure integrato in una pipeline di annotazione. In Figura 10 è schematizzato il processo di annotazione di un testo in lingua italiana che prevede l'utilizzo del tokenizer e del POS tagger. L'implementazione delle chiamate a questi web service verrà più approfonditamente affrontata nel capitolo relativo al motore TASEe (v. 7.1.3).

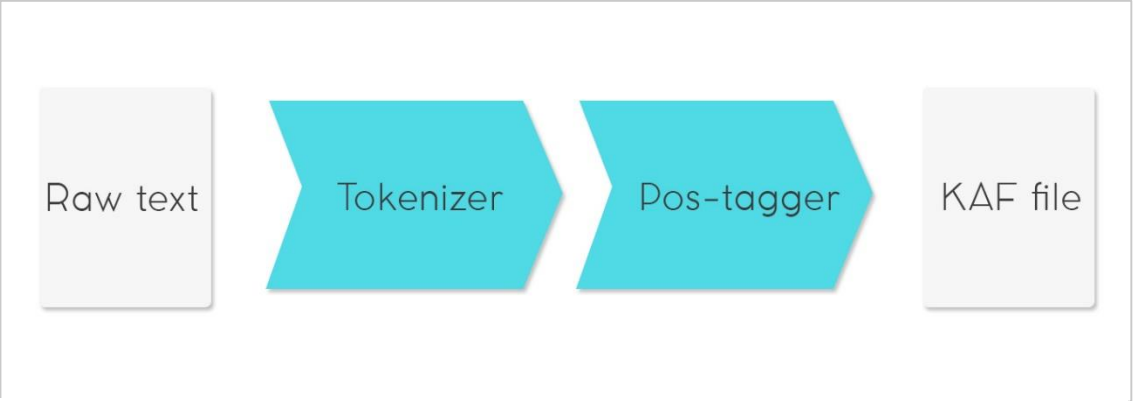

Figura 10 Componenti coinvolti nel processo di annotazione

3.3 Software *TASEe*

Il nome che ho deciso di dare al componente in questione è l'acronimo delle operazioni che tale software gestisce, dunque i processi di trascrizione, annotazione, sottotitolazione e creazione dei file *editable* (**T**ranscription, **A**nnotation, **S**ubtitling and **E**ditable engine). Tale componente è stato realizzato in Java e le tecnologie utilizzate per la sua implementazione sono dunque:

- la piattaforma JDK (Java Development Kit) v1.8.0_51;
- l'ambiente di esecuzione JRE (Java Runtime Environment) v1.8.0_51;
- l'ambiente di sviluppo Eclipse v4.4.2;

La struttura e il funzionamento di questo componente verranno dettagliatamente descritti nel Capitolo 7.

3.4 Software aggiuntivi

Ulteriori software che indirettamente sono stati utilizzati per la realizzazione dell'applicazione sono:

- Adobe Illustrator CS6 per la realizzazione del logo;

- Adobe Photoshop CS6 per la realizzazione delle immagini *tutorial*;
- Notepad++ v6.7.9 per la stesura del codice PHP;

4 L’applicazione web: front-end

In questo capitolo verrà descritto in modo dettagliato il funzionamento dell’applicazione web dal punto di vista dell’utente che ne fa uso (front-end).

4.1 La gestione degli utenti

Come accennato nel capitolo introduttivo, SubItalian prevede due tipologie di utenti: *learner* e *tutor*, che all’interno dell’applicazione hanno ruoli diversi.

4.1.1 Gli utenti *learner*

Gli utenti *learner* costituiscono il target cui SubItalian si rivolge, ovvero, gli apprendenti della lingua italiana. Un utente learner ha accesso alle seguenti funzionalità:

- possibilità di caricare un video di cui voglia generare trascrizione e sottotitolazione (v. 4.4);
- possibilità di scegliere uno dei video disponibili (tutti quelli da lui caricati e quelli caricati e resi pubblici da altri utenti) per vederne trascrizione e sottotitolazione (v. 4.5);
- possibilità di svolgere esercizi (v. 4.7.4) creati dai *tutor* (v. 4.7.3) a partire dalle trascrizioni dei video caricati e resi pubblici;
- possibilità di accedere al forum.

4.1.2 Gli utenti *tutor*

I *tutor* sono utenti di madrelingua italiana (o comunque con un’adeguata padronanza dell’italiano) che, oltre a poter usufruire di SubItalian per tutte le funzioni elencate per gli utenti *learner*, hanno il compito di aiutare gli utenti *learner* accedendo alle seguenti funzionalità:

- possibilità di revisionare e correggere la trascrizione e/o l’annotazione generate automaticamente dal sistema (v. 4.6)*;*
- possibilità di creare gli esercizi (v. 4.7.3) che verranno svolti dagli utenti *learner;*
- possibilità di analizzare statistiche riguardo alla composizione demografica degli utenti *learner* e le loro performance ottenute svolgendo le esercitazioni; tale funzionalità che può risultare utile per scopi di ricerca.

4.2 La registrazione

L'utente non loggato che accede alla pagina principale (http://servizi2.synthema.it/subitaliannew/index.php) di SubItalian, verrà reindirizzato in una pagina di benvenuto (Figura 11) in cui vengono mostrate due opzioni: *login*, da scegliere nel caso in cui si l'utente sia già registrato e dunque possieda username e password per accedere al portale, e *JoinUs*, da scegliere nel caso in cui l'utente voglia registrarsi al portale.

Figura 11 Pagina di benvenuto di SubItalian

Se sceglie *JoinUs*, prima di poter accedere al form di registrazione, l'utente dovrà selezionare la tipologia di utenza con cui si vuole registrare: *learner* o *tutor* (Figura 12). I campi del form da compilare variano in base alla tipologia di utenza selezionata; i campi che l'utente deve compilare in ogni caso sono:

- *username*: deve essere univoco e dunque l'utente verrà notificato nel caso in cui inserisca uno username già utilizzato;
- *email*: l'utente verrà notificato nel caso in cui la stringa inserita sia malformata;
- *password* (e ripetizione password): l'utente verrà notificato nel caso in cui le due password non corrispondano.

Al fine di poter fornire agli utenti *tutor* utili statistiche riguardo alle tendenze dell'apprendimento dell'italiano L2, gli utenti *learner* dovranno compilare quattro ulteriori campi:

- *data di nascita*

- *lingua madre*
- *sesso*
- *motivazioni sulla volontà di apprendimento dell'italiano* (opzionale)

Gli utenti *tutor* invece dovranno documentare la loro conoscenza della lingua italiana allegando un'autocertificazione che, in caso di dubbi, potrà essere controllata dall'amministratore dell'applicazione.

Una volta compilato correttamente il form di registrazione, l'utente riceverà una mail all'indirizzo di posta da lui inserito per confermare la propria registrazione; cliccando dunque sul link presente nel contenuto di tale mail, l'utente verrà reindirizzato nella pagina di *login* (Figura 12) di SubItalian, in cui potrà inserire username e password e accedere quindi al portale.

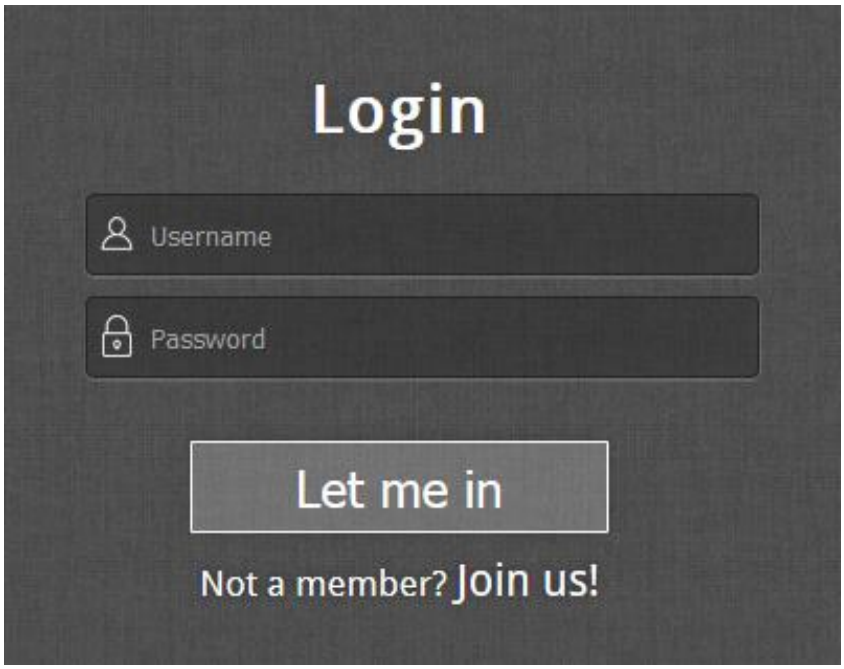

Figura 12 Form di Login

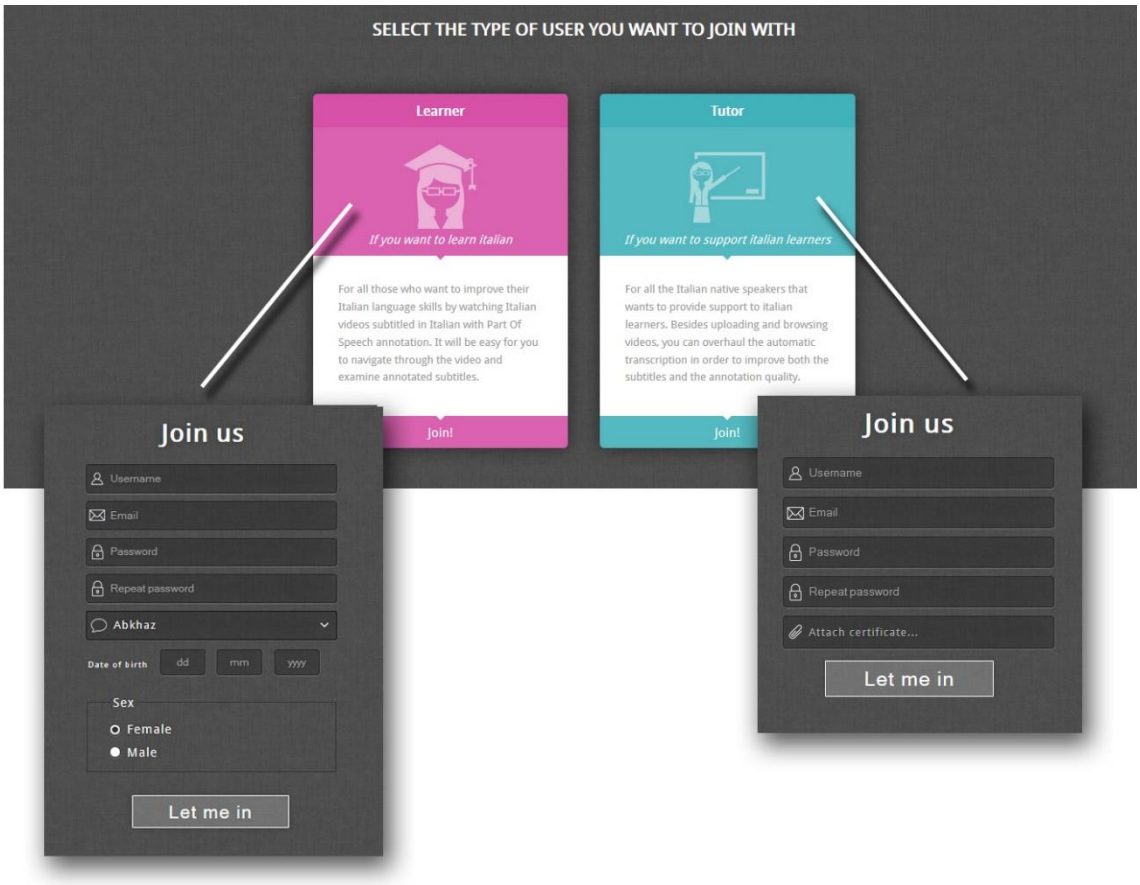

Figura 13 Pagina che permette di selezionare la tipologia di utente con cui registrarsi e relativi form di registrazione

4.3 La pagina *index*

Una volta effettuato il login, l'utente viene reindirizzato alla pagina principale. Nel caso di un utente *learner* tale pagina mostra soltanto tre pannelli (Figura 14.a): uno per l'upload (v.4.4), uno per la visualizzazione dei video (v.4.5) e uno per lo svolgimento delle esercitazioni (v. 4.7.4). La scelta di queste tre sezioni è stata effettuata per suggerire all'utente *learner* il percorso più logico da seguire:

1. fase di erogazione del materiale audiovisivo;
2. fase di *apprendimento* mediante la fruizione del materiale erogato e processato (trascritto e annotato) da SubItalian;
3. fase di *test* delle competenze acquisite in seguito alla fruizione del materiale.

La sequenzialità di questi tre passaggi non è comunque obbligatoria; l'utente potrà ad esempio fruire del materiale audiovisivo processato o svolgere delle esercitazioni senza prima aver caricato alcun video.

Nel caso di un utente *tutor* la pagina *index* mostra altri due pannelli (Figura 14.b): uno per la revisione (v.4.6) e uno per la visualizzazione delle statistiche (v. 4.8). Il pannello per le esercitazioni risulta essere uguale a quello degli utenti *learner* ma cliccandovi, l'utente *tutor* viene indirizzato alla pagina dedicata alla creazione degli esercizi (v. 4.7.3).

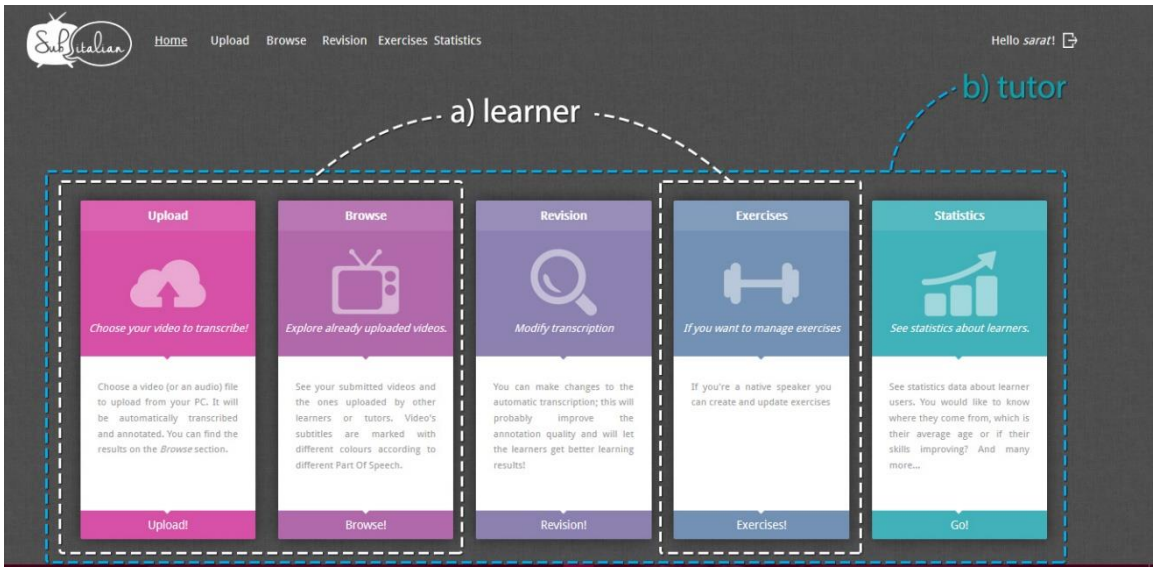

Figura 14 Pagina iniziale

Anche in questo caso la scelta di tali sezioni è stata effettuata per suggerire al *tutor* il percorso ottimale da seguire:

1. fase di erogazione del materiale audiovisivo;
2. fase di fruizione del materiale: in seguito all'erogazione del materiale audiovisivo caricato e processato da SubItalian, il *tutor* può accedere a questa sezione per rendersi conto dell'accuratezza dei risultati restituiti dal sistema e capire dove è necessario intervenire;
3. fase di revisione: in seguito alla visione del materiale il *tutor* potrà intervenire al fine di rendere attendibili i risultati del sistema;
4. fase di creazione delle esercitazioni: a partire dal materiale revisionato il *tutor* potrà generare le esercitazioni;

La sezione delle statistiche è da ritenersi estranea al percorso che il *tutor* dovrebbe seguire; si tratta infatti di una fase di monitoraggio a cui è logico accedere in qualsiasi momento.

Così come per le tre sezioni viste per gli utenti *learner*, anche in questo caso l'accesso sequenziale è comunque obbligatorio.

4.4 La sezione *upload*

La sezione *upload*, disponibile per entrambe le tipologie di utenza, permette di caricare un video di cui si vuole generare trascrizione e sottotitolazione annotate. Cliccando sul campo di input (Figura 15.a), l'utente potrà navigare tra le cartelle del proprio pc per selezionare il video desiderato. L'interfaccia mostra inoltre un checkbox con label *public* (Figura 15.b) che l'utente può spuntare nel caso in cui voglia rendere pubblico il suo video agli altri utenti; in caso contrario soltanto lui potrà accedere a tale video.

Una volta scelto il video, il pulsante di upload (Figura 15.c), dapprima inattivo, sarà reso attivo e potrà dunque essere cliccato per avviare il caricamento del video sul server. L'utente sarà notificato dello stato di avanzamento dell'upload mediante una *progress-bar* che specifica la percentuale del caricamento (Figura 15.d). Una volta terminato il trasferimento del file, partirà il processo di trascrizione (v. 7.1.2) e l'interfaccia, al posto della progress bar, mostrerà la label *Transcribing*. Terminato il processo di trascrizione, potrà partire il processo di annotazione (v. 7.1.3) e dunque la label di notifica mostrata dall'interfaccia diventerà *Annotating*. Terminato anche questo step partirà il processo di generazione dei sottotitoli annotati (v. 7.1.5) e quello per la generazione dei file EDT (v. 7.1.6) e l'utente ne verrà notificato mediante le label *Subtitling* e *Creating editable files* (in realtà, se non si verifica nessun errore, queste due fasi sono talmente rapide che l'utente non avrà modo di visualizzare il cambiamento di label). A questo punto, se l'intero processo è andato a buon fine, l'utente verrà reindirizzato nella pagina browse (v. 4.5) in cui potrà vedere il risultato del video appena caricato (e di altri, come spiegato nella sezione successiva).

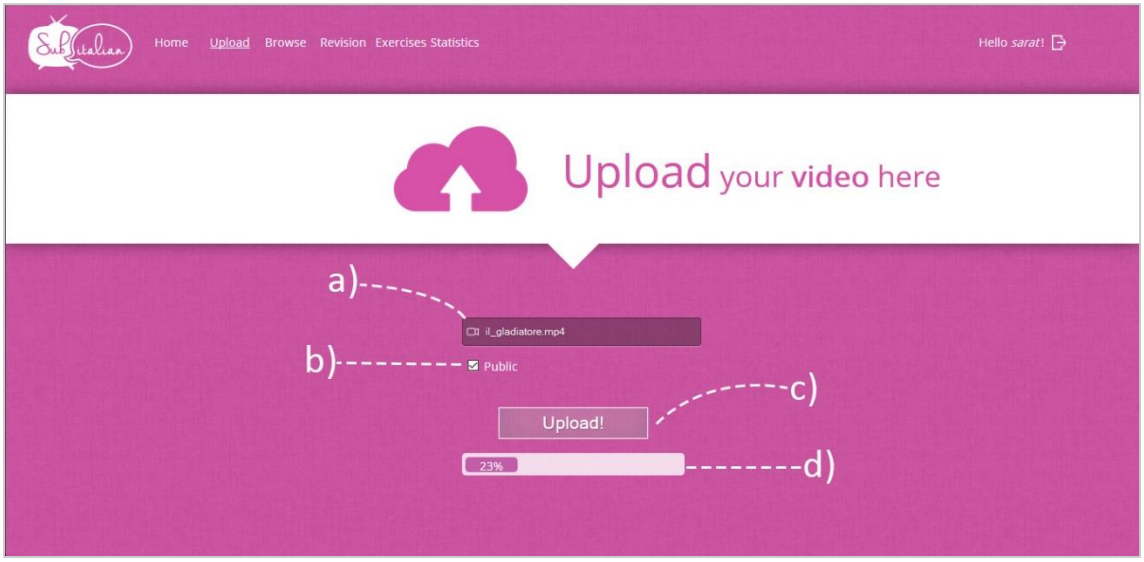

Figura 15 Sezione dedicata all'upload dei file

4.5 La sezione browse

In questa sezione, chiamata *browse*, l'utente potrà accedere alla visualizzazione dei video con la sottotitolazione annotata linguisticamente. Se l'utente giunge in questa pagina reindirizzato dalla pagina *upload* dopo il caricamento di un video, il video che di default viene visualizzato è quello appena caricato. Tramite il menu a tendina posto sulla sinistra (Figura 16.a) l'utente può comunque selezionare un video differente; la lista mostrerà tutti i video da lui caricati e i video caricati e resi pubblici dagli altri utenti. A fianco delle voci di video che sono stati completamente revisionati dagli utenti *tutor* (v. 4.6) saranno mostrate delle spunte verdi; in questo modo un utente *learner* potrà sapere di quali trascrizioni e annotazioni fidarsi maggiormente. Al centro è mostrato il player del video (Figura 16.b) e sulla destra del player è mostrata la legenda che descrive i colori assegnati a ogni parte del discorso (Figura 16.c).

Cliccando su *See full transcription* (Figura 16.d) l'utente potrà visualizzare la trascrizione integrale del video (con lo stesso markup utilizzato per i sottotitoli). Sopra il testo è mostrata una lista di checkbox (Figura 16.e) con cui è possibile selezionare le part-of-speech che si vogliono evidenziare nel testo. Cliccando con il tasto sinistro su una parola comparirà un *balloon* contenente ulteriori informazioni morfologiche riguardo a tale parola (Figura 16.f). Cliccando invece con il tasto destro appare un tooltip (Figura 16.g) che mostra la traduzione della parola cliccata nella lingua madre dell'utente (specificata al momento della registrazione).

Tenendo premuto il tasto *ctrl* e cliccando con il tasto sinistro su una parola, la riproduzione del video riprenderà dal momento in cui tale parola viene pronunciata.

Inoltre, per facilitare l'interazione con il player e rendere più confortevole l'analisi dei sottotitoli, sono stati implementati dei comandi da tastiera[16]:

- *ctrl* + *spazio*: permette di mettere in pausa o riprendere l'esecuzione del video;
- *ctrl* + *freccia destra*: permette di scorrere il video in avanti di tre secondi;
- *ctrl* + *freccia sinistra*: permette di scorrere il video indietro di tre secondi;

Nel caso in cui l'utente che sta visualizzando la pagina sia un *tutor*, in fondo alla trascrizione integrale verrà mostrato un pulsante *Revisiona* (Figura 16.h) che permette di passare alla sezione *revision* (v. 4.6) modificando trascrizione e/o annotazione del video corrente.

Nel caso in cui chi sta visualizzando la pagina sia un *learner* e il file che è stato caricato non sia stato revisionato, l'utente può sollecitare i *tutor* cliccando sul pulsante *Request Revision* (Figura 16.i). A tutti gli utenti *tutor* verrà inoltrata una mail automatica per notificarli del fatto che un utente *learner* ha richiesto la revisione del video indicato in tale mail (viene mostrato un link che li indirizzerà direttamente alla sezione di revisione di quel video).

[16] Tali comandi saranno utilizzati anche nella sezione per la revisione della trascrizione e/o dell'annotazione al fine di rendere più confortevole l'interazione con l'editor di modifica.

Figura 16 Sezione browse che permette di scegliere un video visualizzandone sottotitoli e trascrizione

4.6 La sezione *revision*

Tale sezione, disponibile soltanto per gli utenti *tutor*, permette di revisionare la trascrizione e/o l'annotazione generate automaticamente. Come per la sezione *browse*, sulla sinistra è presente un menu a tendina (Figura 17.a) tramite cui è possibile selezionare il video del quale si desidera modificare trascrizione/annotazione (nel caso in cui si sia giunti in questa sezione provenendo dalla pagina *browse*, il video selezionato di default sarà quello che si stava visualizzando). Sempre sulla sinistra è collocato il player (Figura 17.b) con cui è possibile interagire, oltre che utilizzando i classici pulsanti di controllo messi a disposizione dal player stesso, tramite comandi da tastiera, gli stessi spiegati per sezione *browse* (v.4.5). Tale interazione da tastiera si rivela ancora più importante in questa fase in quanto permette al *tutor* di utilizzare il mouse per interagire con l'editor di revisione (Figura 17.c) senza dover ogni volta ricollocare il cursore da sinistra a destra. L'editor di revisione presenta la trascrizione integrale annotata con lo stesso markup. Un menu a tendina posto sopra l'editor (Figura 17.d) permette di scegliere cosa si voglia revisionare: trascrizione o annotazione.

4.6.1 Revisione della trascrizione

Nel primo caso, per cambiare la parola trascritta, l'utente dovrà cliccare con il tasto sinistro del mouse sopra la parola che desidera editare. A fianco di questa, comparirà un campo di input testuale (Figura 17.e) in cui è possibile scrivere la nuova parola. Nel caso in cui nella trascrizione siano state omesse una o più parole, è possibile inserire nell'input testuale tutte le parole che stanno tra la parola cliccata e quella successiva; il motore calcolerà in automatico i timestamp di ogni parola inserita. Cliccando con il tasto destro sopra a una parola, compare un menu a tendina che permette di:

- rimuovere la parola cliccata;
- ripristinare il valore che la parola cliccata aveva prima dell'ultima modifica;
- rimuovere un eventuale elemento di punteggiatura agganciato alla parola cliccata;
- inserire un elemento di punteggiatura dopo la parola cliccata.

Per salvare le modifiche apportate è presente un pulsante *Salva modifiche* (Figura 16.f) che permette di salvare le modifiche apportate fino a quel momento. Questo non renderà il file *revisionato* (v. 4.5) ma permetterà di interrompere la revisione e di riprenderla in un secondo momento senza perdere le modifiche apportate. Il processo di trascrizione verrà nuovamente avviato a partire dal testo editato (v.7.1.7). L'utente sarà notificato dello stato di annotazione mediante una label *Annotating, please wait;* una volta terminato il processo, comparirà nuovamente il pulsante *Salva modifiche* per apportare ulteriori modiche.

4.6.2 Revisione dell'annotazione

Ovviamente la qualità della trascrizione incide notevolmente sulla qualità dell'annotazione linguistica. Ma è possibile comunque che, anche dopo aver revisionato la trascrizione e dunque avere un testo perlopiù corretto, persistano alcuni errori di annotazione. In questo caso è possibile selezionare dal menu a tendina (Figura 17.d) la voce *Annotazione* che permette di modificare la parte del discorso e le informazioni morfologiche di ogni singola parola. Cliccando con il tasto sinistro su una parola, compare un menu a tendina (Figura 17.g) che mostra la lista delle parti del discorso con il relativo colore di annotazione. A parte le categorie grammaticali *congiunzione*, *preposizione* e *avverbio*, le altre voci, al passaggio del mouse, mostrano un sotto-menu riportante le

informazioni morfologiche che è possibile assegnare a tale categoria. Selezionando una voce da questa lista, l'utente cambierà la PoS e l'informazione morfologica del token cliccato. Come per la modalità di revisione della trascrizione, anche qui è possibile ripristinare il valore della PoS precedente l'ultima modifica cliccando con il tasto destro del mouse sulla parola e selezionando la voce *Restore previous annotation*. Anche in questo caso per apportare le modifiche effettuate è necessario cliccare il pulsante *Salva modifiche* (Figura 17.f); questa volta il salvataggio impiegherà meno tempo in quanto non verrà fatto ripartire il processo di annotazione, ma verranno soltanto modificati i file SRT e EDT sostituendo le informazioni modificate (v. 7.1.8).

Soltanto quando il *tutor* riterrà che trascrizione e annotazione siano completamente corrette potrà cliccare il pulsante *Rilascia revisione* (Figura 17.h); a questo punto il file sarà visibile nella sezione *browse* come *revisionato* (v. 4.5).

Al di sotto di tale pulsante è presente inoltre un pulsante *Crea esercitazioni* (Figura 17.i) con cui l'utente *tutor* viene reindirizzato nella sezione per la creazione delle esercitazioni (v. 4.7.3), avendo come testo di default quello appena revisionato.

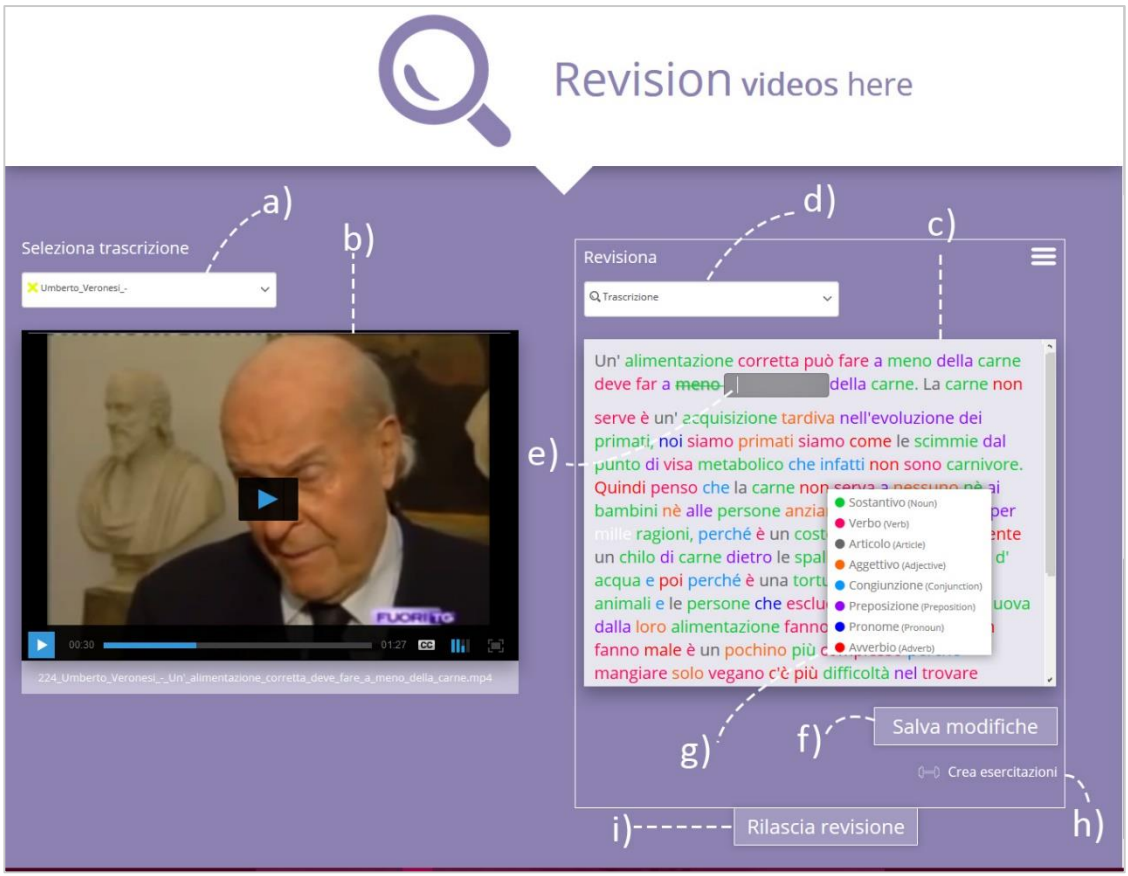

Figura 17 Sezione revision dedicata alla revisione di trascrizione e annotazione

4.7 Le esercitazioni

Questa sezione è disponibile in due diverse modalità a seconda della tipologia di utenza: *creazione* delle esercitazioni per quanto riguarda gli utenti *tutor* e *svolgimento* di tali esercitazioni per quanto riguarda gli utenti *learner*. Le tipologie di esercizi disponibili sono quattro: cloze facilitato, cloze orale facilitato, part-of-speech tagging (analisi grammaticale) e trascrizione.

4.7.1 *Cloze*

Il termine *cloze* deriva dal principio che sta alla base della legge di chiusura (closure law) della teoria della scuola di pensiero Gestalt[57]. Secondo tale legge la mente umana è portata a vedere *chiuse* figure che in realtà non lo sono, riconducendole a figure note. Se ad esempio a una linea che disegna un cerchio manca una porzione di arco, l'occhio umano interpreterà comunque il disegno come un cerchio immaginando il pezzo di arco mancante. Trasportando questo principio sul testo scritto diremmo che se a un testo mancano alcune parole se ne dovrebbe comunque comprendere il contenuto *immaginando* le parole mancanti grazie al contesto entro cui sono collocate. Il cloze è

quindi una tecnica di esercitazione che consiste nell'inserire in una porzione di testo parole precedentemente cancellate. In realtà il cloze nasce con una finalità differente. Tale procedura, messa a punto da Wilson Taylor nel 1953, era stata elaborata per valutare la leggibilità di un testo. In un secondo momento diventa uno standard come esercizio didattico, comunemente utilizzato durante la fase si apprendimento della seconda lingua al fine di testare le capacità di comprensione del testo dello studente. In un cloze le parole possono essere cancellate in modo perlopiù casuale oppure seguendo una certa regolarità. Un esercizio cloze può infatti avere diverse caratteristiche:

- cloze a crescere: all'interno del testo si inizia a cancellare ogni settima parola e con l'aumentare della definizione del contesto, si diminuisce l'intervallo tra una cancellazione all'altra (aumentando la difficoltà dell'esercizio), cancellando prima ogni sei e successivamente anche ogni cinque parole (al disotto di tale intervallo il testo diventa non ricostruibile);
- cloze facilitato: le parole mancanti vengono mostrate in calce al testo dell'esercizio, talvolta con qualche parola intrusa. Mentre in un cloze non facilitato al risolutore è solitamente consentito inserire una parola che non sia esattamente quella cancellata, bensì anche un sinonimo che comunque ricostruisca l'esatto concetto, in questa tipologia di cloze si è costretti ad inserire una delle parole proposte in calce;
- cloze orali: si fa uso di un supporto audio/video e si stoppa l'esecuzione di una registrazione audio lasciando che lo studente immagini la parola o il concetto che segue[54][55].

In SubItalian vengono proposte due tipologie di cloze:

- cloze facilitato (standard): come prevede tale tipologia di cloze, in fase di risoluzione dell'esercizio all'utente *learner* verranno mostrate in calce alla pagina le parole cancellate in ordine sparso con una parola intrusa;
- cloze *orale* facilitato: tale tipologia non è catalogabile tra nessuna di quelle elencate sopra, ma è stata ideata in fase di realizzazione del progetto, dato il contesto *audiovisivo* che caratterizza SubItalian. In fase di risoluzione dell'esercizio, il *learner* avrà a disposizione un player audio che riproduce la parte di audio corrispondente alla porzione di testo che compone l'esercizio. L'utente potrà quindi interagire con il player (anche in questo caso ha la possibilità di

interagire tramite gli stessi comandi da tastiera visti per lezioni *browse* e *revision*) per ascoltare le parole mancanti e cercare di trascriverle correttamente.

Ovviamente con questi due tipi di cloze si valutano due tipi di competenza differenti: nel primo caso, come in un cloze standard, si misurano le capacità di comprensione del testo mentre nel secondo caso vengono messe alla prova le capacità di trascrizione.

La scelta di un cloze facilitato è dovuta al fatto che sarebbe stato ben più complesso eseguire una correzione automatica che prevedesse di valutare l'adeguatezza di un sinonimo della parola mancante; costringendo l'utente *learner* a scegliere tra una delle parole presentategli, la correzione sarà senz'altro attendibile.

4.7.2 *PoST* (part-of-speech tagging)

L'etichetta part-of-speech tagging può essere interpretata da due punti di vista differenti:

- dal punto di vista dell'NLP si intende la procedura automatica mediante la quale vengono annotati i testi, come si è visto per l'annotatore di OpeNER (v. 3.2.2);
- dal punto di vista didattico si intende una tipologia di esercitazione, in italiano chiamata analisi grammaticale, mediante la quale un umano deve assegnare ad ogni parola che compone una frase la corretta parte del discorso. Tale tipologia di esercitazione è inclusa in SubItalian.

4.7.3 Trascrizione dell'audio

Questa tipologia di esercitazione, particolarmente apprezzata dagli apprendenti alle prime armi[17], consiste semplicemente nel trascrivere un video. Questo esercizio può essere considerato una versione più complessa del *cloze orale facilitato* e permette di valutare le capacità di associazione tra forma orale e forma scritta dell'italiano. In SubItalian il *learner* ha a disposizione il player video e un editor di testo dove poter editare la trascrizione. La porzione di audio da trascrivere viene selezionata dal *tutor* al momento della creazione dell'esercizio.

4.7.4 Creazione delle esercitazioni

Gli utenti *tutor* che giungono alla pagina relativa alla creazione delle esercitazioni potranno scegliere la tipologia di esercizio da creare mediante l'apposito menu a tendina (Figura 18.a). Le scelte possibili sono dunque: cloze facilitato; cloze orale facilitato; part-

[17] Secondo quanto testimoniato dai collaboratori del consorzio ICoN.

of-speech tagging; trascrizione. Sulla sinistra è presente un altro menu a tendina (Figura 18.b) per mezzo del quale l'utente può scegliere il video dalla cui trascrizione vuole prelevare la porzione di testo. Sotto il menu a tendina è riportata la trascrizione integrale (Figura 18.c) del video selezionato. Per scegliere la porzione di testo che costituirà l'esercizio, il *tutor* deve semplicemente selezionarlo dalla trascrizione cliccando con il tasto sinistro del mouse sulla prima parola che desidera e, tenendo premuto il tasto, trascinare la selezione fino all'ultima parola che vuole includere nell'esercizio (Figura 18.d). Nel momento in cui il mouse viene rilasciato, il testo compare sulla destra dell'interfaccia (Figura 18.e). L'utente (fintanto che non premerà il pulsante *Salva*) può cambiare idea in qualsiasi momento e dunque eseguire una nuova selezione; il testo mostrato sulla destra verrà sostituito con quello della nuova selezione. È anche possibile selezionare e modificare uno degli esercizi precedentemente creati mediante il terzo menu a tendina (Figura 18.f).

4.7.4.1 Creazione cloze

La creazione di un esercizio cloze è identica sia che l'utente abbia deciso di creare un cloze facilitato (standard) sia che abbia selezionato un cloze orale facilitato. Una volta quindi scelta la porzione di testo desiderata, il *tutor* può procedere alla cancellazione delle parole e ha due possibilità:

- cancellare manualmente le parole: in questo caso sarà sufficiente che clicchi sulla parola che desidera cancellare (passando il mouse sopra ogni parola non ancora cancellata il cursore assume le sembianze di una *x* rossa). Al posto della parola cancellata appariranno tanti *underscore* quante sono le lettere che componevano la parola cancellata (Figura 18.g). In caso di ripensamenti, è possibile ripristinare il valore della parola cliccando sullo spazio vuoto creato (il cursore, passando il mouse sopra le parole cancellate assume le sembianze di una freccetta verde di ripristino).
- lasciare che il sistema crei automaticamente un *cloze a crescere*: in questo caso l'utente deve spuntare il checkbox *Crea automaticamente cloze a crescere*, scegliere il massimo intervallo di cancellazione (tra 7, 6 e 5) e ogni quanto decrementare l'intervallo nel caso in cui si scelga di non limitarsi alla cancellazione ogni sette parole. Se ad esempio il *tutor* desidera decrementare l'intervallo fino a cinque, dopo aver effettuato due cancellazioni ogni sette parole e due cancellazioni ogni sei parole, dovrà selezionare, tra i tre radio button

mostrati nella sezione *Massimo intervallo di cancellazione* (Figura 18.h), quello con valore 5 e inserire 2 nel campo input sotto la label *Intervallo di incremento cancellazione* (Figura 18.i). Se le impostazioni settate dal *tutor* prevedono un testo più lungo della porzione che è stata selezionata, all'utente verrà chiesto se desidera proseguire (interrompendo prematuramente il decremento) o se riformulare l'esercizio (aumentando la porzione di testo o cambiando le impostazione per la creazione automatica). Una volta creato l'esercizio, il *tutor* può comunque modificarlo con la stessa modalità con cui può creare l'esercizio manualmente.

Una volta cancellate le parole dal testo, prima di poter salvare l'esercizio, il *tutor* deve valutarne la difficoltà. L'interfaccia mostra tre radio button (Figura 18.l) tramite cui il *tutor* decide se l'esercitazione appena creata è da ritenersi facile, mediamente difficile o difficile; in questo modo gli utenti *learner* potranno decidere di iniziare a svolgere esercitazioni più facili per poi passare ad esercitazioni più complesse. A questo punto, per creare definitivamente l'esercizio, il *tutor* deve cliccare sul pulsante *Salva* (Figura 18.m).

Figura 18 Sezione exercise, dedicata alla creazione di esercizi (utenti tutor)

4.7.4.2 *Creazione post*

La creazione di un'esercitazione di tipo part-of-speech tagging è decisamente immediata. La soluzione dell'esercizio sta proprio nell'annotazione delle parole e il testo mostrato, essendo un EDT (v. 7.1.6), contiene già tutti i metadati necessari alla definizione della part-of-speech di ogni parola (Figura 19.a). Il *tutor* dovrà semplicemente spuntare uno o più checkbox relativi alle part-of-speech che desidera che vengano annotate in quell'esercizio (Figura 18.b) e valutare il livello di difficoltà (Figura 18.c). Fatto questo, come per le esercitazioni cloze, deve confermare la creazione cliccando sul pulsante *Salva*. (Figura 19.d).

Figura 19 Creazione di un esercizio PoST (utenti tutor)

4.7.4.3 *Creazione esercizio di trascrizione*

Per creare un esercizio di trascrizione, il *tutor* deve semplicemente selezionare la porzione di testo che desidera far trascrivere al risolutore (Figura 20). Come per le altre tipologie

Figura 20 Creazione esercizio di trascrizione

di esercizio deve selezionare il livello di difficoltà e premere il pulsante *Salva* per creare definitivamente l'esercitazione. In questo esercizio, ancor più che negli altri, al fine di garantire una correzione attendibile è fondamentale che la trascrizione automatica sia stata revisionata.

4.7.5 Svolgimento delle esercitazioni

Accedendo alla sezione esercitazioni, gli utenti *learner* avranno la possibilità di svolgere gli esercizi creati dagli utenti *tutor*. L'interfaccia mostra tre pannelli di cui, inizialmente, soltanto il primo è attivo (Figura 21.a); questo permette di selezionare la tipologia di esercitazione che si intende svolgere. Scelta la tipologia, viene attivato il secondo pannello (Figura 21.b) che permette invece di scegliere il livello di difficoltà dell'esercizio. A questo punto si attiva il terzo pannello che serve soltanto per confermare le due scelte effettuate (prima della conferma è ancora possibile modificarle). Cliccando sul pulsante *Done* (Figura 21.c), il sistema recupera in modo casuale uno degli esercizi presenti nel database che rispecchino le caratteristiche specificate dall'utente (tipologia e difficoltà). Nel caso in cui l'utente abbia già svolto tutte le esercitazioni presenti con le caratteristiche specificate, verrà notificato dal sistema che gli permetterà di scegliere se aspettare che gli utenti *tutor* creino nuovi esercizi o se riprendere a svolgere gli esercizi già effettuati (magari per valutare un eventuale miglioramento).

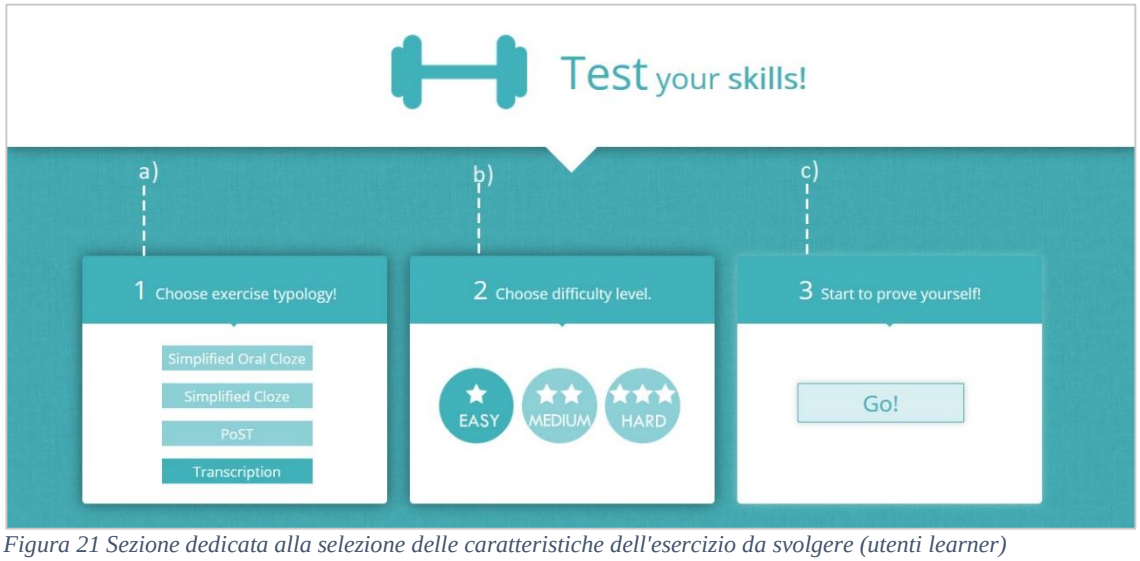

Figura 21 Sezione dedicata alla selezione delle caratteristiche dell'esercizio da svolgere (utenti learner)

4.7.5.1 Svolgimento esercizi cloze

Nel caso di un esercitazione di tipo cloze orale facilitato, l'utente potrà ascoltare la porzione di audio relativa al testo dell'esercizio mediante il player posto sotto il testo

dell'esercizio (Figura 22.a). L'utente dovrà cliccare sopra ogni spazio vuoto (Figura 22.b) ed inserire la parola mancante. Dopo aver inserito tutte le parole mancanti, dovrà premere il pulsante *Done* (Figura 22.c) per procedere alla correzione (il sistema notificherà l'utente nel caso in cui non abbia inserito tutte le parole mancanti). Sulla sinistra della pagina vi sono due ulteriori pulsanti, comuni a tutte le tipologie di esercitazione: il primo (Figura 22.d) porta alla sezione precedente e permette quindi di creare un nuovo esercizio scegliendone le caratteristiche e il secondo (Figura 21.e) carica un nuovo esercizio con le stesse caratteristiche dell'esercizio che si stava svolgendo.

Figura 22 Svolgimento di un esercizio di tipo cloze orale facilitato

Nel caso di un'esercitazione di tipo cloze facilitato standard, la modalità di esecuzione è simile a quella vista per il cloze facilitato orale. La differenza sta nel fatto che le parole da inserire sono mostrate in calce al testo dell'esercizio (Figura 22.a) e l'utente deve trascinarle (Figura 23.b) una per una all'interno del testo, collocandole nello spazio vuoto

che ritiene opportuno. Sia il pulsante *Done* che i due pulsanti ai lati dell'esercizio hanno le stesse funzionalità descritte per il cloze orale facilitato.

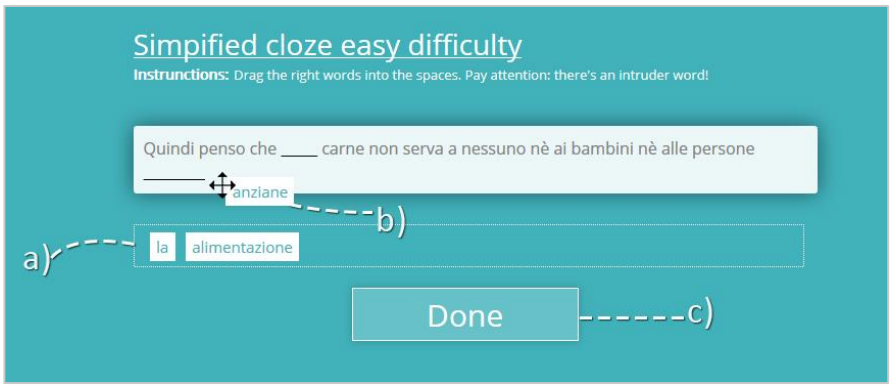

Figura 23 Svolgimento di un esercizio di tipo cloze facilitato standard

La correzione dei due tipi di esercizi è identica e mostra al *learner* il punteggio raggiunto (Figura 24.a) calcolato come percentuale di risposte esatte sul totale delle risposte date, mostra in verde le parole inserite correttamente (Figura 24.b) e in rosso quelle errate (Figura 24.c) che presentano a fianco la correzione.

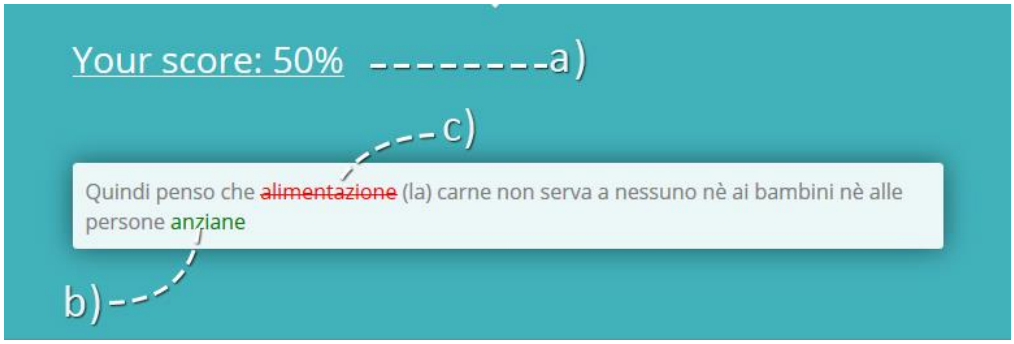

Figura 24 Risultato di esercizi cloze

4.7.5.2 Svolgimento PosT

Nel caso di un'esercitazione di tipo PoST, l'interfaccia mostra all'utente il testo dell'esercizio colorato in grigio (Figura 25.a). Le istruzioni per lo svolgimento dell'esercizio specificano quali part-of-speech annotare (Figura 25.b). Cliccando con il tasto sinistro del mouse sopra una parola, comparirà un menu a tendina (Figura 25.c) contenente una o più voci che rappresentano le parti del discorso previste (con il relativo colore di annotazione). Tale menu contiene inoltre una voce *Remove annotation* che permette di rimuovere l'annotazione precedentemente scelta. L'utente dovrà quindi selezionare la parte del discorso con cui ritiene di annotare la parola cliccata; tale parola assumerà dunque il colore relativo alla parte del discorso selezionata. Prima di aver cliccato il pulsante *Done*, sarà sempre possibile modificare o rimuovere l'annotazione di ogni parola.

Figura 25 Svolgimento di un esercizio di tipo PoST

La correzione di un esercizio PoST mostra all'utente il punteggio ottenuto (Figura 26.a e 26.b) (il calcolo del punteggio è descritto nella Sezione 5.8.2.c) per ogni part-of-speech che si richiedeva di annotare. Viene inoltre mostrata la correzione dell'esercizio: le parole annotate correttamente vengono mostrate in verde (Figura 26.c), le parole non annotate vengono mostrate in arancione (Figura 26.d) e le parole annotate erroneamente vengono mostrate in rosso (Figura 26.e).

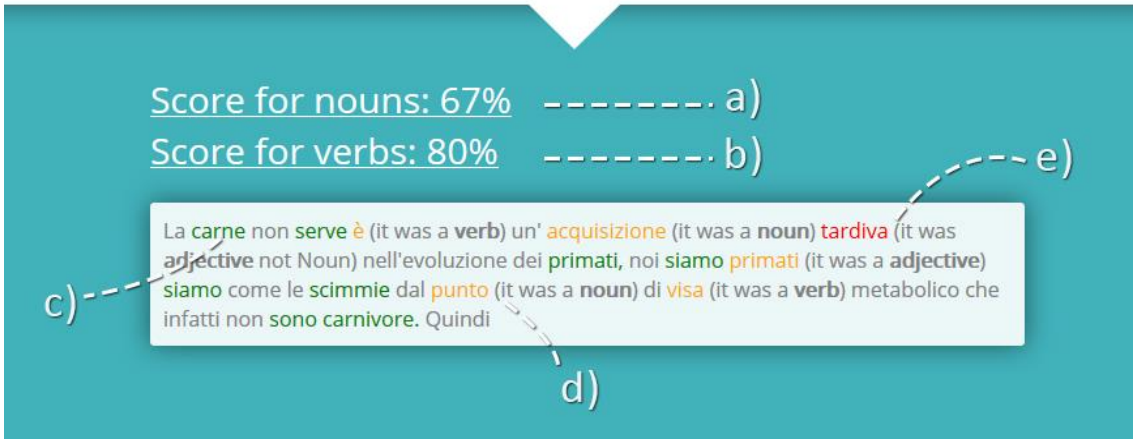

Figura 26 Risultato di un esercizio PoST

4.7.5.3 Svolgimento esercizio di trascrizione

Nella sezione dedicata allo svolgimento dell'esercizio di trascrizione, viene riprodotta la porzione di video (Figura 27.a) corrispondente alla trascrizione selezionata dal *tutor* (v. 4.7.4.3). A fianco del player si trova l'editor testuale (Figura 27.b) all'intero del quale il *learner* deve editare la trascrizione. Anche in questo caso è possibile interagire con il player tramite i comandi da tastiera rammentati nelle istruzioni dell'esercizio (Figura 27.c); in questo modo il risolutore potrà scrivere agevolmente senza dover staccare le mani dalla tastiera per sospendere e riprendere l'esecuzione del video tramite il mouse. Il learner non è tenuto a trascrivere anche la punteggiatura in quanto non verrà tenuta in considerazione in fase di correzione.

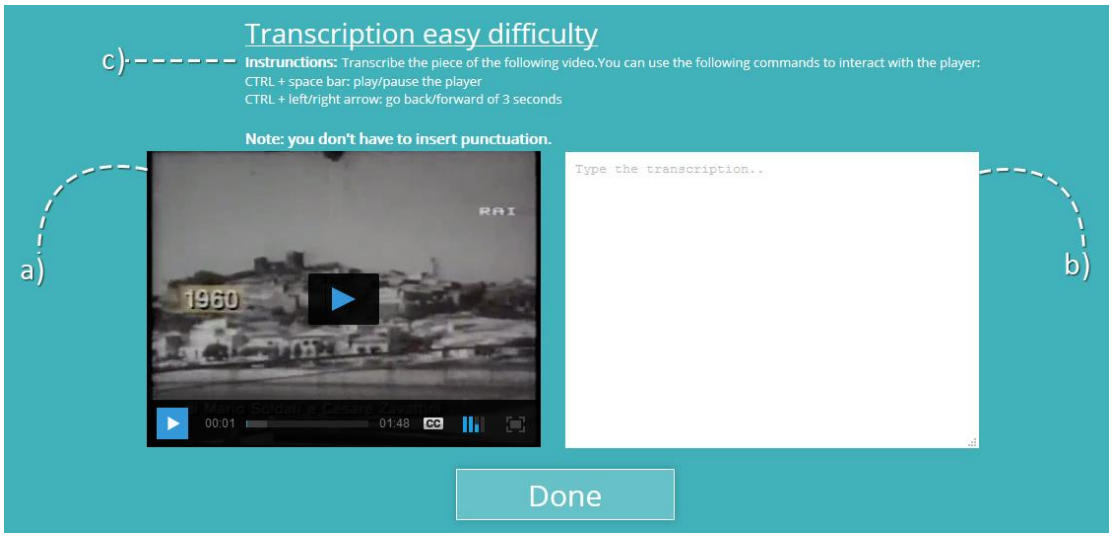

Figura 27 Svolgimento di un esercizio di trascrizione

La correzione mostra il punteggio percentuale ottenuto (Figura 28.a) (il calcolo del punteggio è spiegato nella Sezione 5.8.2.d)), la trascrizione originale (Figura 28.b) e quella inserita dal *learner* (Figura 28.c).

Figura 28 Risultato di un esercizio di trascrizione

4.8 Le statistiche

Nella sezione statistiche, l'utente *tutor* può esaminare alcune caratteristiche riguardo agli utenti *learner*. Accedendo alla pagina, l'utente si trova davanti due pannelli attraverso cui può decidere quale tipologia di informazioni analizzare. Cliccando sul primo (Figura 29.a), caratteristiche demografiche, vengono mostrati due grafici:

- il primo (Figura 30.a), è un diagramma a colonne in pila che descrive la distribuzione degli utenti per fasce d'età e per sesso. Sull'asse delle ascisse sono riportate le fasce di età e sull'asse delle ordinate è indicato il numero di utenti *learner* iscritti al portale. Le colonne, in pila, mostrano il numero di utenti di sesso maschile (in azzurro) e il numero di utenti di sesso femminile (in rosa). Tale tipologia di grafico è particolarmente idonea a rappresentare questo tipo di informazione in quanto è pensata proprio per accorpare i dati dell'asse delle ascisse[18] suddividendoli in intervalli regolari; sarebbe inopportuno visualizzare il numero di utenti per ogni età possibile. Inoltre, la suddivisione delle colonne fornisce una chiara visione della distribuzione di maschi e femmine per ogni classe di età.
- Il secondo (Figura 30.b) è un grafico a torta che mostra la ripartizione degli apprendenti per lingua madre. Cliccando su una fetta viene mostrato un *tooltip* (Figura 30.c) che riporta il numero di maschi e il numero di femmine iscritti a SubItalian con tale lingua. Si è scelto di utilizzare il grafico a torta poiché è funzionale a rappresentare percentuali la cui somma costituisce l'intero insieme di dati; un diagramma a colonne in questo caso avrebbe fornito una lettura della distribuzione molto meno immediata. Inoltre, a differenza della distribuzione delle fasce d'età che per mantenere una certa chiarezza richiede un ordine di presentazione cronologico, per rappresentare la distribuzione della lingua madre degli utenti non è necessario alcun ordine di presentazione e nel grafico a torta non esiste tale vincolo.

[18] Ci si riferisce all'asse dell'ascisse poiché il grafico è a colonne ma ovviamente il discorso è equivalente per l'asse delle ordinate in un grafico a barre.

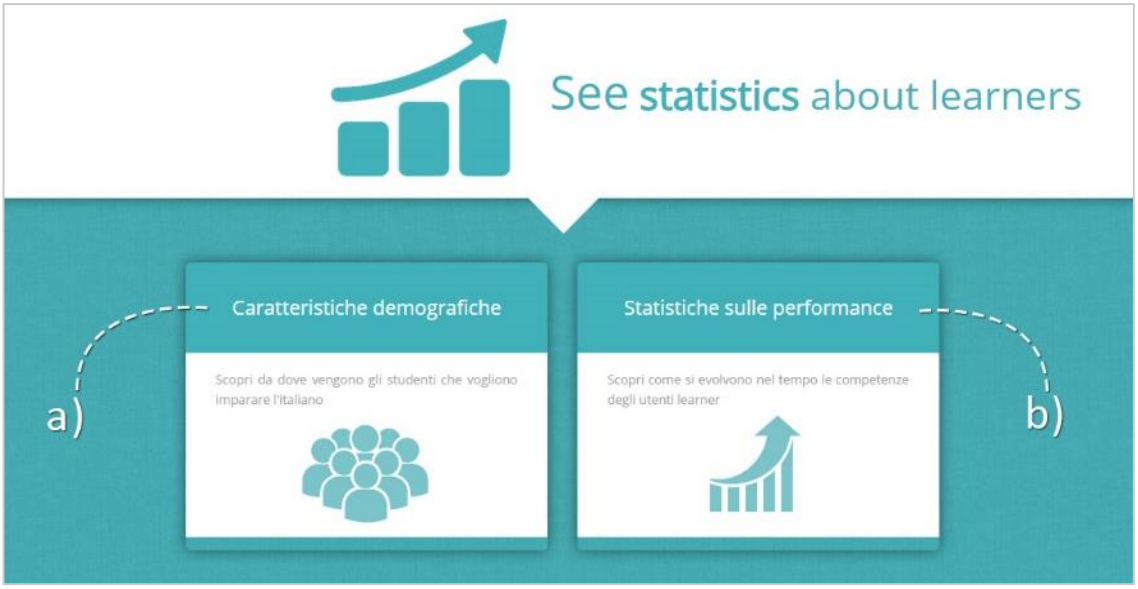

Figura 29 Sezione statistiche

Figura 30 Statistiche demografiche

Scegliendo invece il secondo box (Figura 29.b) viene mostrato un grafico lineare (*line*)(Figura 31) che riporta l'andamento di ciascun utente *learner* in termini di punteggio ottenuto dalle esercitazioni svolte nel tempo. L'asse delle ascisse riporta il range temporale, mentre l'asse delle ordinate riporta il punteggio percentuale ottenuto per ogni

esercizio svolto. Le linee uniscono dunque i punti che rappresentano lo svolgimento di un'esercitazione; passandovi sopra con il mouse vengono mostrati dei *tooltip* (Figura 31.a) che riportano i dati di tale esercitazione: data e ora di svolgimento dell'esercizio, punteggio ottenuto, lingua madre, nome e sesso dell'utente che lo ha svolto. Il sesso degli utenti è comunque visibile anche dal tipo di linea: quelle tratteggiate indicano utenti di sesso femminile, quelle punteggiate utenti di sesso maschile. Sopra al grafico sono riportati quattro *checkbox* (Figura 31.b) che permettono di selezionare le tipologie di esercizio di cui si vogliono vedere le performance. Selezionando ad esempio cloze facilitato orale e cloze facilitato non verranno mostrati dati relativi alle esercitazioni di tipo part-of-speech tagging. Sotto al grafico è invece riportato un selettore temporale (Figura 31.c) che permette di zoomare avanti o indietro nel range temporale, nonché di *shiftare* la selezione avanti e indietro nel tempo.

Si è scelto di utilizzare questo tipo di grafico poiché idoneo a rappresentare l'andamento di una certa variabile nel tempo. Tramite la rappresentazione lineare è possibile sia avere una visione generale dell'evoluzione delle performance di ogni utente, sia mettere in risalto ogni occorrenza di svolgimento di un esercizio. Il selettore temporale è utile in quanto il range temporale entro cui i *learner* possono svolgere le esercitazioni è molto ampio e certamente non è per tutti lo stesso. Si supponga di aver optato per un asse temporale *fisso* e che tra il primo e l'ultimo esercizio svolto siano intercorsi due anni. Le performance di un *learner* che abbia svolto esercitazioni (con una qualsiasi frequenza) durante tutto l'arco temporale, saranno perfettamente esaminabili; ma le performance di un *learner* che abbia svolto una serie di esercizi soltanto nei primi due mesi saranno senz'altro illeggibili. Tramite il selettore temporale invece sarà possibile *zoomare* tale periodo e analizzare chiaramente anche le performance di questo utente.

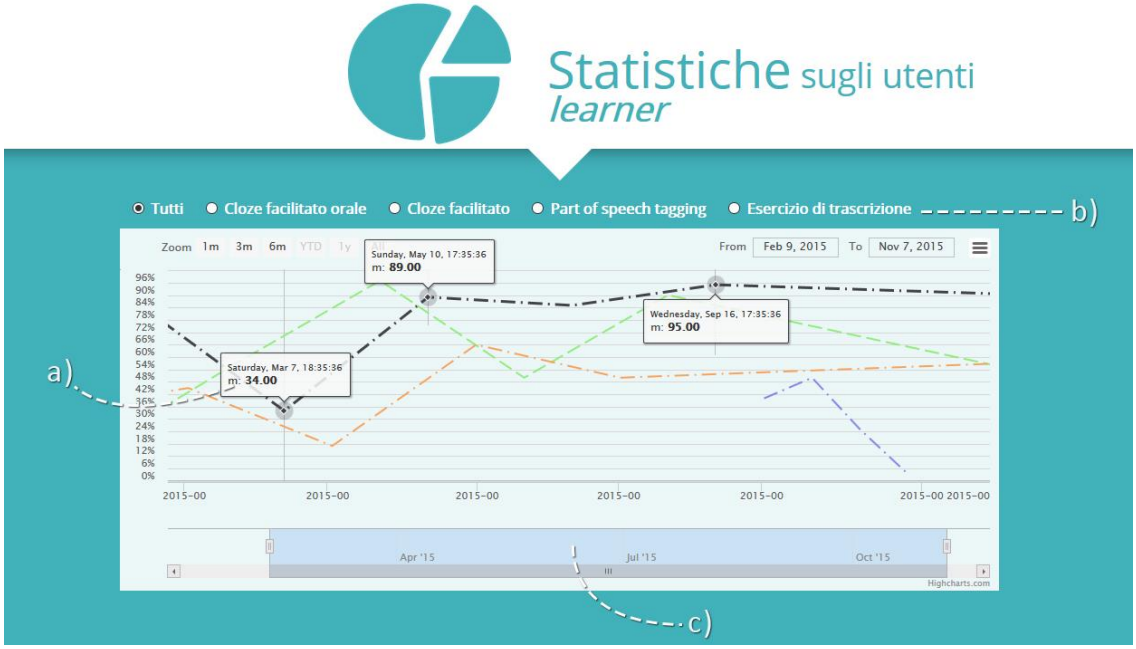

Figura 31 Statistiche sulle performance

4.9 Il forum

La sezione del forum è suddivisa in tre sezioni: quella che mostra le categorie di discussione, quella che riporta i thread aperti per una certa categoria e quella che mostra i post lasciati per un certo thread.

4.9.1 Le categorie di discussione

Accedendo alla sezione forum, la prima sezione che viene mostrata è quella riportante la lista delle categorie di discussione attualmente presenti. Ogni voce della lista riporta:

- titolo della categoria (Figura 32.a);
- numero di thread presenti per quella categoria (Figura 32.b);
- titolo, autore e data di creazione dell'ultimo thread aperto (Figura 32.c).

Cliccando sopra una delle categorie di discussione si passa alla seconda sezione.

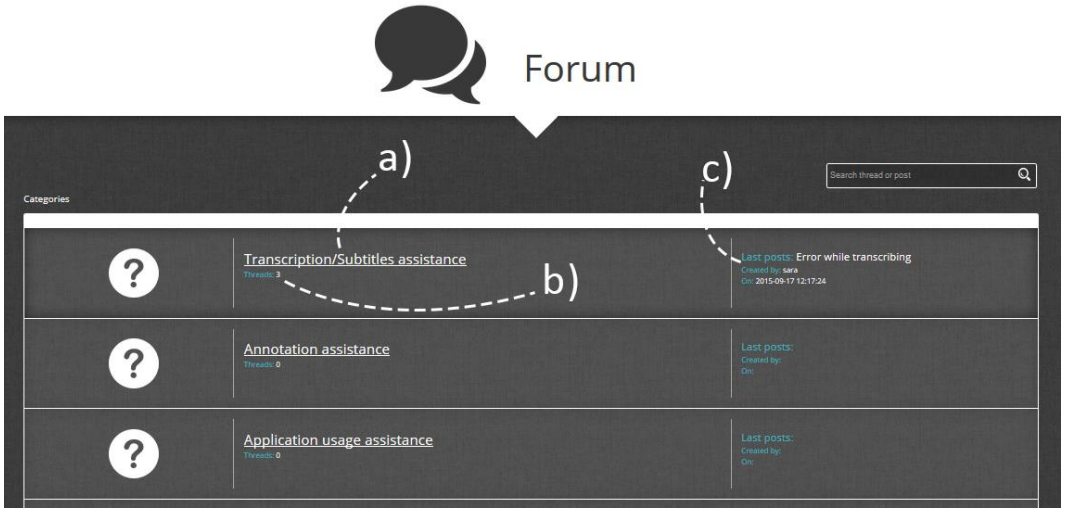

Figura 32 Categorie di discussione del forum

4.9.2 *I thread*

Nella seconda sezione è riportata la lista dei thread aperti all'interno della categoria selezionata nella sezione precedente. Questa lista è presentata in ordine anticronologico rispetto alla data di creazione e ogni voce riporta:

- titolo, autore e data di creazione del thread (Figura 33.a);
- numero di messaggi in risposta (post) al messaggio iniziale (topic) (Figura 33.b);
- autore e data di creazione dell'ultimo post scritto (Figura 33.c).

Cliccando sul pulsante *Open thread* (Figura 33.d) viene mostrato un form che permette di creare una nuova discussione inserendo titolo del thread e corpo del messaggio che

costituirà il primo post del thread; il topic. Non appena creato, il thread verrà mostrato in cima alla lista.

Cliccando sopra uno dei thread si passa alla terza sezione.

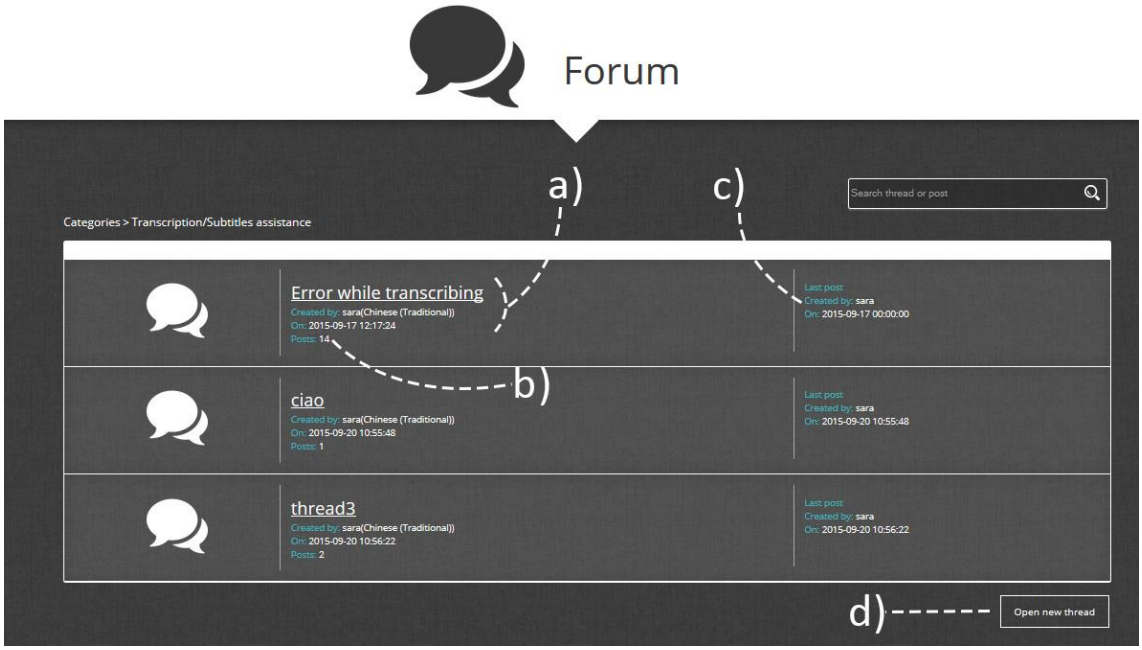

Figura 33 Sezione dei thread per la categoria di discussione "Transcription/Subtitles assistance"

4.9.3 *I post*

La terza sezione contenente infine la lista dei messaggi (compreso il topic) appartenenti al thread selezionato. La lista è ordinata cronologicamente in base alla data di creazione del messaggio stesso e ogni voce riporta:

- nome e lingua madre dell'autore e data di creazione del post (Figura 34.a);
- titolo del thread (Figura 34.b) scelto dall'autore al momento della creazione e preceduto da *Re:* in tutti i messaggi di risposta al topic (Figura 34.c);
- corpo del post (Figura 34.d);

Per partecipare alla discussione e lasciare un messaggio, l'utente deve cliccare sul pulsante *Reply* (Figura 34.e). Viene quindi mostrato un form attraverso cui inserire il messaggio di risposta che verrà mostrato in fondo alla lista degli attuali messaggi.

L'utente può navigare tra queste tre sezioni tramite il menu di navigazione (Figura 34.f) posto in cima a ciascuna delle tre liste.

Sia i thread che i post creati da utenti tutor vengono evidenziati in azzurro. È inoltre disponibile in tutte e tre le sezioni una barra di ricerca (Figura 34.h) tramite cui l'utente

può cercare post di suo interesse digitando una o più parole chiave. Il sistema esegue una ricerca di tali parole all'interno dei titoli dei thread e dei massaggi. Al termine della ricerca vengono mostrati, anch'essi sotto forma di lista, i post recuperati.

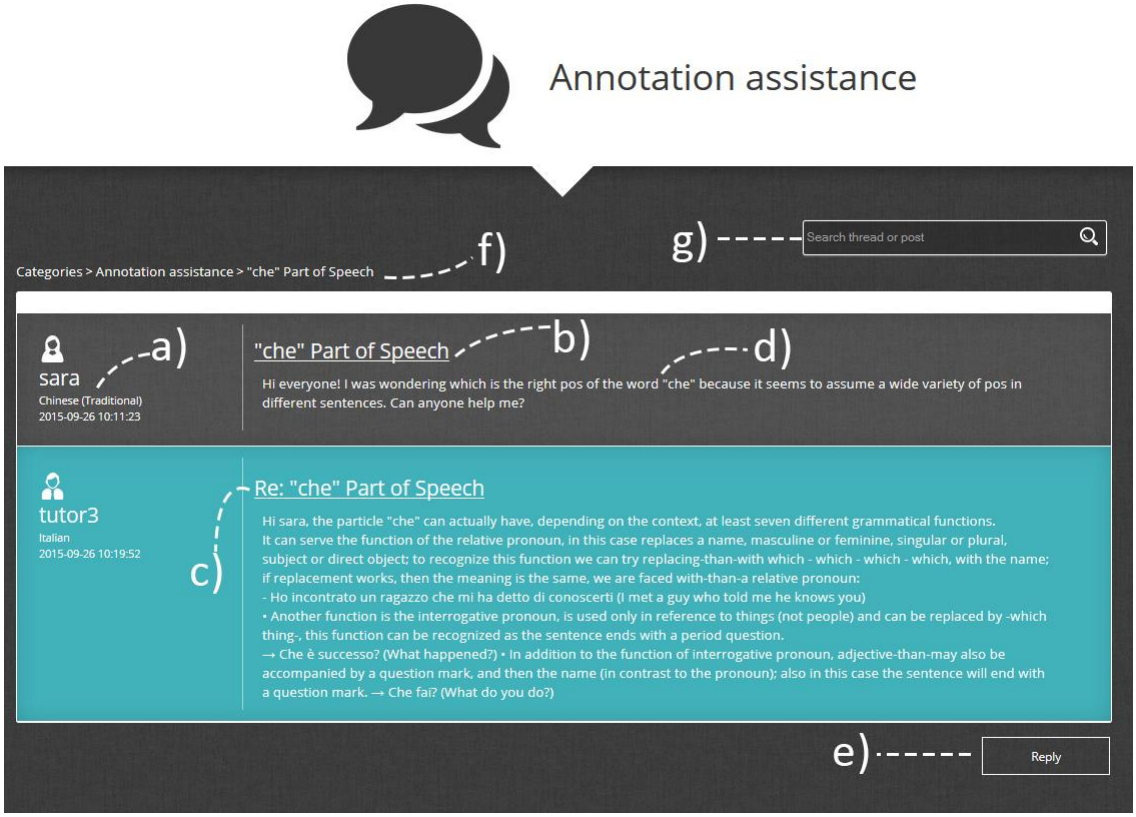

Figura 34 Messaggi di un thread

4.10 I tutorial

In ogni sezione è inoltre presente un tasto [?] tramite cui viene mostrato un tutorial circa la pagina corrente. Il tutorial è strutturato in una o più immagini che possono essere scorse a piacimento. Tali immagini riportano uno screenshot della sezione corrente arricchito da informazioni riguardanti i componenti salienti della pagina e le interazioni possibili. Tali tutorial sono importanti per orientare l'utente e guidarlo a compiere i task in modo corretto. Si rivela ad esempio particolarmente utile nella sezione *revision*: poiché per lo svolgimento dell'operazione di revisione i comandi a disposizione sono molti, il tutorial aiuterà l'utente *tutor* ad utilizzarli nel modo giusto.

5 L'applicazione web: ciò che l'utente non vede

In questo capitolo verrà descritta la struttura delle directory contenenti i file che costituiscono l'applicazione.

Verrà poi mostrato un esempio di come avviene, a livello di codice, il passaggio di dati tra *presentation tier*, *application tier* e *data tier*. Tale esempio sarà di aiuto alla comprensione delle operazioni che in linea di massima avvengono in background durante l'utilizzo dell'applicazione spiegato nel capitolo precedente.

5.1 Struttura delle directory

Tutte le cartelle e i file che compongono l'applicazione web si trovano all'interno della directory c:/*wamp/www*, radice delle applicazioni web sviluppate con la piattaforma Wamp. Al suo interno vi sono poi altre 5 cartelle:

- PHP: contiene tutti gli script PHP richiamati da Java Script;
- CSS: contiene i fogli di stile di ogni pagina HTML;
- js: contiene tutti gli script JavaScript implementati nonché i JavaScript che implementano JWPlayer e alertify;
- data: all'interno di questa cartella si trovano altre 4 sottocartelle:
    - media: contiene tutti i video caricati dagli utenti;
    - exercises: contiene tutti i file relativi alle esercitazioni create dai *tutor* - i file contenuti in questa cartella possono quindi avere estensione .clzf (cloze facilitato), .clzo (cloze facilitato orale), post (part-of-speech tagging o .transc (trascrizione);
    - tutorCertificates: contiene le autocertificazioni allegate dai *tutor* al momento della registrazione;
    - transcriptionFiles: contiene le seguenti tipologie di file:
        - file generati dal processo di trascrizione di TASEe (v. 7.1.2) che hanno estensione .ssntxml;
        - file generati dal processo di annotazione di TASEe (v. 7.1.3) che hanno estensione .kaf;
        - file *SubRip* generati dal processo di sottotitolazione di TASEe (v. 7.1.5) che hanno estensione .srt;
        - file generati dal processo di creazione degli *editable* di TASEe (v. 7.1.6) che hanno estensione .edt;

Anche se questi formati verranno ampiamente trattati nel Capitolo 7, è necessario introdurre qui il formato *editable* in quanto fortemente connesso all'applicazione web. Si tratta di un file contenente la trascrizione integrale incapsulata in appositi tag HTML (elementi *span*) aventi determinati attributi che specificano le caratteristiche di ogni parola. Questo permette di prelevare il contenuto del file ed agganciarlo al DOM delle pagine HTML. Si veda ad esempio una porzione di *editable* che incapsula il verbo *può*.

```
<span class="V" posdescita="verbo" posdesceng="verb"
offset="126" start="1571" end="1601" punct=""
morphofeatita="modale, presente" morphofeateng="modal,
present tense" token="può">può</span>
```

L'attributo *class* di questi elementi riporta l'abbreviazione della part-of-speech di quella parola (in questo caso *V* per *verbo*). A queste classi è associata una regola nei fogli di stile per cui, una volta iniettato l'EDT all'interno della pagina HTML, ogni part-of-speech verrà colorata con il colore assegnatole. Il valore dell'attributo *start* (che indica il momento in cui all'interno del video tale parola viene pronunciata) sarà invece necessario per gestire l'interazione con il player: quando l'utente clicca su una parola tale valore viene recuperato e il cursore di riproduzione del video viene spostato in quel punto. Sarà inoltre necessario nella fase di creazione delle esercitazioni (v. 5.8.1) per identificare la porzione di trascrizione da cui l'utente *tutor* crea il testo dell'esercitazione.

- images: contiene tutte le immagini utilizzate nell'applicazione;
- logs: contiene dei file di log generati dagli script PHP. Ogni file di log è nominato con lo stesso titolo dello script PHP che lo genera, ed ha estensione .log.

È possibile generalizzare il funzionamento del l'applicazione osservando la Figura 35 che mostra l'architettura *three-tier* descritta nella Sezione 3.1, nel caso specifico della sezione *upload*. Ogni pagina HTML che compone l'interfaccia include un file Java Script e un foglio di stile CSS che hanno il suo stesso nome. Il Java Script gestisce le funzionalità previste da quella pagina nonché tutti gli eventi scatenati dall'interazione con l'utente. Ad esempio la pagina *upload.php* include il foglio di stile *upload.*CSS che definisce le regole di stile per tale pagina e lo script *upload.js* che invoca gli opportuni script

dell'*application tier* per gestire l'upload e il processing dei video (*upload_file.php*, *check_transcription.php*, *check_annotation.php* e così via). Analogamente la pagina *browse.php* include lo script *browse.js* che gestisce le funzionalità per la visualizzazione del video sottotitolato e chiama gli opportuni script dell'*application tier*. Stessa cosa per tutte le altre pagine. Inoltre ogni pagina HTML include un altro script, chiamato *global.js*, che si occupa della gestione di tutte le funzioni e gli eventi comuni a più di una pagina, come ad esempio gli eventi per l'interazione con il player (v. 5.6).

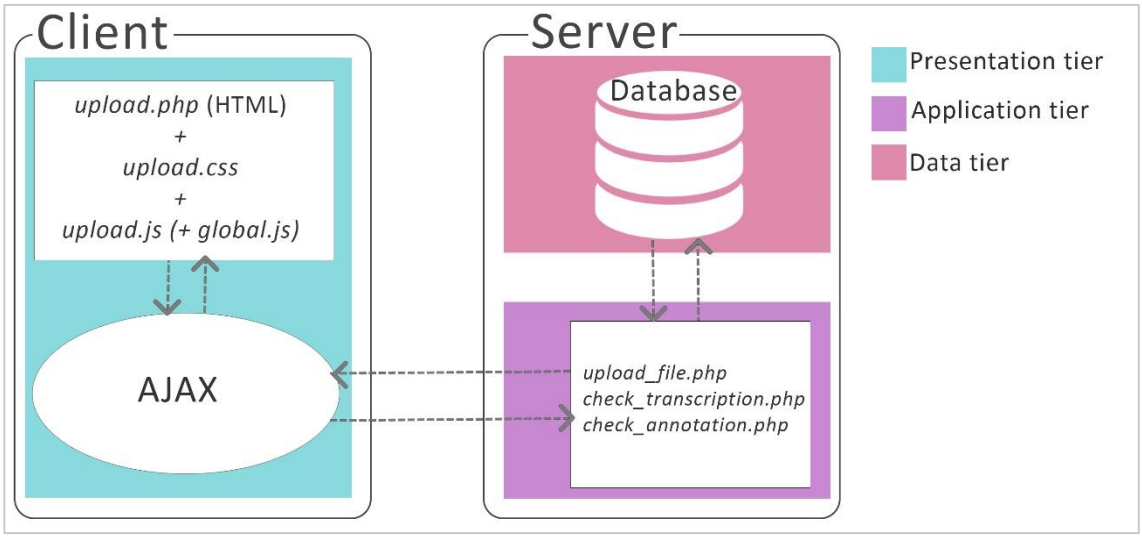

Figura 35 Architettura three-tier nel caso della sezione upload

5.2 Esempio di comunicazione *three-tier*

Prima di analizzare il funzionamento del processo che avviene in background mentre l'utente utilizza l'applicazione, si veda un esempio di quello che è un tipico passaggio di dati tra *presentation tier*, *application tier* e *data tier*.

Il codice che segue simula ciò che avviene in un'ipotetica operazione di login.

Il primo snippet mostra la gestione dell'evento *click* su un pulsante avente id *login.* Allo scatenarsi di questo evento vengono recuperati i valori di quelli che ipotizziamo essere due campi di input testuale aventi id rispettivamente *username* e *password*. Viene quindi invocata la funzione *checkLogin* a cui vengono passati i valori appena recuperati.

```
$("#login").click(function(){
    var username = $("#username").text();
    var password = $("# password").text();
    checkLogin(username, password);
});
```

La funzione *checkLogin* implementa una chiamata AJAX che specifica il tipo di richiesta HTTP (in questo caso POST), l'url dello script dell*'application tier* che si vuole invocare (in questo caso *check_login.php*, che risiede sotto la cartella PHP (v.5.1)), i dati che devono essere passati a tale script (in questo caso username e password) e il codice che deve essere eseguito quando lo script ha terminato la sua esecuzione.

```
function checkLogin(usernameValue, passwordValue){
   $.ajax({
      type:"POST",
      url: "PHP/check_login.php",
      data:{username:usernameValue, password:passwordValue},
      success:function(result){
      //codice da eseguire alla fine dell'esecuzione del PHP
      }
   });
)
```

Lo script *check_login.php* invocato recupera i valori passati tramite POST, si connette al database (tramite la libreria MySQLi) e recupera i valori dei campi *username* e *password* contenuti nella tabella *user* del database. Se lo username inserito dall'utente è presente nel database e la password associata corrisponde a quella inserita, lo script restituisce *true* altrimenti restituisce *false*.

```
<?PHP
  $username=$_POST["username"];
  $password=$_POST["password"];
  $db_servername="localhost";
  $db_username="root";
  $conn=new mysqli($db_servername, $db_username, "admin", "subitalian");
  $query = "SELECT password FROM user WHERE username='$username'";
  $result = mysqli_query($conn, $query);
  $logged=false;
  if($result->num_rows>0){
    if($result->fetch_assoc()["password"]==$password){
      $logged=true;
    } else{
      $logged=false;
    }
  } else {
    $logged=false;
  }
  return $logged;
?>
```

A questo punto viene eseguito il codice Java Script all'interno di *success* che controlla il valore del risultato a seconda del quale esegue una certa porzione di codice; in questo caso genera un alert con messaggi diversi a seconda che l'utente abbia inserito username e password corretti o meno.

```
//codice da eseguire alla fine dell'esecuzione del PHP
if(result==true){
    alert("Loggato!");
} else if(result==false) {
    alert("Combinazione username/password errata");
}
```

Nelle sezioni che seguono, quando ci si riferisce alla *gestione di un evento*, all'*invocazione di uno script PHP* e alla sua *risposta* al client si sottintendono dei passaggi che ricalcano la struttura di quelli appena esemplificati.

Di seguito, seguendo quelli che sono i passaggi spiegati durante la descrizione dell'applicazione front-end (v. 4), verrà spiegato ciò che accade in background.

5.3 Registrazione e login

L'utente che voglia registrarsi a SubItalian si recherà all'indirizzo della pagina principale dell'applicazione (http://servizi2.synthema.it/subitaliannew/index.php). Del codice PHP iniettato nella pagina controlla se è attiva o meno una sessione[19] utente. Nel caso non sia attiva, tale codice PHP reindirizza l'utente alla pagina *welcome.php* (Figura 11). Uno script *welcome.js* gestisce l'evento *click* sui due pulsanti (*Join Us* e *Login*) e, se viene cliccato il pulsante *Join Us* per procedere alla registrazione, *welcome.js* esegue il reindirizzamento alla pagina *joinusPre.php* (Figura 12). Lo script *joinuspre.js* gestisce l'evento *click* sui due elementi *div*. Il valore della classe di questi due *div* indica la rispettiva tipologia di utenza; in questo modo, al momento del click, lo script controlla il valore della classe e in base a questo procederà a reindirizzare l'utente alla pagina di registrazione vera e propria settando tale valore nel parametro *userType* dell'url di reindirizzamento, che dunque risulterà essere:

- http://servizi2.synthema.it/subitaliannew/joinus.php?**userType=l** per registrare un utente *learner*
- http://servizi2.synthema.it/subitaliannew/joinus.php?**userType=t** per registrare un utente *tutor*.

Il codice PHP all'interno della pagina *joinus.php* controlla il valore di tale parametro (userType) e, in base a questo, stamperà il codice HTML degli adeguati campi del form di registrazione che, come spiegato nella Sezione 4.1.3, differiscono dall'una all'altra tipologia di utente. Lo script *joinus.js* gestisce l'evento *click* sul pulsante di invio del form

[19] La gestione delle sessioni viene effettuata mediante la variabile globale $_SESSION e viene spiegata nella Sezione 5.3 dedicata al login.

allo scatenarsi del quale chiama lo script *join_user.php* a cui vengono passati tutti i valori dei campi. Lo script invocato si occupa di eseguire i seguenti passaggi:

- interroga il database per controllare che non sia già presente un utente (*learner* o *tutor* che sia) con lo username specificato;
- utilizza delle espressioni regolari per controllare la correttezza dell'indirizzo di posta inserito;
- esegue un confronto di stringhe sulle due password inserite per accertarsi che siano uguali;
- utilizzando degli operatori booleani sulla stringa della data di nascita immessa, controlla che sia corretta;
- nel caso di un utente *tutor* recupera il file allegato come autocertificazione e lo aggiunge nella cartella *tutorCertificates* (v. 5.1).

Nel caso in cui uno dei suddetti controlli fallisca, lo script reindirizza nuovamente l'utente nella pagina *joinus.php* specificando nel parametro *error* dell'url il codice di errore relativo al campo per cui tale errore si è verificato. La pagina contiene del codice PHP che controlla se tale campo è settato; in caso affermativo controlla il codice di errore e, a seconda di questo genera del codice Java Script che:

- genera un alert con l'appropriato messaggio di errore per notificare l'utente;
- cambia il CSS del campo per cui si è verificato l'errore, colorandone il bordo in rosso.

Nel caso in cui invece non vi siano errori, lo scritp *join_user.php* aggiunge un record alla tabella *user* del database settando il campo *confirmed* a 0. Dopodiché, mediante il server mail *sendmail* (v. 3.1), prepara ed invia una mail all'utente appena registrato. Al suo interno è presente un'url (su cui viene richiesto di cliccare) che si presenta come:

http://servizi2.synthema.it/subitaliannew/PHP/confirm_registration.php?***username***=*nome_inserito*

Alla sua invocazione, tale pagina recupera il valore del parametro *username* ed esegue una query al database per aggiornare il record relativo a tale utente, settando il campo *confirmed* a 1. A questo punto l'utente viene reindirizzato alla pagina di login tramite cui può finalmente accedere al portale.

La pagina di login ed i relativi script interrogano il database per controllare l'esistenza e la corrispondenza di username e password inseriti dall'utente, esattamente come visto nel codice di esempio (v. 5.2). Nel caso di esito positivo, lo script *login.php* memorizza username e tipologia dell'utente all'interno di una struttura associativa memorizzata nella variabile di sessione $_SESSION[20]. Tale struttura appare come:

```
$_SESSION[session_id()]["username"]=$username;
$_SESSION[session_id()]["userType"]=$userType;
```

La funzione *session_id()* recupera l'id della sessione corrente, univoco per ogni sessione. Salvando i dati (in questo caso *username* e *userType*) nella variabile di sessione, è fondamentale che vengano memorizzati all'interno di elementi che abbiano chiave univoca (appunto, l'id di sessione) al fine di garantire che non vi siano interferenze tra i dati di utenti loggati contemporaneamente. A questo punto l'utente viene reindirizzato alla pagina *index.php*.

Nel caso di esito negativo vengono generati un messaggio di errore e un ulteriore elemento di tipo input in cui viene chiesto all'utente di inserire il proprio indirizzo di posta nel caso abbia dimenticato i dati di accesso e desideri essere notificato tramite mail.

5.4 Sezione *index*

Nella pagina iniziale è iniettato del codice PHP che controlla il valore di *userType* all'interno della corrente struttura di sessione (settato al momento del login) e, a seconda di questo, rende visibili i pannelli adeguati, diversi per le due tipologie di utente (v. 4.3).

Lo script *index.js* gestisce soltanto gli eventi *click* sui vari pannelli. Gli elementi DOM di ogni pannello hanno settato come valore dell'attributo class la sezione a cui reindirizzano. Al momento del click, *index.js* controlla tale valore ed esegue un reindirizzamento alla sezione scelta.

5.5 Sezione *upload*

Per poter scegliere il video da caricare, l'interfaccia prevede un campo input di tipo *file* che consente di navigare nel proprio pc per selezionare il video desiderato. Lo script *upload.js* gestisce l'evento click sul pulsante *Upload*. Al verificarsi di tale evento viene invocato lo script *upload_file.php* che gestisce l'upload del file tramite la variabile

[20] http://PHP.net/manual/en/book.session.php

$_FILE[21] di PHP. Una volta terminato il caricamento del file sul server, tale variabile verrà riempita automaticamente da PHP con i seguenti valori:

```
$uploadedFilePath=$_FILE["file"]["tmp_name"];
$uploadedFileName=$_FILE["file"]["name"];
```

Il primo valore indica il path al file caricato da PHP in una cartella temporanea di default. Il secondo valore indica il nome di tale file (lo stesso nome che aveva nel pc dell'utente che ha eseguito l'upload).

Lo script dunque sposta il file nella cartella *media* (v. 5.1) sotto la root dell'applicazione. Terminato lo spostamento, *upload_file.php* procede nell'invocare il motore TASEe tramite la seguente riga di codice:

```
shell_exec("java -jar c:/subitalian/tase.jar 57_il_gladiatore
transcribe");
```

Grazie alla funzione *shell_exec*, PHP può simulare la linea di comando e dunque invocare il file JAR del motore TASEe specificando il nome del file corrente come primo parametro e *transcribe* come secondo parametro. La sintassi di tale comando verrà spiegata al momento in cui si tratterà del motore TASE (v.7).

Dopo aver invocato TASE, lo script aggiunge un record alla tabella *transcription* del database, settando (oltre a tutti gli altri campi previsti da tale tabella (v. 6.2)) il valore del campo *status* a *T*, indicante che il file da quel momento è in fase di trascrizione. Al momento dell'inserimento del record nel database, il nome del file caricato sul server viene reso univoco facendogli precedere una stringa che indica il valore incrementale del campo *id* della tabella transcription. Ad esempio, supponiamo di aver caricato il video con nome *il_gladiatore.mp4*; se l'ultimo record inserito nel database (ovvero l'ultimo video caricato) aveva id 56, il nostro video sarà inserito nel db con id 57 e dunque il suo nome univoco sarà *57_il_gladiatore.mp4*.

A questo punto, se l'upload è andato a buon fine ed è stato invocato TASEe, lo script PHP risponde positivamente a *upload.js* che inietta nell'HTML dell'interfaccia la label *Transcribing* (per notificare l'utente dello stato di avanzamento) e invoca un altro script: *check_transcription.php*. Questo script controlla ciclicamente la cartella *transcriptionFiles* (v.5.1) fintanto che non trova un file che indichi che è terminato il

[21] https://secure.php.net/manual/it/features.file-upload.post-method.php

processo di trascrizione ed è iniziato quello di annotazione (la generazione di tali file è spiegata nel Capitolo relativo al motore TASE (v. 7.1.2)). Nel caso in cui in fase di trascrizione si sia verificato un errore, lo script PHP aggiorna il record relativo a quel file settando il campo status a "E" (errore) e risponde negativamente a *upload.js* che stamperà un apposito messaggio di notifica. Se invece la trascrizione è andata a buon fine, lo script PHP aggiorna il record nella tabella *transcription* relativo a quel file settando il campo status a "A" che indica che TASEe è entrato nella fase di annotazione. Il client riceve quindi una risposta positiva, cambia il valore della label nell'interfaccia da *Transcribing* a *Annotating* e invoca il successivo script, *check_annotation.php*. Analogamente a *check_transcription.php*, questo script controlla ciclicamente la cartella transcriptionFiles finché non trova i file generati dal processo di annotazione (v. 7.1.3), in base a cui aggiorna il database e notifica il client che aggiorna la label. Stessa cosa avviene per controllare il processo di sottotitolazione e di creazione dei file *editable*.

Se tutto è andato a buon fine, l'ultimo file che viene generato dal motore TASEe all'interno della cartella *transcriptionFiles* è un EDT. Dunque, *upload.js* può reindirizzare l'utente alla sezione *browse.*

5.6 Sezione *browse*

Nella pagina *browse*.php, lo script *browse.js* si occupa di:

- inizializzare il player JWPlayer specificando il path del video appena caricato (all'interno della cartella *media*) e il path del relativo file SRT (all'interno della cartella *transcriptionFiles*);
- chiamare lo script *load_edt.php* passandogli il nome del video corrente; questo restituisce al client il contenuto del file EDT che viene iniettato nell'apposito contenitore HTML. Per ogni elemento *span browse.js* controlla se è presente l'attributo *punct*; se è presente ne viene recuperato il valore e nell'elemento *span* corrente viene innestato un elemento *punct* avente come valore il segno di punteggiatura recuperato (questo permette di visualizzare la punteggiatura senza aterare il contenuto degli elementi *span*);
- chiamare lo script *html_get_video_list.php* che interroga la tabella *transcription* del database per recuperare la lista di tutti i file caricati dall'attuale utente e quelli pubblici degli altri utenti con la relativa informazione sullo stato di revisione. Lo script parsa poi questa lista di titoli concatenandovi dei tag in modo da creare il codice HTML di un menu a tendina che viene restituito al client e iniettato

nell'apposito elemento DOM (agli elementi che contengono titoli di file revisionati viene aggiunta la classe *revisioned*; in questo modo, specificando una regola nel CSS, tali elementi appariranno colorati in verde);

- gestire l'interazione con il player e con la trascrizione, mediante la gestione dei seguenti eventi:

    - *ctrl* + *barra spaziatrice*: lo script chiama la funzione *onPlay*() o *onPause*() di JWPlayer per fermare o riavviare l'esecuzione del video;
    - *ctrl* + *freccia sinistra/ctrl* + *freccia destra*: lo script recupera l'attuale posizione del player (ovvero l'offset temporale in secondi) mediante la funzione *getTime*() di JWPlayer, sottrae o aggiunge tre secondi e chiama la funzione *seekTo*() di JWPlayer specificando l'offset temporale cui spostare il cursore di riproduzione;
    - *click* sul pulsante *See full transcription/Hide full transcription* (Figura 16.d): lo script mostra o nasconde l'elemento DOM in cui, a prescindere, è stato caricato l'EDT per mostrare la trascrizione integrale;
    - *ctrl* + *click* su una parola della trascrizione: lo script recupera l'attributo *start* della parola (contenuto nei metadati dell'EDT) su cui ha cliccato e chiama la funzione *seekTo()* passandogli l'offset temporale recuperato per muovere il cursore di riproduzione in tale posizione;

- gestire l'evento *click* su una parola della trascrizione: lo script genera il contenuto HTML dell'elemento riportante le informazioni morfologiche della parola cliccata (Figura 16.e); tali informazioni sono recuperate dagli attributi della parola stessa contenuti dei metadati dell'EDT (v.7.1.6). gestire l'evento click con il tasto destro del mouse su una parola della trascrizione: lo script recupera la parola che è stata cliccata, chiama uno script get_language_info.php che recupera lo username dell'utente corrente dalla variabile di sessione `$_SESSION[session_id()]["logged"]` (v. 5.3) e interroga la tabella *user* del database per recuperare l'id della lingua madre dell'utente. Recuperato tale id, fa una seconda chiamata al database, stavolta alla tabella *language* per ottenere la sigla (ad esempio *fr*) e descrizione (ad esempio *Francese*) corrispondenti all'id di tale lingua. La sigla della lingua viene utilizzata per effettuare una chiamata HTTP all'applicazione *https://www.translate.com/* che permette di effettuare richieste AJAX specificando lingua sorgente, lingua di traduzione e testo da tradurre. Ad

esempio la chiamata all'url *https://www.translate.com/translator/ajax_translate*, effettuata settando i parametri:

- *text_to_translate="vita"*
- *source_lang="it"*
- *translated_lang="fr"*

restituisce allo script il valore *vie*. Questo valore, insieme alla descrizione della lingua viene iniettato nell'elemento HTML che va a costituire il tooltip (Figura 16.f) dapprima nascosto.

- gestire l'evento *click* sulla label *Revisiona* (Figura 16.g)(solo nel caso di utenti *tutor*): reindirizza l'utente nella sezione di revisione specificando come parametro dell'url il nome del file che si stava visualizzando (http:// servizi2.synthema.it/subItalian/revision.php?***filename***=57_il_gladiatore.)
- gestire l'evento *click* sul pulsante *Request revision* (Figura 16.h) (solo per utenti *learner*)*:* quando un utente di tipo *learner* clicca su questo pulsante viene chiamato uno script *request_revision.php* a cui viene passato il nome del file corrente. Questo script utilizza il server di post *Sendmail* per inviare una mail a tutti gli utenti *tutor*. Il contenuto di tale mail sollecita i *tutor* a revisionare il file indicato, a cui è possibile accedere mediante il link che punta alla sezione *browse* e che ha come parametro *filename* il nome del file da revisionare: http:// servizi2.synthema.it/subItalian/revision.php?***filename***=57_il_gladiatore

5.7 Sezione *revision*

L'utente può giungere in questa sezione o dalla pagina *index*, o dalla pagina *browse*, o eventualmente tramite il link contenuto nella mail di richiesta di revisione (v. 5.6). Nel secondo e nel terzo caso, il codice PHP contenuto nella pagina HTML recupera l'attributo *filename* specificato nell'url generata e, in base a questo, genera il contenuto dinamico della pagina, caricando il video e la trascrizione del file specificato. Come nel caso di *browse.js*, *revision.js* si occupa di inizializzare il player e la trascrizione integrale, parsando l'EDT nello stesso modo spiegato nella sezione precedente. A seconda del tipo di revisione che l'utente *tutor* decide di effettuare (Figura 16.a), lo script *revision.js* carica l'EDT in un *div* differente a cui sono agganciati eventi diversi. Gli eventi gestiti dallo script per la revisione della **trascrizione** sono:

- *click* su una parola del testo: lo script crea un elemento HTML di tipo input e lo aggancia all'EDT inserendolo dopo l'elemento *span* della parola cliccata. A questo elemento input è agganciata una serie di eventi:
    - *tasto invio*: lo script sostituisce all'attuale valore dell'elemento cliccato quello inserito nel campo input (se è vuoto non apporta modifiche) e inserisce il vecchio valore in un attributo *oldValue* dell'elemento stesso in modo da poter ripristinare il valore precedente. A questo punto l'elemento input viene distrutto;
    - *tasto canc*: lo script distrugge l'elemento input senza apportare modifiche;
    - *freccia destra*: lo script si comporta come nel caso del tasto invio ma, piuttosto che distruggere il campo input, lo svuota e lo aggancia alla parola successiva.
- *click* destro su una parola del testo: lo script, recuperando le coordinate della posizione attuale del mouse, mostra il menu a tendina (creato manualmente con del codice HTML) dapprima nascosto che possiede quattro voci:
    - *delete this token*: lo script rimuove dall'EDT l'elemento cliccato;
    - *restore previous value*: nel caso in cui siano state modifiche alla parola cliccata, lo script recupera il valore dell'attributo *oldValue* generato al momento della modifica;
    - *remove punctuation*: nel caso in cui alla parola cliccata sia agganciato un elemento <*punct*> (v. 5.6), l'elemento viene rimosso;
    - *add punctuation*: lo script mostra un sottomenu che permette di scegliere il segno di punteggiatura da aggiungere alla parola selezionata. Cliccandovi viene dunque agganciato un elemento <*punct*> alla parola selezionato;
- *click* sul pulsante *Salva*: il comportamento dello script allo scatenarsi di questo evento varia a seconda che si stia revisionando la trascrizione o l'annotazione. In ogni caso però al testo recuperato vengono tolti gli elementi <*punct*> per ripristinare un EDT pulito.
    - Nel caso della trascrizione, *revision*.js chiama uno script *update_edt.php* che modifica il file EDT (dopo averne fatto un backup) recuperando il contenuto revisionato. Lo script procede dunque nell'invocare nuovamente il motore TASEe nello stesso modo con cui l'ha invocato per far partire la trascrizione (v. 5.5) ma stavolta il parametro passato in input

oltre al nome del file, sarà *annotate* (si ricorda che la spiegazione è fornita nel Capitolo 7). Il motore dunque salterà il passaggio di trascrizione passando a quello di annotazione e sottotitolazione del contenuto aggiornato.

  - Nel caso della revisione dell'annotazione, verrà invocato lo script *update_only_edt.php* che invocherà il motore TASEe specificando come secondo parametro *subtitle*. Questa volta TASEe aggiornerà soltanto il file SRT dei sottotitoli e l'EDT.

- *click* sul pulsante *File revisionato:* allo scatenarsi di questo evento *revision.js* chiama uno script *revisioned_file.php* che aggiorna al tabella *transcription* settando il valore del campo *status* del record riguardante il file corrente a *R* (revisionato). Tramite *Sendmail* viene dunque inviata una mail di notifica al proprietario del file riguardo all'avvenuta revisione. La lista dei file (Figura 17.a) viene nuovamente e agganciata al DOM mediante lo script *html_get_video_list.php* (v. 5.6) per mostrare in verde anche il file appena revisionato.

5.8 Sezione *exercises*

Le pagine dedicate agli esercizi sono due: *exercises_t.php* e *exercises_l.php*. La prima è dedicata alla creazione delle esercitazioni (la *t* che segue l'underscore sta a significare *tutor*), la seconda al loro svolgimento (la *l* dopo l'underscore sta a significare *learner*). Le due pagine vengono spiegate separatamente nelle due sezioni che seguono.

5.8.1 Processo di creazione delle esercitazioni

All'interno della pagina *exercises_t.php* è incluso ancora una volta lo script *html_get_video_list.php* (v. 5.6) che interroga il database per recuperare la lista dei video. I nomi dei file restituiti vengono quindi utilizzati per riempire il menu a tendina mostrato sulla sinistra dell'interfaccia (Figura 18.a). All'interno dello script *exercises_t.js* viene controllato qual è il file attualmente selezionato in tale menu a tendina e ne viene recuperato il nome che viene passato ad uno script *load_edt.php*. Questo script recupera il contenuto dell'EDT specificato e lo restituisce al client che lo stampa all'interno dell'apposita sezione testuale. Nel caso in cui l'utente sia giunto nella pagina *exercises_t.php* a partire dalla pagina principale (e dunque selezionando il pannello *esercizi*) la trascrizione mostrata sarà quella appartenente al video più recentemente inserito. Nel caso invece in cui l'utente sia giunto in questa pagina tramite la sezione di

revisione, l'attuale url conterrà l'attributo *filename* che specifica appunto il nome del file che si stava revisionando. All'interno di *exercises_t.php*, del codice PHP controlla se tale attributo è settato o meno; in caso positivo, viene generato del codice Java Script che simula la selezione nel menu a tendina di tale video e dunque viene richiamato lo script *load_edt.php* passando il nome del video *selezionato*.

Oltre allo script *html_get_video_list.php*, all'interno di questa pagina è incluso lo script *html_get_exercises_list.php*, che interroga il database per recuperare la lista degli esercizi precedentemente creati. Anche in questo caso i nomi dei file sono ordinati in modo decrescente per data d'inserimento e anche questo script incapsula i nomi dei file degli esercizi all'interno di appositi tag per generare un menu a tendina.

Lo script *exercises_t.js* gestisce i seguenti eventi:

- *change* sul menu a tendina: quando l'utente seleziona un video da questo menu a tendina, *exercises_t.js* chiama nuovamente lo script *load_edt.php* passando il nome del video selezionato. Dunque la sezione contenente la trascrizione sarà aggiornata con il contenuto dell'EDT del file selezionato.
- *change* sul menu a tendina per la selezione della tipologia di esercizio che si desidera creare. Quando l'utente cambia la tipologia di esercizio da creare, *exercises_l.js* rende visibile una certa porzione di elementi DOM entro i quali viene costruito l'esercizio della tipologia scelta. A seconda infatti della tipologia di esercizio, l'aspetto e le azioni che è possibile effettuare sulla porzione di testo cambiano. All'interno del *div* dedicato alla creazione degli esercizi di tipo cloze, ad esempio, è agganciato un evento *click* per cui viene cancellata (o ripristinata) la parola cliccata. Nel div dedicato alla creazione di esercizi post è invece gestito l'evento *click* sui *checkBox* delle PoS: quando il checkBox non è spuntato le parole hanno una classe *inactive* a cui è associata una regola nei CSS che le rende di colore grigio; quando il checkBox viene spuntato, tale classe viene rimossa e la parola assume il colore della relativa PoS.
- *change* sul menu a tendina per la selezione di un esercizio precedentemente creato. Quando si seleziona un esercizio tra quelli elencati nella lista, *exercises_l.js* chiama uno script *load_exercise.php* a cui passa il nome dell'esercizio. Questo script recupera il file EDT contenente l'esercizio selezionato e lo restituisce al client che lo inietta nella pagina a fianco della trascrizione integrale.

- *mousedown+mouseup* sulla trascrizione. Quando un utente preme il tasto sinistro del mouse su una parola della trascrizione (*mousedown*), *exercises_t.js* memorizza il valore dell'attributo *start* di tale elemento e resta in ascolto di un evento *mouseup* che si scatena quanto il pulsante viene rilasciato. Questi due eventi servono quindi per gestire la selezione della porzione della trascrizione che diventerà il corpo dell'esercitazione. Quando si verifica l'evento *mouseup* viene memorizzato il valore dell'attributo *end* dell'ultima parola selezionata e vengono recuperati tutti gli elementi *span* contenuti tra il primo e l'ultimo elemento della selezione. Questa porzione di EDT viene quindi iniettata nel *div* dedicato alla realizzazione dell'esercizio (Figura 18.e).
- *click* su una parola del testo di un esercizio di tipo cloze (l'esercizio post non ha bisogno della gestione di alcuna interazione) in fase di creazione. Al momento del *click*, *exercises_t.js* controlla se l'elemento *span* ha o meno la classe *canceled.* Se non la possiede significa che la parola non è ancora stata cancellata e dunque all'elemento *span* viene aggiunta tale classe. Viene quindi calcolata la lunghezza della parola cliccata e viene generata una stringa contenente il corrispondente numero di caratteri underscore che viene sostituita al valore dell'elemento cliccato (la parola è comunque memorizzata all'interno dell'attributo *token* dell'elemento *span*). Nel caso invece in cui la parola cliccata contenga già la classe *canceled,* il *click* viene interpretato come ripristino del valore precedente. Lo script dunque recupera il valore dell'attributo *token* dell'elemento *span* cliccato (contenente il valore precedente) e lo inserisce come valore dello stesso.
- *click* sul pulsante *Salva*: quando il *tutor* preme tale pulsante, *exercise_t.js* recupera il contenuto del *div* contenente l'EDT dell'esercizio creato e lo passa ad uno script *save_exercise.php* insieme ad altri parametri che variano a seconda della tipologia di esercizio che si sta salvando:
    - nel caso di un esercizio cloze facilitato o cloze orale facilitato vengono passati il valore del massimo intervallo di cancellazione, il valore del suo intervallo di decremento e il livello di difficoltà dell'esercizio;
    - nel caso di un esercizio di part-of-speech tagging, viene passato il valore delle PoS da annotare e il valore del livello di difficoltà;
    - nel caso di un esercizio di trascrizione viene passato soltanto il livello di difficoltà.

A questo punto lo script *save_exercise.php* aggiunge un record alla tabella *exercise* del database e salva nella cartella *exercise* (v. 5.1) un file con estensione *clzf*, *clzo*, *post* o *transc* a seconda della tipologia creata. Mentre i primi tre tipi di file hanno la stessa struttura di un EDT, il file dell'esercizio di trascrizione contiene il testo puro (come si vedrà nella sezione successiva la correzione di tale esercizio non richiede informazioni aggiuntive).

5.8.2 Processo di svolgimento e correzione delle esercitazioni

La pagina *exerices_l.php*, accessibile agli utenti *learner*, è divisa in tre sezioni e al momento dell'accesso soltanto la prima è resa visibile. In questa prima sezione l'utente può scegliere, tramite tre pannelli, le caratteristiche dell'esercizio che vuole svolgere (Figura 21).

Qui, lo script si occupa di gestire l'interazione con i tre pannelli: quello per la scelta della tipologia di esercizio, quello per la scelta della difficoltà e quello per la conferma delle impostazioni scelte. Lo script gestisce dunque i seguenti eventi:

- *click* sul primo pannello per la selezione della *tipologia* di esercizio (Figura 21.a). Quando l'utente clicca su uno dei tre pulsanti viene recuperato il valore di un attributo *exType* settato per ogni *div*, che indica appunto la tipologia dell'esercizio corrispondente; tale valore viene quindi memorizzato in una variabile *exType;*
- *click* sul secondo pannello per la selezione della *difficoltà* dell'esercizio (Figura 21.b). Quando l'utente clicca su uno dei tre pulsanti viene recuperato il valore che indica il grado di difficoltà dell'esercizio (tale informazione è memorizzata come valore di un attributo *difficulty* settato per ogni elemento DOM contenete il pulsante) che viene memorizzato in una variabile *difficulty*;
- *click* sul terzo pannello per la conferma delle *impostazioni* selezionate (Figura 21.c). Quando l'utente clicca sul pulsante *start* del terzo pannello, *exercises_l.js* fa una chiamata al server invocando uno script *check_exercise_availability.php* a cui passa i valori delle due variabili appena recuperate (*exType* e *difficulty*). Questo script per prima cosa interroga il database richiedendo la lista degli esercizi che rispondono alle caratteristiche specificate dall'utente. Se l'insieme restituito è vuoto, lo script risponde al client con un messaggio NOEX e *exercises_l.js* genera un *alert* per notificare l'utente che non sono ancora presenti esercizi con le caratteristiche da lui specificate. Se invece nel database è presente almeno un esercizio, viene eseguita una seconda query chiedendo nuovamente di

recuperare tutti gli esercizi che rispondano alle caratteristiche specificate e il cui id non sia presente nella tabella *performance* per l'utente attuale. In questo modo vengono recuperate soltanto le esercitazioni che non siano già state svolte da quell'utente. Se l'insieme restituito da questa interrogazione è vuoto significa che l'utente ha già risolto tutti gli esercizi presenti nel database e dunque *check_exercises_availability.php* risponde al client con un messaggio ALLDONE e il client genera un *prompt* attraverso il quale chiede all'utente se vuole comunque risolvere nuovamente uno degli esercizi già svolti o aspettare che gli utenti *tutor* creino nuove esercitazioni. Se l'utente decide di non continuare non viene svolta alcuna azione, altrimenti *exercises_l.js* procede all'invocazione di un secondo script, *perform_exercise.php* (invocato anche nel caso in cui vi siano ancora esercizi di quel tipo non ancora svolti da quell'utente, ovvero se la seconda interrogazione al database eseguita nello script *check_exercises_availability.php* restituisce almeno un record). A questo script vengono passati tipologia e difficoltà dell'esercizio e un parametro booleano *doAgain* che è settato *true* nel caso in cui si vogliano selezionare anche esercizi che l'utente ha già svolto (in seguito alla riposta ALLDONE) e *false* nel caso in cui si voglia selezionare l'esercitazione soltanto tra quelle non ancora svolte. Lo script interroga nuovamente il database sulla base delle informazioni passategli chiedendo di ordinare i risultati in modo casuale e di selezionare soltanto il primo record; questo permette di selezionare casualmente uno degli esercizi presenti nel database. Lo script recupera il contenuto dell'EDT dell'esercizio selezionato casualmente e risponde al client passandogli tale contenuto. *exercises_l.*js quindi nasconde la prima sezione (per la selezione delle caratteristiche) e mostra la seconda sezione, quella per lo svolgimento dell'esercizio.

In questa sezione l'utente può vedere il testo del proprio esercizio, ovvero il contenuto EDT restituito da *perform_exercise.php* che viene iniettato in un *div* HTML.

Lo script *exercises_l.js* gestisce eventi differenti a seconda della tipologia di esercizio che è stato scelto nella prima sezione.

- **Cloze orale facilitato.**

  Per quanto riguarda l'esercizio cloze orale facilitato (Figura 22), *exercises_l.js* gestisce:

- *click* sulle parole mancanti del testo dell'esercizio. Quando l'utente clicca su un elemento *span* contenente una parola mancante, lo script inserisce al posto di tale *span* un elemento *input* all'interno del quale l'utente può digitare del testo. Inoltre setta una variabile globale booleana *editing* con valore *true* che indica che il sistema è in fase di editing.
- *tasto enter*. Quando l'utente preme il tasto invio, lo script controlla il valore della variabile *editing*; se è *true* significa che l'utente stava inserendo una delle parole mancanti e quindi viene confermato il valore inserito all'interno dell'elemento *span* e a tale *span* viene aggiunto un attributo *inserted* per contrassegnare che quella parola mancante è stata inserita. La variabile *editing* viene a questo punto settata nuovamente a *false.*
- *tasto cancel*. Quando l'utente preme il tasto *esc*, lo script controlla il valore della variabile *editing*; se è *true* significa che l'utente stava inserendo una delle parole mancanti e quindi viene sganciato l'elemento input dallo *span* selezionato e settata la variabile *editing* nuovamente a *false* senza apportare alcuna modifica all'elemento che si stava inserendo;
- tutti gli eventi per l'interazione con il player spiegati nelle Sezione 5.6;
- *click* sul pulsante *Done.* Quando l'utente *learner* ha terminato l'esercizio e clicca sul pulsante *Done*, lo script controlla che tutti gli elementi *span* con classe *canceled* abbiano anche classe *inserted* (ovvero che tutte le parole mancanti siano state inserite). Se qualche parola non è stata inserita, lo script genera un messaggio di *alert* in cui suggerisce all'utente di completare l'esercizio per poterne vedere la correzione. Se invece tutte le parole sono state inserite, esegue una chiamata al server invocando uno script *correct_exercise.php* a cui passa il nome dell'esercitazione, la sua tipologia e l'EDT contenente l'esercizio svolto. Lo script memorizza in un file temporaneo il contenuto dell'esercizio svolto e recupera il contenuto del file EDT originale. Tramite la libreria *DOMDocument*[22] fornita dal PHP, tale script scorre il primo ed il secondo file confrontando i valori degli elementi *span* aventi classe *canceled* con i valori degli *span* corrispondenti nel file EDT originale (la corrispondenza, come verrà spiegato nel Capitolo 7, è verificata mediante l'attributo *offset* che indica la posizione di tale parola all'interno testo) e per ogni risposta esatta viene

[22] http://php.net/manual/en/class.domdocument.php

incrementata una variabile *correctAnswers*. Durante il parsing viene costruita una stringa che ricalca un EDT. In caso di risposta sbagliata vengono però aggiunti all'elemento *span* un attributo *rightAnswer*, che riporta quello che sarebbe stato il valore esatto, e un attributo *wrongAnswer* contenente il valore inserito dall'utente. Questi due attributi serviranno per formattare il testo della correzione all'interno della pagina HTML in modo che l'utente ne abbia una chiara visione (v. 4.7.4). Una volta parsato l'intero documento, viene calcolato il punteggio come percentuale di parole inserite correttamente sul totale delle parole cancellate. A questo punto si aggiunge un record alla tabella *performance* che riporta l'id dell'utente che ha svolto l'esercitazione, l'id dell'esercitazione, la data e il punteggio.

- **Cloze facilitato standard.**

  Per quanto riguarda il cloze facilitato standard (Figura 23) gli eventi gestiti sono:

- *trascinamento di una parol*a: alle parole poste in calce al testo dell'esercizio è agganciata la funzione *draggable*() fornita da JqueryUI. Nel momento in cui l'utente rilascia la parola che stava trascinando, viene controllata la posizione del mouse: se si trova al di sopra di un elemento *span* con classe *canceled*, il valore della parola trascinata viene settato come valore di tale elemento *span*. A tale elemento viene aggiunta una classe *inserted* per specificare che tale parola è stata inserita. La parola trascinata viene quindi resa invisibile.
- *click* su una parola del testo con classe *inserted:* se si clicca su una parola già inserita, viene recuperato il valore della parola tramite cui si ricerca la parola da trascinare (resa invisibile al momento dell'inserimento) e la si rende nuovamente visibile. Al posto dell'elemento cliccato verrà nuovamente aggiunto uno spazio (ovvero tanti underscore quante sono le lettere della parola mancante).
- *click* sul pulsante *Done.* L'evento viene gestito analogamente a quello visto per il cloze orale facilitato.
- **Esercizio PoST.**

  Per quanto riguarda l'esercizio PoST (Figura 25) gli eventi gestiti sono:

- *click* su una parola del testo dell'esercizio. Quando l'utente clicca su un elemento *span* del testo, *exercise_l.js* mostra il menu a tendina, dapprima nascosto, la cui posizione viene settata dinamicamente recuperando le coordinate dell'attuale posizione del mouse (dunque accanto all'elemento cliccato). Il menu contiene la lista delle PoS che l'esercizio prevede di inserire.

- *click* su una voce del menu contenente le PoS: quando l'utente clicca una delle voci mostrate nel menu contestuale, ne viene recuperata la classe (che indica appunto la PoS) che viene settata come classe dell'elemento *span* cliccato che di conseguenza assumerà il colore relativo alla PoS corrispondente (le parole non ancora annotate sono invece colorate in grigio). Il menu contestuale a questo punto viene nuovamente nascosto.
- *click* sul pulsante *Done.* Allo scatenarsi di questo evento *exercises_l.js* invoca lo script *correct_exercise.php* specificando stavolta la tipologia *post*. Anche in questo caso viene memorizzato il contenuto dell'esercizio svolto in un file temporaneo e viene recuperato il contenuto del file EDT originale. Tramite la libreria *DOMDocument* si scorrono gli elementi *span* di entrambi i file e se ne confrontano i valori delle classi. A differenza delle esercitazioni *cloze*, in cui gli errori potevano essere commessi soltanto laddove era richiesto di inserire le parole, in questo caso è possibile che l'utente annoti anche parole che non avrebbe dovuto poiché non appartenenti alla classe grammaticale richiesta. Quindi diventa necessario utilizzare una misura di valutazione che tenga in considerazione sia *falsi positivi* che *falsi negativi*: la misurazione scelta è l'*F1-measure*[23]. Ad esempio, se nell'esercitazione veniva richiesto di annotare i sostantivi, l'F1-measure verrà calcolata prendendo in considerazione il numero di tutti i sostantivi annotati come tali (*veri positivi*), il numero di sostantivi non annotati (*falsi negativi*) e le parole annotate come sostantivi ma appartenenti in realtà ad un'altra classe grammaticale (*falsi positivi*). Si calcolano quindi *precision* e *recall:*

$$precision = \frac{true\ positive}{true\ positive + false\ positive}$$

$$recall = \frac{true\ positive}{total}$$

E infine:

$$F1measure = \frac{precision}{recall}$$

Il risultato dell'F1-measure è un valore compreso tra 0 e 1 e, al fine di mantenersi conformi al punteggio percentuale degli esercizi precedenti, tale valore viene

[23] Wikipedia, voce *F1 score* https://en.wikipedia.org/wiki/F1_score

moltiplicato per 100. Si aggiunge quindi un record alla tabella *performance* che riporta l'id dell'utente che ha svolto l'esercitazione, l'id dell'esercitazione, la data e il punteggio ottenuto calcolato come media delle F1-measure di ogni PoS richiesta.

Come per gli esercizi *cloze,* per ogni risposta data viene costruita una stringa che ricalca un EDT; in questo caso ad ogni span contenente la risposta data verrà aggiunto un attributo *result* che può assumere tre valori: truePositive, falsePositive e falseNegative. In base a questi attributi, il CSS associato alla pagina potrà assegnare i relativi colori (v. 4.7.5.2).

- **Esercizio di trascrizione.**
  Per quanto riguarda lo svolgimento dell'esercizio di trascrizione (Figura 27), gli eventi gestiti sono:

a) tutti quelli che riguardano l'interazione con il player (v. 5.6);
b) il *click* sul pulsante *Done* per invocare lo script *correct_exercise.php* specificando *transcription* come tipologia di esercizio. In questo caso lo script recupera il testo della trascrizione originale dal file *.transc* (v. 5.8.1) e il testo della trascrizione realizzata dal *learner*. Questi vengono *splittati* e inseriti in due differenti array di token. Per misurare la somiglianza tra i due testi viene quindi calcolato il coseno di similitudine[24] tra i due array. Come l'F1-measure, anche il coseno è un valore che varia tra 0 e 1; pertanto, anche in questo caso, viene moltiplicato per 100. A questo punto viene aggiunto un nuovo record alla tabella *performance.*

Qualsiasi sia la tipologia di esercizio che è stato corretto, *correct_exercise.php* risponde al client comunicando il punteggio ottenuto e il testo contenente elementi riguardanti la correzione. Viene quindi nascosto il *div* contenente la sezione corrente e viene mostrata la terza sezione che mostra all'utente il punteggio ottenuto e la correzione dell'esercizio (Figura 24, Figura 26, Figura 28). Nel caso degli esercizi cloze e post, all'interno di un div viene iniettata la stringa contenente l'EDT dell'esercizio svolto e all'interno di ogni *span* che abbia l'attributo *wrongAnswer* vengono agganciati altri due elementi *span*: uno ha settato *wrongAnswer* come classe e ha come valore, appunto, la risposta data dall'utente; l'altro ha settato *rightAnswer* come valore della classe e il suo valore è la risposta corretta. A queste due classi sono associate delle regole nei fogli di stile per cui

[24] Wikipedia, voce *Coseno di similitudine* https://it.wikipedia.org/wiki/Coseno_di_similitudine

il testo della risposta sbagliata (*wrongAnswer*) viene mostrato di colore rosso, mentre quello della risposta corretta viene mostrato in verde. Nel caso invece di un esercizio di trascrizione vengono mostrati affiancati due div testuali contenenti rispettivamente la trascrizione originale e quella realizzata dal risolutore. Lo script *exercise_l.js* gestisce a questo punto altri due eventi:

- *click* sul pulsante *back:* cliccando su tale pulsante lo script nasconde la sezione corrente e mostra la prima sezione per permettere all'utente di selezionare nuovamente le caratteristiche dell'esercizio da svolgere;
- *click* sul pulsante *doAgain:* cliccando su tale pulsante lo script nasconde la sezione corrente e simula l'evento *click* sul pulsante di conferma della prima sezione (Figura 19.c) in modo da far partire nuovamente la catena di funzioni (viene dapprima controllata la disponibilità delle esercitazioni e poi selezionato un esercizio casuale).

5.9 Statistiche

Quando l'utente *tutor* clicca sul pannello relativo alle statistiche l'interfaccia mostra due pannelli: uno per visualizzare dati demografici, l'altro per ottenere informazioni riguardo all'andamento medio degli utenti *learner*.

In questa prima sezione della pagina *statistics.php*, lo script *statistics.js* gestisce soltanto l'evento *click* sui due pannelli. Al momento del click, lo script recupera la classe specificata per quel pannello, nasconde i pannelli stessi e mostra la porzione di HTML relativa alla tipologia di statistiche che l'utente ha selezionato.

Se si è scelto di visualizzare le statistiche demografiche, lo script *statistics.js* esegue le seguenti operazioni:

- chiama uno script *get_sex_age_data.php* che interroga la tabella *users* del database richiedendo tutti i record che abbiano tipologia di utente *learner* e la cui registrazione al portale sia stata confermata. Lo script inizializza un array associativo che ha due chiavi, *m* e *f* (maschio e femmina), i cui valori sono a loro volta array di nove elementi che rappresentano le fasce di età. Il valore di questi elementi (che inizialmente viene settato a zero) indicherà quindi il numero di utenti registrati rientranti in quella fascia d'età. Si esegue dunque un ciclo sulla lista dei risultati ottenuti dalla query effettuata e, controllando il sesso e la data, viene incrementato l'opportuno valore nell'array associativo. Una volta terminato

il ciclo sui risultati, l'array viene restituito al client. Lo script *statitics.js* a questo punto fa uso della libreria *HighCharts* per costruire in un apposito *div* il grafico di tipo *stacked columns* (Figura 30.a) di cui vengono specificate le etichette da assegnare alle categorie dell'asse delle ascisse (fasce di età), il titolo del grafico e i titoli degli assi. Vengono poi specificati i dati da inserire nel grafico, ovvero l'array associativo appena restituito dal server.

- chiama uno script *get_first_language_ripartition.php* che, come lo script precedente, interroga la tabella *users* del database richiedendo tutti i record che abbiano tipologia di utente *learner* e la cui registrazione al portale sia stata confermata. Lo script inizializza un array associativo le cui chiavi rappresenteranno gli id delle lingue e i valori il relativo numero di utenti iscritti. Si esegue un ciclo sulla lista dei risultati ottenuti dalla query effettuata e si controlla il valore del campo *first_language*; se esiste già un elemento nell'array associativo avente come chiave l'id della lingua del record corrente, il suo valore viene incrementato; in caso contrario tale elemento viene aggiunto all'array e inizializzato a 1. Una volta terminato il ciclo sui risultati, l'array viene restituito al client. Lo script *statitics.js* a questo punto fa uso della libreria *Highcharts* per costruire, in un apposito *div*, il grafico di tipo *pie chart* (Figura 30.b) di cui viene specificato il titolo e i dati da inserirvi vengono recuperati dall'array associativo appena restituito dal server. Alle fette di questo grafico viene inoltre agganciato un evento *click* al verificarsi del quale *statistics.js* recupera il valore del nome di quella categoria (ovvero il nome della lingua madre) e invoca il server chiamando lo script *get_sex_ripartition_for_first_language.php* a cui passa tale valore. Lo script interroga prima la tabella *language* per recuperare l'id del paese avente quella descrizione e poi esegue una query nella tabella *user* per recuperare tutti i record il cui *first_language_id* corrisponda all'id recuperato dalla query precedente. Viene inizializzato un array associativo avente "f" e "m" come chiavi e 0 come valori iniziali. Si esegue quindi un ciclo sui record restituiti e si incrementa il valore dell'elemento nell'array associativo avente come chiave il valore del campo *sex* del record corrente. Terminato il ciclo sui record, l'array viene restituito al client che ne i legge i valori per poterli mostrare nell'interfaccia: vengono recuperate le attuali coordinate della posizione del cursore per posizionare e rendere visibile un *div*, dapprima nascosto, in cui vengono iniettati i valori dell'array (numero di maschi e numero di femmine per lingua). A questo

elemento è agganciato un evento *mouseleave* per cui quando l'utente sposta il cursore al di fuori di esso, viene reso nuovamente invisibile.

Se invece si è scelto di visualizzare le statistiche relative alle performance ottenute dagli utenti learner nel corso del tempo, e dunque viene aperta la sezione con il grafico di tipo spline (Figura 31), lo script statistics.js svolge le funzioni elencate di seguito.

- Chiama uno script *get_performances_in_time.php* che dapprima interroga la tabella *user* per ottenere la lista degli utenti di tipo learner, di cui recupera id e sesso. Scorre poi la lista di tali utenti interrogando la tabella *performance* per recuperare l'elenco delle esercitazioni svolte da ciascun utente; di ogni record vengono recuperati i campi relativi a data di svolgimento e punteggio ottenuto. A questo script php viene passato un parametro *exType* che specifica la tipologia di esercizio di cui si vogliono vedere le performance. Per default tale valore è nullo e dunque vengono recuperati indistintamente tutti i tipi di esercizio. Selezionando invece uno dei *radio button* presenti sopra il grafico, il parametro *exType* assume il valore della tipologia di esercizio selezionata e dunque la tabella performance sarà interrogata filtrando i risultati ai soli esercizi di tale tipologia. Una volta recuperata la lista di esercitazioni svolte, viene costruito un array multidimensionale che contiene la lista degli utenti learner che a sua volta contiene il sesso dell'utente e un array riportante la lista di coppie *data di svolgimento-punteggio* che costituiscono ogni esercitazione. L'array costruito viene restituito al client che lo utilizza per costruire il grafico di tipo *spline* (Figura 31) (messo a disposizione dalla libreria *Highstock*). Durante la costruzione di tale grafico, per ogni linea disegnata, viene controllato il sesso dell'utente per poter settare lo stile della linea corrente (tratteggiata per le femmine, punteggiata per i maschi).

5.10 Il forum

All'interno della pagina *forum.php*, lo script *forum.js* si occupa di gestire le funzionalità elencate di seguito.

- Chiama uno script *load_categories.php* che interroga:
  - la tabella *category* del database per recuperare le categorie attualmente presenti (id e descrizione);

    - la tabella thread del database per recuperare il numero di thread per ogni categoria e titolo, autore e data dell'ultimo thread aperto.

I risultati vengono restituiti al client che costruisce dinamicamente l'HTML riportante la lista delle categorie (Figura 32). Ad ogni voce della lista è agganciato un evento *click* allo scatenarsi del quale *forum.js* recupera l'id della categoria cliccata, nasconde l'attuale sezione e mostra la sezione riportante l'insieme dei thread.

- Chiama uno script *load_threads.php* che interroga:
    - la tabella *thread* per ottenere il titolo di ogni thread appartenente alla categoria precedentemente cliccata (di cui si ha l'id);
    - la tabella *post* per ottenere il numero di post lasciati per ogni thread oltre all'autore e alla data dell'ultimo post lasciato.

In questa sezione, oltre all'evento *click* su una voce della lista, è gestito l'evento *click* sul pulsante *Open thread*; allo scatenarsi di tale evento viene mostrato un form per l'apertura di un nuovo thread. Compilato il form, lo script *forum.js* recupera il valore dei campi inseriti dall'utente, ovvero *titolo* e *messaggio* del nuovo thread, e chiama uno script *open_thread.php* che, dopo aver recuperato l'id dell'utente attualmente loggato (tramite la variabile di sessione - v.5.3) aggiunge un record alla tabella *thread*. Aggiunge inoltre alla tabella *post* un record che avrà come *threadId* quello del thread appena creato e avrà il campo *isTopic* settato a 1 per indicare che si tratta del messaggio di apertura del thread.

- Chiama uno script *load_posts.php* quando viene cliccata una delle voci della lista dei thread. Lo script interroga la tabella *post* per ottenere (in ordine cronologico) autore (e sua lingua madre), data di creazione, titolo e corpo del messaggio di ogni post appartenente a quel thread (di cui si ha l'id). I risultati vengono restituiti al client che genera la lista HTML (Figura 33). Viene inoltre gestito l'evento *click* sul pulsante *Reply*, allo scatenarsi del quale viene mostrato il form che permette di digitare il messaggio di risposta. Compilato il form, lo script *forum.js* recupera il contenuto del messaggio appena inserito e chiama uno script *add_post.php* che aggiunge un record alla tabella *post* specificando il *threadId* corrente e settando questa volta il campo *isTopic* a 0 (poiché si tratta di un messaggio di risposta al topic).

- Gestisce la funzionalità di ricerca restando in ascolto di un evento *click* o della pressione del pulsante invio da tastiera. Allo scatenarsi di uno dei due eventi *forum.js* recupera il valore dal campo testuale (ovvero le parole chiave inserite per la ricerca) e lo passa ad uno script *search_keywords.php* che interroga le tabelle *post* e *thread* per recuperare tutti i post che abbiano tali parole chiave all'interno del titolo o nel corpo del messaggio. I risultati vengono restituiti al client che costruisce la lista HTML dei risultati.

6 Il Database

Come si è visto nel Capitolo 3, il database costituisce il terzo livello dell'architettura *three-tier* dell'applicazione web: il *data tier*. Dunque, il database MySql memorizza e gestisce i dati dell'applicazione. La struttura del database è mostrata nello schema logico riportato in Figura 36. Come si evince da tale schema, le tabelle che compongono il database sono otto: *user*, *transcription*, *exercise*, *performance*, *language*, *category*, *Thread e Post* (le ultime tre riguardanti la gestione dei dati del forum).

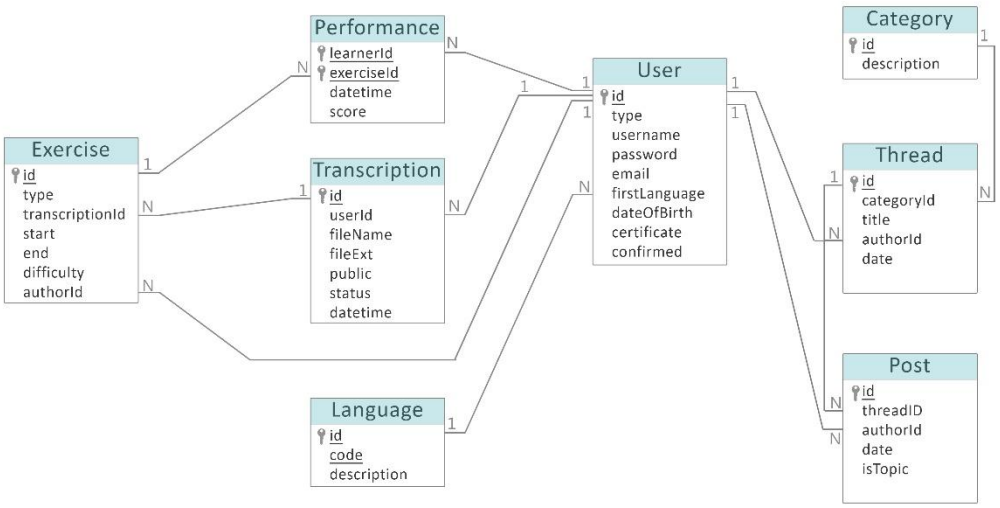

Figura 36 Schema logico del database

Di seguito viene descritta la funzione delle varie tabelle e i campi di cui sono composte. Per ogni tabella viene inoltre specificato in quale occasione vengono inseriti e aggiornati i record e in quali casi avviene la sua interrogazione.

6.1 La tabella *user*

La tabella *user* contiene informazioni riguardo agli utenti registrati al portale. I record vengono creati al momento della registrazione (v. 5.3) dell'utente e vengono aggiornati (vedi campo *confirmed*) quando viene confermata l'iscrizione tramite il link che l'utente riceve via mail. L'interrogazione a questa tabella avviene al momento del login per recuperare username e password e verificare la corrispondenza con i quelli inseriti dall'utente e in varie altre occasioni; ad esempio ogni qual volta ci sia bisogno di recuperare il valore dello username dell'utente a partire dal suo id. Questa tabella contiene i seguenti campi:

- id: un intero che identifica univocamente ogni utente;

- *type*: stringa di lunghezza 1 che indica la tipologia di utente. Può assumere i seguenti valori:

a. L: indica l'utente di tipo *learner*
b. T: indica l'utente di tipo *tutor*

- *username*: lo username scelto dall'utente al momento della registrazione (anche questo è univoco per ogni utente);
- *password*: stringa alfanumerica scelta dall'utente al momento della registrazione;
- *email*: l'indirizzo email inserito dall'utente al momento della registrazione;
- *firstLanguage* (solo per utenti *learner*; per gli utenti *tutor* tale campo sarà lasciato vuoto): intero che identifica univocamente il paese di provenienza dell'utente (il nome della lingua è specificato nella tabella *language);*
- *sex* (solo per gli utenti *learner*): stringa di lunghezza 1 che indica il sesso specificato dall'utente. Può assumere i seguenti valori:
  - M: sesso maschile
  - F: sesso femminile
- *dateOfBirth* (solo per utenti *learner*; per gli utenti *tutor* tale campo sarà lasciato vuoto): campo di tipo *DATE* che indica la data di nascita dell'utente;
- *firstLanguage* (solo per utenti *learner*; per gli utenti *tutor* tale campo sarà lasciato vuoto): indica l'id della lingua madre dell'utente - la descrizione della lingua è riportata nella tabella *language* (v. 6.5);
- *certificate* (solo per utenti *tutor*; per gli utenti *learner* tale campo sarà lasciato vuoto): nome del file per la certificazione di madrelingua, allegato dall'utente al momento della registrazione;
- *confirmed:* valore booleano che indica se l'utente ha confermato o meno la registrazione. Può assumere i seguenti valori:

a. 0: l'utente non ha ancora confermato la sua registrazione
b. 1: l'utente ha confermato la registrazione e risulta correttamente iscritto al portale

6.2 La tabella *transcription*

Tale tabella contiene le informazioni relative ai file caricati (sia da parte di utenti *tutor* che da parte di utenti *learner*). I record vengono inseriti in questa tabella non appena termina il trasferimento sul server del file scelto nella sezione upload (v. 5.5) e vengono

aggiornati ogni volta che il processo termina una fase (vedi campo *status).* L'interrogazione viene eseguita quando, nelle sezioni *browse*, *revision* e *exercises* viene mostrato il menu a tendina contenente la lista dei file, propri e pubblici, caricati sul server e processati. La tabella *transcription* contiene i seguenti campi:

- *id*: intero auto incrementale che identifica univocamente ogni trascrizione;
- *user*Id: id dell'utente che ha caricato il file;
- *filename*: stringa che identifica il nome del file caricato (senza estensione);
- *ext*: stringa che specifica l'estensione del file multimediale caricato;
- *public*: valore booleano che indica se l'utente ha reso pubblico (e qindi visibile da tutti gli altri utenti) o meno il file caricato. Può assumere i seguenti valori:
    - 0: l'utente non ha reso pubblico il video caricato
    - 1: l'utente ha reso pubblico il video caricato
- *status*: stringa di lunghezza 1 che indica lo stato attuale del processo. Può assumere i seguenti valori:
    - T: significa che il video è stato caricato sul server ed è in fase di trascrizione;
    - A: significa che la fase di trascrizione è stata portata a termine correttamente e il testo prodotto è in fase di annotazione;
    - S: significa che la fase di annotazione è stata portata a termine correttamente ed è iniziata la fase di creazione dei sottotitoli e dei file EDT;
    - E: significa che, in una delle tre fasi precedenti, il processo è andato in errore;
    - R: significa che il file è stato revisionato da un *tutor*.
- datetime: campo di tipo DATETIME che indica data e orario in cui il file è stato caricato sul server. Il formato di tale campo è quello previsto dallo standard MySql ovvero *yyyy-mm-dd hh:mm:ss*.

6.3 La tabella *exercise*

Questa tabella contiene le informazioni riguardo agli esercizi creati. I record vengono inseriti o aggiornati in questa tabella nel momento in cui il *tutor* nella sezione dedicata alla creazione delle esercitazioni (v. 5.8.1) salva l'esercizio creato. Questa tabella viene interrogata nella sezione *exercises* (per *tutor*) per creare il contenuto del menu a tendina

riportante la lista degli esercizi precedentemente creati (Figura 18.f). I campi specificati sono:

- *id*: un intero auto incrementale che identifica univocamente ogni esercitazione;
- *type*: stringa di lunghezza 4 che indica la tipologia di esercizio creato. Può assumere i seguenti valori:
    - CFO: indica che l'esercizio creato è un cloze orale facilitato
    - CF: indica che l'esercizio è un cloze facilitato
    - POST: indica che l'esercizio è un part-of-speech tagging
    - TRANSC: indica un esercizio di trascrizione
- *transcriptionId*: intero che identifica l'id del video dalla cui trascrizione è stato creato l'esercizio.
- *start*: indica il timestamp iniziale della prima parola che compone il testo l'esercitazione;
- *end*: indica il timestamp finale dell'ultima parola che compone il testo dell'esercitazione;
- *difficulty*: indica il livello di difficoltà con cui il *tutor* ha valutato l'esercitazione. Può assumere i seguenti valori:
    - 0: indica un esercizio facile
    - 1: indica un esercizio mediamente facile
    - 2: indica un esercizio difficile
- *authorId*: riporta l'id del *tutor* che ha creato l'esercizio.

6.4 La tabella *performance*

La tabella *performance* riporta le informazioni riguardo alle esercitazioni svolte dai singoli utenti. I record di questa tabella vengono inseriti ogni qual volta un utente *learner* svolga un'esercitazione e non vengono mai aggiornati. La tabella viene interrogata nella sezione *statistics* per recuperare i punteggi raggiunti dagli utenti. I campi contenuti in questa tabella sono:

- *idLearner*: intero che indica l'id dell'utente *learner* che ha svolto l'esercizio;
- *idExercise*: intero che indica l'id dell'esercitazione svolta;
- *datetime*: campo di tipo DATETIME che riporta la data e l'orario in cui è stato svolto l'esercizio;

- *score*: campo di tipo float che indica il punteggio (espresso in percentuale) ottenuto dall'utente per quell'esercizio.

6.5 La tabella *languages*

Questa tabella contiene le informazioni relative alle lingua che l'utente *learner* può selezionare al momento della registrazione. I record sono stati inseriti a priori da uno script *fill_languages_table.php* tramite cui sono stati letti i nomi dei paesi da un file testuale e inseriti incrementando automaticamente l'id. L'interrogazione a questa tabella avviene, appunto, in fase di registrazione per riempire il menu a tendina per che permette di selezionare il luogo di nascita. La tabella contiene dunque i campi:

- *id*: intero che identifica univocamente il paese;
- *code*: stringa che identifica la sigla della lingua associata all'id;
- *description*: stringa contenente il nome della lingua associata all'id.

6.6 Le tabelle del *forum*

6.6.1 La tabella *category*

La tabella *category* contiene informazioni relative alle categorie di discussione attualmente presenti nel database e viene interrogata nella prima sezione del forum (v. 5.10) che mostra, appunto, la lista delle categorie di discussione.

I campi che compongono tale tabella sono:

- *id:* intero che identifica univocamente la categoria;
- *description:* stringa contenente la descrizione della categoria.

6.6.2 La tabella *thread*

Questa tabella contiene informazioni riguardo alle discussioni aperte per ogni categoria. Viene interrogata nella seconda sezione del forum (v. 4.9) in cui, una volta selezionata la categoria di discussione, viene mostrata la lista dei thread aperti in tale categoria. I campi di questa tabella sono:

- *id*: intero che identifica univocamente il thread;
- *idCategory*: intero che indica l'id della categoria di appartenenza;
- *title*: stringa che riporta il titolo del thread;
- *idAuthor*: intero che identifica l'autore che ha creato il thread;
- *date*: campo di tipo DATETIME che indica data e ora di creazione del thread.

6.7 La tabella *post*

La tabella post contiene informazioni riguardo ai post lasciati per un certo thread. Viene interrogata nella terza sezione del forum (v. 4.9) in cui, una volta selezionata la categoria di discussione, viene mostrata lista dei thread aperti in tale categoria. I campi di questa tabella sono:

1. *id:* intero che identifica univocamente il post;
2. *threadId:* intero che identifica l'id del thread di cui fa parte il post;
3. *isTopic:* valore booleano che indica se il post è il primo messaggio del thread (topic);
4. *authorId:* intero che identifica l'utente che ha generato il post;
5. *date:* campo di tipo DATETIME che indica data e orario in cui il post è stato scritto;
6. *content:* stringa che identifica il testo del post.

7 TASEe

Il motore TASEe (Transcription, Annotation, Subtitling and Editing engine) risiede sulla stessa macchina su cui risiede l'applicazione web e viene invocato dagli script PHP che costituiscono *l'application tier*. Questo componente, come suggerito dall'acronimo, gestisce i processi di trascrizione, annotazione, sottotitolazione e generazione dei file *editable*. Si tratta di un software realizzato in *Java* ed esportato in formato JAR in modo che possa essere invocato mediante linea di comando. La funzione *shell_exec* di PHP (v. 5.5) permette appunto di simulare la linea di comando e dunque di invocare il motore. Il JAR, chiamato appunto *TASEe.jar*, prende in input due parametri: il nome univoco del file da processare e una parola chiave che specifica qual è la prima operazione da eseguire all'interno dell'intero workflow del programma. Infatti, come si può notare dalla schematizzazione mostrata in Figura 41, al momento del suo avvio il programma interpreta il comando che gli viene dato in input e in base a questo inizia la sua esecuzione in punti diversi. Il workflow completo, che prevede quindi trascrizione, annotazione, sottotitolazione e creazione dei file *editable*, viene eseguito non appena terminato l'upload del video da parte dell'utente specificando l'opzione *transcribe* in input (v. 5.5).

Il programma è strutturato in tre package (Figura 37):

- ASR (Figura 38): contiene le classi che implementano le *chiamate* da effettuare ai web service di SpeechScribe per effettuare la trascrizione;
- Annotator (Figura 39): contiene le classi che gestiscono le *chiamate* ai web service di OpeNER;
- SubItalian (Figura 40): contiene le classi che gestiscono l'intero workflow e implementano i metodi per la creazione degli SRT e dei file *editable*.

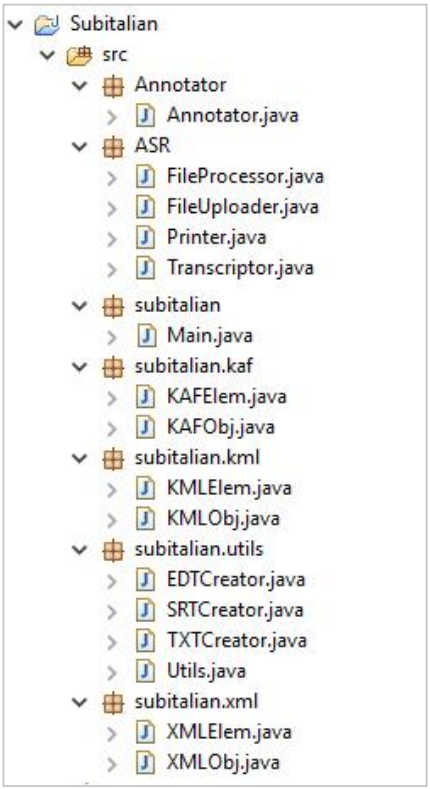

Figura 37 Struttura del progetto Java

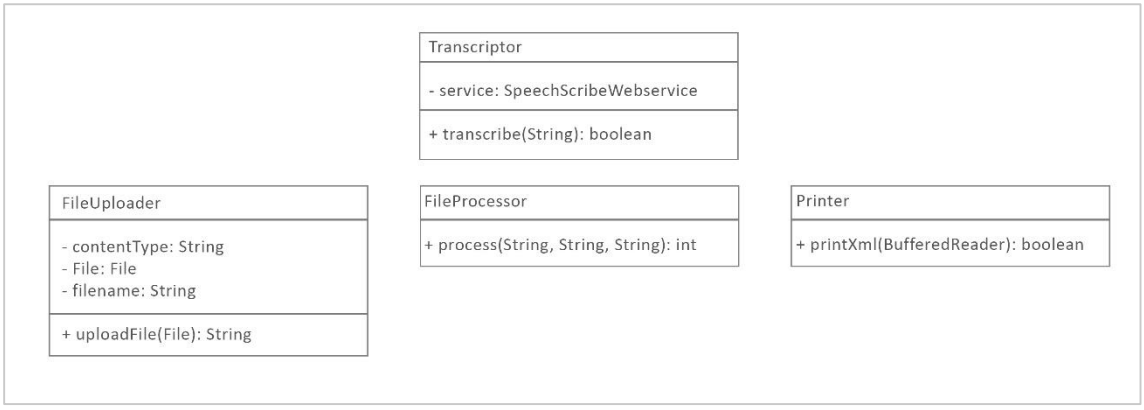

Figura 38 Package ASR

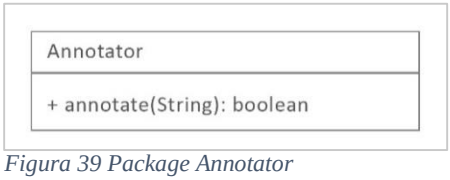

Figura 39 Package Annotator

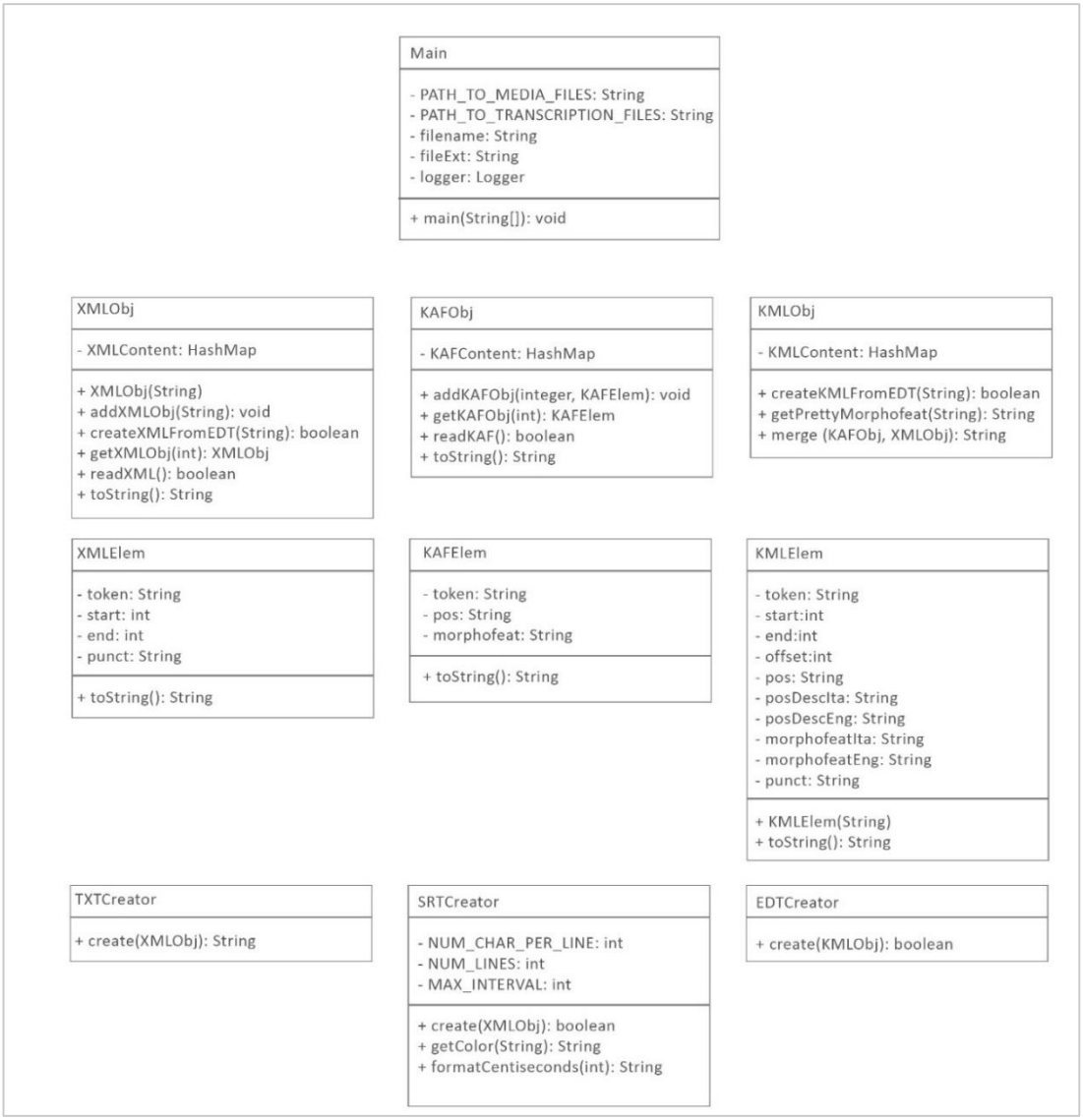

Figura 40 Package SubItalian

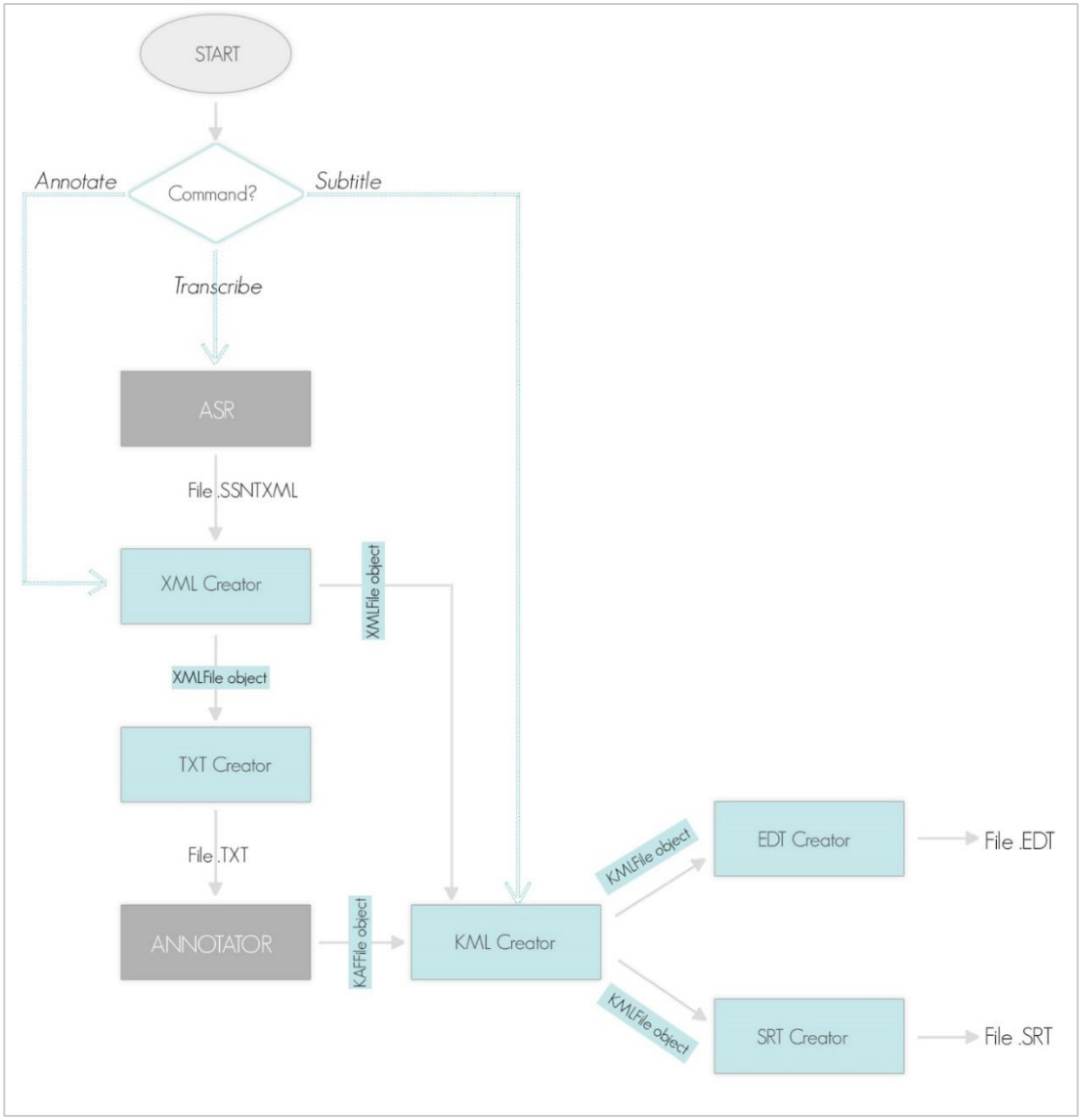

Figura 41 Workflow TASEe

7.1.1 Il *main*

La classe *main* all'interno del package SubItalian è la classe che viene eseguita al momento dell'invocazione del JAR[25] da parte dello script PHP. È qui dunque che si controllano i parametri forniti in input e si decide quali operazioni eseguire. Per spiegare il processo in modo completo, si assuma che il comando specificato in input sia *transcribe*; il primo task da eseguire è quindi la trascrizione. All'interno del *main* si leggono inoltre alcune opzioni di configurazione specificate in un file di configurazione (*congif.prop*); tra queste, il path della directory contenente i file video caricati dagli utenti e il path alla directory contenente i vari file testuali generati (xml, kaf, srt, edt) (v. 5.1). I

[25] https://it.wikipedia.org/wiki/JAR_%28formato_di_file%29

riferimenti assoluti ai file vengono quindi costruiti concatenando a questi path il nome del file passato in input.

7.1.2 Il processo di trascrizione

Il package ASR gestisce la prima fase del workflow completo, ovvero la trascrizione, implementando i web service messi a disposizione da SpeechScribe (v. 3.2.1). Il *main* invoca il metodo *transcribe* della classe *Transcriptor* contenuta in questo package, specificando in input il nome del file da trascrivere. Questo metodo si occupa quindi di eseguire una sequenza di chiamate ai web service che permettono appunto di ottenere la trascrizione desiderata. Per prima cosa il video che ha terminato l'upload sul server deve essere dato in pasto ai web service; viene dunque eseguito il metodo *UploadFile* della classe *FileUploader* che, mediante il path al file specificato, costruisce un oggetto Java di tipo *File*[26] e, dopo aver settato una serie di parametri, esegue la funzione messa a disposizione dai web service, tramite cui tale file viene preso in gestione dall'ASR. Se l'esito è positivo, il metodo *uploadFile* restituisce una stringa *mediaHash* che identifica univocamente il file appena caricato. A questo punto è possibile iniziare la trascrizione vera e propria gestita dal metodo *process* della classe *FileProcessor*. Tale metodo ha due parametri: il mediaHash restituito da *uploadFile* e la lingua del video appena caricato; nel nostro caso, l'italiano. Dopo aver settato una serie di parametri, tale metodo esegue una chiamata al web service che dà inizio alla trascrizione. Non appena effettuata tale chiamata, viene eseguito un ciclo che interroga il web service riguardo allo stato di avanzamento del processo di trascrizione. Tale ciclo termina quando il processo di trascrizione è stato portato a termine si verifica un errore. In questo secondo caso il metodo *transcribe* restituisce un valore *false* e viene interrotta l'esecuzione del programma dopo aver generato, all'interno della cartella *transcriptionFiles* (v.5.1), un file di errore avente la seguente forma:

[nome_del_file].TRANSCRIPTION.ERROR

In caso di successo invece *uploadFile* restituisce un valore positivo e si prosegue invocando il metodo *printXML* della classe *Printer*. Tale metodo, specificando ancora una volta il *mediaHash* del file in questione, interroga il web service richiedendo l'output dell'audio trascritto che viene letto e stampato, all'interno della cartella transcriptionFiles, sul file:

[26] http://docs.oracle.com/javase/7/docs/api/java/io/File.HTML

[nome_del_file].SSNTXML

La trascrizione ha una struttura xml contenente diversi metadati. Si veda di seguito una porzione di tale xml.

```
<TranscriptSegment>
  <TranscriptWordList>
      <Word conf="0.8824" end="461" norm="un'" start="450">un'</Word>
      <Word conf="0.959" end="527" norm="alimentazione" start="462">alimentazione</Word>
      <Word conf="0.9749" end="582" norm="corretta" start="528">corretta</Word>
      <Word conf="0.7592" end="613" norm="può" start="583">può</Word>
      <Word conf="0.9142" end="640" norm="fare" start="614">fare</Word>
      <Word conf="0.8937" end="644" norm="a" start="641">a</Word>
      <Word conf="0.9606" end="677" norm="meno" start="645">meno</Word>
      <Word conf="0.9711" end="699" norm="della" start="678">della</Word>
      <Word conf="0.2291" end="753" norm="carne"start="700">carne</Word>
  </TranscriptWordList>
</TranscriptSegment>
```

Nello snippet è mostrato un esempio di segmento di trascrizione; l'intera trascrizione è infatti suddivisa in una serie di segmenti, identificati dall'ASR a seconda delle pause o della sintassi dei discorsi riconosciuti nell'audio. Tale suddivisione non è però di interesse per i nostri scopi. Ciò che è sufficiente tenere in considerazione sono i nodi *figli* dei vari segmenti, ovvero gli elementi <Word> che contengono ogni singolo token (un elemento di punteggiatura non è memorizzato come valore di tale elemento, bensì come attributo della parola che segue) e hanno i seguenti attributi:

- *conf*: indica la probabilità che tale token sia stato riconosciuto correttamente;
- ***start***: indica il timestamp (espresso in centesimi di secondo) del momento in cui la parola inizia ad essere pronunciata;
- ***end***: indica il timestamp (espresso in centesimi di secondo) del momento in cui termina la parola;
- *norm*: contiene il valore del token trascritto con la prima lettera maiuscola nei casi in cui si trovi a inizio frase;
- **punct**: tale attributo è presente solo se dopo il token si trova un elemento di punteggiatura.

Ai fini della realizzazione dei file di sottotitolazione, le due informazioni essenziali da estrarre da questo file, oltre ai valori dei token, sono i due timestamp (inizio e fine parola). Per poter mantenere e gestire questa informazione, necessaria negli step successivi, viene chiamato dal *main* il metodo *readXML* della classe *XMLObj* che legge, parsa e memorizza

l'intera trascrizione in un'istanza di *XMLObj*, contenente un oggetto di tipo *HashMap* in cui la chiave di ogni *entry* rappresenta l'offset testuale dell'elemento (non essendo esplicito nel file SSNTXML, tale valore viene calcolato incrementando una variabile che tiene conto della lunghezza dei token incontrati) e il valore rappresenta un oggetto di tipo *XMLElem.* Tale oggetto memorizza dunque tutte le informazioni relative a un token (*start*, *end*, *punct* e *token*). In Figura 42 è mostrata la struttura di tali oggetti.

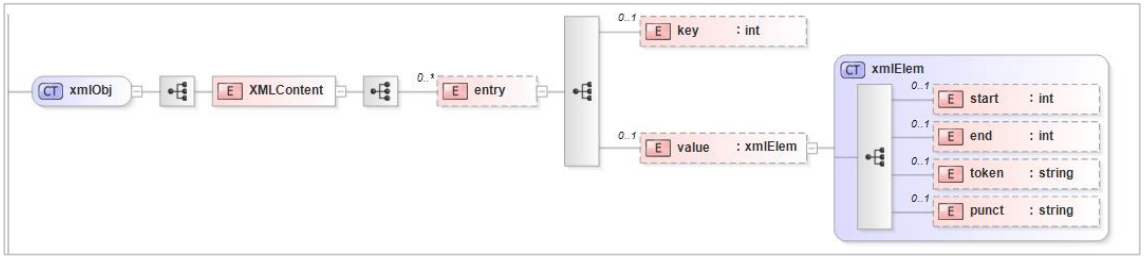

Figura 42 Struttura di un XMLObj

Se il metodo *readXML* va a buon fine, il *main* riceve una risposta positiva e l'esecuzione del programma può procedere; in caso contrario verrà generato un file di errore del tipo:

[nome_del_file].READXML.ERROR

Nel frattempo lo script *check_transcription.php* (v. 5.5) che, dopo aver invocato TASEe, inizia a controllare la cartella *transcriptionFiles*, troverà uno dei due file generati in caso di errore (ovvero in fase di trascrizione o in fase di lettura dell'xml) o il file contenente la trascrizione avente estensione SSNTXML.

7.1.3 Il processo di annotazione

Una volta terminato con successo il processo di trascrizione, il *main* può proseguire la sua esecuzione procedendo con il processo di annotazione. Come spiegato nella Sezione 3.2.2, i web service di annotazione prendono in input un testo puro; per questo, a partire dall'xml della trascrizione appena creato, è necessario generare un file *txt*. Il *main* invoca quindi il metodo *createTxt* della classe *Utils*, passandogli in input l'oggetto *XMLObj* creato (v. 7.1.2). Tale metodo parsa l'oggetto ed estrae i valori della variabile *token* di ogni *XMLElem* che vengono mano a mano stampati sul file plain-text. Una volta creato il file *txt* si possono invocare i web service di OpeNER; il *main* chiama quindi il metodo *annotate* della classe *Annotator* specificando il path del file *txt* appena creato. Le chiamate ai web service devono essere eseguite mediante il protocollo HTTP eseguendo delle richieste di tipo POST ad un'url che viene costruita concatenando all'indirizzo di base

dei web service (http://opener.olery.com) il nome del componente che si vuole invocare (ad esempio http://opener.olery.com/tokenizer). Si deve eseguire una chiamata per ogni componente che si desidera includere nel processo di annotazione. L'output di ogni chiamata è strutturato in modo tale da essere interpretato dal componente successivo; infatti, fatta eccezione per il primo componente che prende in input il testo puro, i componenti successivi prendono in input l'output del componente precedente. Nel nostro caso sarà sufficiente eseguire due chiamate: quella al *tokenizer* e quella al POS tagger. Dunque il metodo *annotate* fa una prima chiamata POST a http://opener.olery.com/tokenizer specificando come parametri in input il contenuto del *txt* appena generato e la sua lingua. L'output viene dunque passato come parametro in input alla seconda richiesta POST http://opener.olery.com/pos-tagger mediante la quale viene annotato il testo. In caso di errore durante l'annotazione, viene generato il file:

[nome_del_file].ANNOTATOR.ERROR

In caso di successo viene invece generato un file di output strutturato secondo il formato KAF[5], definito in sede al progetto OpeNER. Qualsiasi siano i componenti che si decide di integrare nel processo di annotazione, l'output restituito sarà sempre strutturato secondo questo formato. Si tratta di una struttura xml contenente tanti livelli di annotazione quanti sono i componenti che si è deciso di includere nel processo. Nel nostro caso, dunque, il KAF avrà due livelli: quello generato dal tokenizzatore e quello generato dal POS tagger.

Il tokenizzatore, spezzettando il file di testo puro, fornisce il primo livello di annotazione. All'interno del KAF, ogni token è inserito all'interno di un elemento <wf> (*word form*) contenente diversi attributi.

```
<wf wid="w2" sent="1" para="1" offset="0" length="3">un'</wf>
<wf wid="w3" sent="1" para="1" offset="4" length="13">alimentazione</wf>
<wf wid="w4" sent="1" para="1" offset="18" length="8">corretta</wf>
<wf wid="w5" sent="1" para="1" offset="27" length="3">può</wf>
<wf wid="w6" sent="1" para="1" offset="31" length="4">fare</wf>
<wf wid="w7" sent="1" para="1" offset="36" length="1">a</wf>
<wf wid="w8" sent="1" para="1" offset="38" length="4">meno</wf>
<wf wid="w9" sent="1" para="1" offset="43" length="5">della</wf>
<wf wid="w10" sent="1" para="1" offset="49" length="5">carne</wf>
```

Come mostrato nello snippet, gli attributi di tale elemento sono[27]:

[27] In neretto gli attributi necessari per questo progetto

- **wid**: identifica univocamente il token e servirà per agganciare il token corrente ai successivi livelli di annotazione;
- sent: indica il numero della frase a cui appartiene il token;
- para: indica il numero del paragrafo a cui appartiene il token;
- **offset**: indica l'offset testuale del token (questo attributo è fondamentale dal momento in cui la posizione della prima lettera del token all'interno del testo è l'unico elemento che permette di collegare tale token al corrispondente token dell'oggetto XMLElem);
- length: indica il numero di lettere di cui è composto il token.

Nello stesso file troviamo i risultati del POS tagger, che genera gli elementi <term>:

```
<!--'-->
<term tid="t2" type="close" lemma="'" pos="Q" morphofeat="CONJ">
    <target id="w2" />
</term>
<!--alimentazione-->
<term tid="t3" type="open" lemma="alimentazione" pos="N" morphofeat="NOU~CS">
    <target id="w3" />
</term>
<!--corretta-->
<term tid="t4" type="open" lemma="correggere" pos="V" morphofeat="VMA~PA">
    <target id="w4" />
</term>
<!--può-->
<term tid="t5" type="open" lemma="potere" pos="V" morphofeat="VMO~RE">
    <target id="w5" />
</term>
<!--fare-->
<term tid="t6" type="open" lemma="fare" pos="V" morphofeat="VMA~IN">
    <target id="w6" />
</term>
<!--a-->
<term tid="t7" type="close" lemma="a" pos="P" morphofeat="PREP">
    <target id="w7" />
</term>
<!--meno-->
<term tid="t8" type="open" lemma="meno" pos="N" morphofeat="NOU~CA">
    <target id="w8" />
</term>
<!--della-->
<term tid="t9" type="close" lemma="di" pos="P" morphofeat="PREP">
    <target id="w9" />
</term>
<!--carne-->
<term tid="t10" type="open" lemma="carne" pos="N" morphofeat="NOU~CS">
    <target id="w10" />
</term>
```

Gli attributi di tale elemento sono:

- *tid*: identifica univocamente l'elemento;
- *type*: indica la classe di appartenenza del token analizzato (aperta o chiusa);
- *lemma*: contiene la forma base del token;
- ***pos***: contiene la parte del discorso identificata per quel token (v. 3.2.2);
- ***morphofeat***: contiene informazioni morfologiche aggiuntive (v. 3.2.2).

L'elemento <term> è legato all'elemento <wf> generato dal tokenizer mediante l'attributo *wid* del suo elemento figlio <target>.

Similmente al metodo *readXml* della classe *XMLObj*, il metodo *readKAF* della classe *KAFObj* parsa questo file creando un'istanza di tale oggetto. Il metodo analizza dapprima il livello generato dal tokenizzatore e, mediante l'identificatore *wid*, recupera gli attributi *pos* e *morphofeat* del livello generato dal *POS tagger*. Con queste informazioni viene creato l'oggetto *KAFElem*.

La classe *KAFObj* è strutturata nello stesso modo della classe *XMLObj*, ed è quindi costituita da un oggetto di tipo *HashMap* in cui le chiavi rappresentano l'offset testuale di ogni token (stavolta esplicito) che costituisce l'annotazione e i valori associati a tali chiavi contengono i rispettivi elementi *KAFElem*, che memorizzano tutti gli attributi del token. Tale struttura si presenta come mostra la Figura 43.

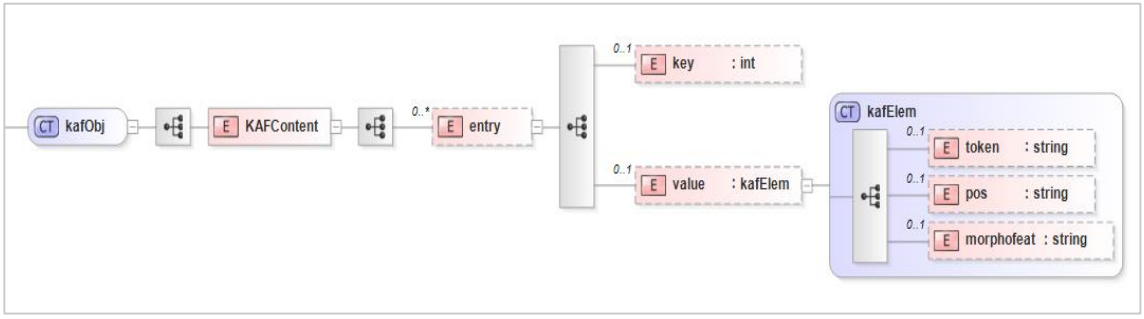

Figura 43 Struttura di un KAFObj

Se non si verifica alcun errore grazie agli oggetti *XMLObj* e *KAFObj*, si è in possesso di tutte le informazioni necessarie (sia quelle temporali fornite dal trascrittore, sia quelle linguistiche fornite dall'annotatore) per generare i sottotitoli annotati.

In caso di errore durante la fase di annotazione viene invece generato il file:

[nome_del_file].READKAF.ERROR

Nel frattempo lo script *check_annotation.php* (v. 5.5) che, dopo l'esito positivo della trascrizione, inizia a controllare la cartella *transcriptionFiles*, troverà o il file generato in caso di errore o il file contenente la trascrizione e avente estensione KAF.

7.1.4 Il *merge*

Una volta terminato correttamente il processo di annotazione, il *main* riceve una risposta positiva dal metodo *annotator*. A questo punto dunque, per poter fornire all'utente la sottotitolazione annotata linguisticamente, è necessario *mergiare* l'oggetto *XMLObj* con l'oggetto *KAFObj* appena costruito. Viene dunque creato un terzo oggetto, chiamato *KMLObj* (dall'unione di **K**af e x**ML**) che, ricalcando la struttura degli altri due oggetti, contiene una lista di oggetti *KMLElem*.

Per costruire tale oggetto è necessario parsare gli oggetti contenenti trascrizione e annotazione cercando, per ogni token della trascrizione, il corrispondente token dell'annotazione facendo affidamento sull'offset testuale (esplicito nell'annotazione e calcolato per la trascrizione). Questo valore è infatti un identificatore univoco di ogni token ed è lo stesso per i token appartenenti agli oggetti di tipo *XMLElem* e quelli appartenenti agli oggetti di tipo *KAFElem*. Ad esempio l'oggetto *XMLElem* che memorizza il token *corretta* sarà strutturato come segue:

```
XMLObj[4]://offset
   XMLElem:
      token:”alimentazione”
      start:462
      end:527
      puct:”,”
```

L'oggetto *KAFElem* che memorizza il medesimo token sarà strutturato come segue:

```
KAFObj[4]://offset
   KAFElem:
      token:” alimentazione”
      pos:”N”
      morphofeat=”NOU~CS”
```

Dunque sarà possibile creare l'oggetto *KMLElem* prelevando le informazioni *pos* e *morphofeat* dall'oggetto *KAFElem* e le informazioni *start, end* e *punct* dall'oggetto *XMLElem*.

A questo scopo il metodo *merge* della classe *KMLObj* prende in input un oggetto *XMLObj* e un oggetto *KAFObj* ed esegue un ciclo sugli oggetti *KAFElem* che compongono l'oggetto *KAFObj*[28] leggendone la chiave, ovvero l'offset testuale, e cercando tra gli oggetti *KAFElem* dell'*HashMap KAFObj* quello che abbia lo stesso offset come chiave. Prima di procedere alla creazione del nuovo oggetto, il metodo *merge* chiama delle funzioni che leggono i valori delle variabili *pos* e *morphofeat* e ne creano una descrizione *human readable* sia in italiano che in inglese. Ad esempio, continuando a vedere l'esempio del token *vita*: a partire dal valore dell'attributo *pos*, ovvero *N*, vengono create le stringhe *Nome* e *Noun* che costituiranno il valore di due nuovi attributi dell'oggetto *KMLElem* ovvero, rispettivamente, *posDescIta* e *posDescEng*. Stessa cosa avviene per l'attributo *morphofeat*: a partire da *NOU~CS*, verranno creati un attributo *morphofeatIta* e un attributo *morphofeatEng* che conterranno rispettivamente le stringhe *Nome di cosa singolare* e *Noun of thing singular*. A questo punto si può procedere all'effettiva creazione dell'oggetto *KMLElem* che contiene gli attributi utili dell'uno e dell'altro formato, più le stringhe *human readable* appena descritte. L'oggetto che memorizza ogni token si presenta quindi come mostrato in Figura 44. Il token *vita*, incapsulato all'interno dell'oggetto *KMLElem*, avrà la seguente struttura:

```
KMLObj[4]://offset
   KMLElem:
      token:"corretta"
      start:462
      end:527
      punct:","
      pos:"N"
      morphofeat="NOU~CS"
```

[28] Tale scelta è arbitraria, si sarebbe potuto effettuare il ciclo anche sugli oggetti XMLElem dell''oggetto XMLElem.

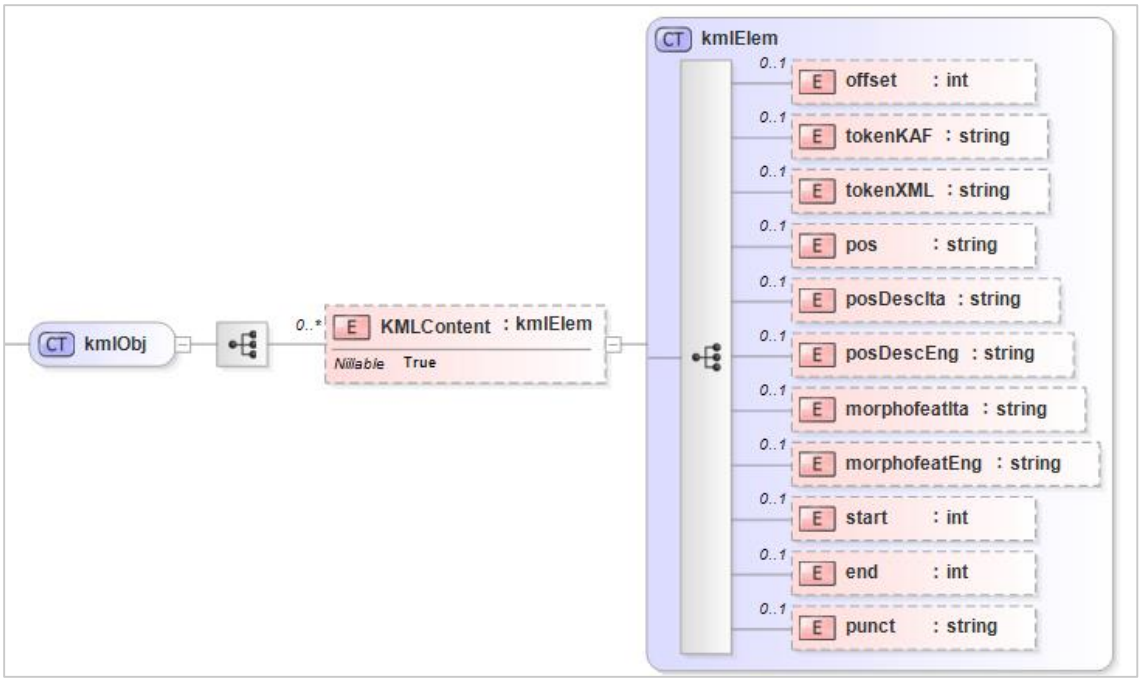

Figura 44 Struttura di un KMLObj

Una volta scorsi tutti gli oggetti *XMLElem*, otterremo un file *KMLObj* contenente tutte le informazioni possibili da cui sarà possibile costruire sia i sottotitoli che i file *editable*.

7.1.5 Il processo di sottotitolazione

Una volta che il metodo *merge* ha risposto positivamente al *main*, si procede alla creazione dei sottotitoli annotati. Il formato scelto per la sottotitolazione è il SubRip[63]. Un file SubRip ha estensione *.srt* ed è composto da una serie di *segmenti* che identificano la porzione di testo da mostrare a video. Ognuno di questi segmenti è formato da tre elementi fondamentali:

a) l'offset del sottotitolo: è un numero sequenziale che indica il numero del sottotitolo;
b) la stringa dei timestamp: si presenta nel formato `hh:mm:ss,fff → hh:mm:ss,fff` (ovvero, *ore:minuti:secondi,millisecondi*) e indica l'inizio e la fine rispettivamente della prima e dell'ultima parola di quel sottotitolo;
c) le linee id testo: rappresentano il testo del sottotitolo che verrà stampato a video.

Le informazioni fornite dall'oggetto *KMLObj* sono sufficienti per ricostruire tale file e il metodo *createSRT* della classe *Utils* è stato implementato a tale scopo. Per prima cosa vengono letti dal file di configurazione (*config.prop*) tre valori fondamentali:

- il numero di linee che si desidera stampare per ogni sottotitolo (per SubItalian è stato stabilito il valore 2);
- il numero massimo di caratteri che ogni linea può contenere (per SubItalian è stato stabilito il valore 40);
- il massimo intervallo di tempo che può intercorrere tra una parola e l'altra all'interno dello stesso sottotitolo (per SubItalian è stato stabilito un secondo); se si supera tale intervallo si inizia un nuovo sottotitolo.

Tenendo in considerazione questi due parametri, il metodo *createSRT*, che prende in input il *KMLObj* appena creato, esegue un ciclo sulla lista dei suoi elementi *KMLElem*. Si eseguono poi due cicli annidati; quello più esterno controlla il numero di linee che sono state stampate e vi esce ogni qualvolta si sia raggiunto il numero massimo di righe per sottotitolo stabilito. Quello più interno controlla invece il numero di caratteri che sono stati stampati per ogni linea; dunque, si esce da tale ciclo ogni qualvolta venga raggiunto il numero massimo di caratteri stampabili per linea stabilito. Quello che sarà il contenuto del file SubRip viene costruito concatenando ad una stringa, dapprima vuota, gli elementi che si devono stampare. Ogni volta che si inizia a costruire un nuovo segmento, si stampa in una stringa, che chiamiamo A, l'offset di tale segmento (più il carattere *a capo*) e in una stringa, che chiamiamo B, lo start della prima parola (che una funzione si occupa di trasformare nel formato richiesto dal SubRip, ovvero hh:mm:ss,fff).

Ogni parola del ciclo (ovvero ogni valore della variabile token di ogni elemento *KMLElem*) invece viene concatenata ad una terza stringa, che chiamiamo C, aggiungendo il carattere *a capo* quando si esce dal ciclo più interno (quello che controlla il numero di caratteri per riga). Alla fine di ogni ciclo più esterno (quello che controlla il numero di righe), si aggiunge alla stringa B (contenente fino a questo momento soltanto il timestamp di inizio segmento) il timestamp di fine segmento, recuperando l'attributo *end* dell'ultimo elemento *KMLElem* incontrato.

Si procede poi concatenando A, B e C per ottenere il segmento completo da concatenare alla stringa globale.

La stampa della stringa C è in realtà più articolata. Per poter mostrare i sottotitoli con colori differenti, è infatti necessario aggiungere alcuni metadati. Il formato SRT supporta la formattazione dei sottotitoli e permette di incapsulare una parola all'interno di un elemento <font> di cui è possibile specificare alcuni attributi: tra questi vi è il colore del testo. Il metodo createSRT controlla dunque il valore *pos* dell'elemento *KMLElem* e in base a questo assegna un certo valore all'attributo color.

Una volta scorso tutto l'oggetto *KMLObj*, la stringa globale viene stampata su un file che ha lo stesso nome del file video ed estensione .SRT.A quel punto i sottotitoli sono pronti per essere visualizzati dal player; di seguito ne è riportato un esempio.

```
1

00:00:04,620 --> 00:00:10,460
<font color=#666666>Un' </font>
<font color=#00cc33>alimentazione </font>
<font color=#ff0066>corretta </font>
<font color=#ff0066>può </font><font color=#ff0066>fare </font>
<font color=#9900ff>a </font>


<font color=#00cc33>meno </font>
<font color=#9900ff>della </font>
<font color=#00cc33>carne? </font>
<font color=#ff0066>Deve </font>
<font color=#ff0066>far </font>
<font color=#9900ff>a </font>
<font color=#00cc33>meno </font>
<font color=#9900ff>della </font>


2
00:00:10,710 --> 00:00:16,070
<font color=#00cc33>carne. </font>
<font color=#666666>La </font>
<font color=#00cc33>carne </font>
<font color=#ff0000>non </font>
<font color=#ff0066>serve </font>
<font color=#ff0066>è </font>
```

```
<font color=#666666>un' </font>
<font color=#00cc33>acquisizione </font>
<font color=#ff6600>tardiva </font>
<font color=#9900ff>nell'evoluzione </font>
<font color=#9900ff>dei </font>
<font color=#00cc33>primati </font>
```

7.1.6 La creazione dell'EDT

Come è possibile constatare dallo schema, lo stesso oggetto *KMLObj* viene utilizzato per creare un file con estensione EDT (**ED**i**T**able), che è stato presentato nella Sezione 5.1, contenente il testo della trascrizione arricchito con markup interpretabile dall'HTML. Viene dunque chiamato il metodo *createEDT* della classe *Utils* che, a partire da un oggetto di tipo *KMLObj*, costruisce un file con estensione EDT in cui parola è incapsulata all'interno di un elemento *span* che riporta, sotto forma di attributi, tutti i valori contenuti nell'oggetto *KMLObj*. Il valore della variabile *pos* dell'oggetto *KMLObj* diventa il valore della classe dell'elemento *span*. In questo modo si potrà facilmente specificare una regola nei fogli di stile per assegnare il colore ad ogni part-of-speech. Tutti gli altri attributi dell'elemento prendono il nome delle variabili che compongono l'oggetto *KMLObj*; dunque gli elementi si presenteranno come:

```
<span class="D" posdescita="articolo" posdesceng="article" offset="0"
start="462" end="461" punct="" morphofeatita="articolo indeterminativo "
morphofeateng="indefinite article" token="Un'">Un' </span>

<span class="N" posdescita="sostantivo" posdesceng="noun" offset="4"
start="495" end="527" punct="" morphofeatita="nome di cosa singolare"
morphofeateng="name of thing singular" token="alimentazione">alimentazione
</span>

<span class="V" posdescita="verbo" posdesceng="verb" offset="18" start="528"
end="582" punct="" morphofeatita="participio" morphofeateng="participle mood"
token="corretta">corretta </span>

<span class="V" posdescita="verbo" posdesceng="verb" offset="27" start="583"
end="613" punct="" morphofeatita="modale, indicativo presente"
morphofeateng="modal, present tense" token="può">può </span>
```

```
<span class="V" posdescita="verbo" posdesceng="verb" offset="31" start="614"
end="640" punct="" morphofeatita="infinito" morphofeateng="infinitive"
token="fare">fare </span>

<span class="P" posdescita="preposizione" posdesceng="preposition" offset="36"
start="641" end="644" punct="" morphofeatita="//" morphofeateng="//"
token="a">a </span>

<span class="N" posdescita="sostantivo" posdesceng="noun" offset="38"
start="645" end="677" punct="" morphofeatita="" morphofeateng=""
token="meno">meno </span>

<span class="P" posdescita="preposizione" posdesceng="preposition" offset="43"
start="678" end="699" punct="" morphofeatita="//" morphofeateng="//"
token="della">della </span>

<span class="N" posdescita="sostantivo" posdesceng="noun" offset="49"
start="700" end="753" punct="" morphofeatita="nome di cosa singolare"
morphofeateng="name of thing singular" token="carne">carne </span>
```

Questo formato sarà fondamentale per:

- la visualizzazione dei sottotitoli e della trascrizione *colorati* e per la visualizzazione delle informazioni morfologiche aggiuntive quando, nella sezione *browse*, si clicca su una parola della trascrizione integrale (v. 5.6);
- la revisione della trascrizione e/o dell'annotazione: il fatto che l'EDT mantenga tutti i valori del *KMLObj* consente di ricostruire tale oggetto Java senza alcuna perdita di informazione;
- la creazione e lo svolgimento di esercizi PoST: si è visto (v. 5.8.1) che al *tutor* che crea un esercizio di questo tipo basterà selezionare la porzione di trascrizione; la soluzione dell'esercizio sta proprio nelle informazioni che fornisce l'EDT.

7.1.7 Il processo di aggiornamento: revisione della trascrizione

Nelle sezioni precedenti è stato preso in esame il workflow completo del programma. Se però al programma viene specificato un comando differente da *transcribe* il ciclo di esecuzione, in parte, cambia. Quando il *tutor* apporta delle modifiche alla trascrizione (v. 5.7), ci sarà bisogno di riannotare il trascritto in quanto è possibile che la modifica di una parola comporti un cambiamento della sua part-of-speech. Dunque in questo caso TASEe, che verrà invocato dallo script PHP specificando il parametro *annotate*, deve saltare la fase di trascrizione e passare a quella di annotazione. Per fare questo senza dover

riscrivere codice, è necessario porsi nelle stesse condizioni in cui si trova il programma alla fine della trascrizione. Questo significa che, a partire dall'attuale EDT deve essere creato un oggetto di tipo *XMLObj*. La classe *XMLObj* prevede infatti un metodo *createXMLObjFromEDT* che, come suggerisce il nome, si occupa di fare quanto richiesto. Tale metodo parsa il file EDT tramite un parser XML di Java (DocumentBuilder)[29], recupera il valore di ogni elemento *span* e tutti gli attributi necessari per costruire l'oggetto desiderato: *start*, *end*, *offset*, *punc*. Una volta parsato l'intero EDT saremo in possesso di un oggetto *XMLObj*, lo stesso creato dal metodo *createXML* (v. 7.1.2) a partire dalla trascrizione dell'ASR. A questo punto è possibile procedere all'annotazione come spiegato nella Sezione 7.1.3.

7.1.8 Il processo di aggiornamento: revisione dell'annotazione

Nel caso in cui il *tutor* in fase di revisione abbia apportato modifiche all'annotazione (v. 5.7), TASEe non dovrà, ovviamente, riannotare il testo e dovrà dunque occuparsi soltanto della creazione dei sottotitoli (il file EDT è già aggiornato). Dunque il programma, che questa volta verrà invocato dallo script PHP specificando il parametro *subtitle*, dovrà porsi nelle stesse condizioni in cui si troverebbe in un workflow completo al termine del processo di merge (v. 7.1.4); ciò significa creare, a partire dall'EDT, un oggetto *KMLObj*. Così come la classe *XMLObj*, anche la classe *KMLObj* mette a disposizione un metodo *createKMLObjFromEDT* che ricostruisce un oggetto di tale tipo. Tale metodo esegue esattamente il procedimento inverso di *createEDT* (v. 7.1.6): piuttosto che partire da un oggetto *KMLObj* per creare un file EDT, parte da un file EDT per ricreare un oggetto *KMLObj*. L'EDT viene parsato e, per ogni nodo *span* incontrato se ne recuperano tutti gli attributi (*offset*, *token*, *class pos*, *posDescIta*, *posDescEng*, *morphofeatIta*, *morphofeatEng*, *start*, *end*, *punct*) tramite cui si crea un oggetto *KMLElem* che viene aggiungo alla lista di *KMLElem* dell'oggetto *KMLObj*. Una volta scorso tutto l'EDT, il *main* potrà procedere nell'invocazione del solo metodo *createSRT* (v. 7.1.5) per produrre i sottotitoli con la nuova annotazione.

[29] http://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilder.HTML

8 I limiti dell'ASR: quali audiovisivi per l'apprendimento?

In questo Capitolo verranno evidenziati i limiti degli attuali motori di trascrizione e in particolare verranno esaminate le performance di SpeechScribe per capire quali siano le tipologie di audiovisivi da cui è possibile aspettarsi buoni risultati e che minimizzino il processo di revisione agli utenti *tutor*.

8.1 Uomo VS Computer

Il compito di un'ASR è senz'altro decisamente impegnativo poiché la complessità del linguaggio umano è, sotto molti aspetti, elevata. Vediamo dunque quali sono le principali sfide che un ASR deve fronteggiare[17].

- L'uomo comunica attraverso le parole, un insieme di produzioni di foni che si susseguono intervallati da piccolissime pause. Il linguaggio verbale è però soltanto una delle strategie che l'uomo mette in atto quando comunica; la prosodia, il tono della voce, la velocità con cui si esprime, le espressioni facciali, i gesti e la postura sono tutti elementi non verbali che aiutano l'interlocutore a interpretare correttamente il messaggio comunicatogli. Di tutto ciò il computer può captare soltanto l'aspetto legato all'emissione di suoni.
- Quando sentiamo due sequenze di suoni identiche, ad esempio *l'ago* e *lago*, la nostra conoscenza pregressa della lingua e del mondo ci permette di captare il corretto significato, considerando il contesto entro cui tale suono è prodotto. Nella frase *Sono andato a pescare al lago* riusciamo perfettamente a capire, sia grazie al contesto sintattico che a quello semantico, che si tratta del *lago* e non de *l'ago*. Ma se i due suoni vengono pronunciati isolati, senza che vengano inseriti all'interno di un discorso di senso compiuto, ci appaiono del tutto uguali. Questo è quello che *sente* il computer; dunque, come noi, il motore di ASR deve contestualizzare quel suono per poterne effettuare la corretta trascrizione. Anche in questo caso si procede spesso in modo statistico sulla base del modello del linguaggio.
- A causa dell'anatomia del nostro corpo (la forma e la lunghezza delle corde vocali, la forma delle cavità orali, la dimensione della lingua e così via) il timbro della voce di ognuno di noi è ben definito e differisce da quello di ogni altro individuo. Ulteriore elemento caratterizzante della produzione orale di un individuo è il suo accento regionale (i dialetti, in quanto idiomi a sè, non sarebbero trascrivibili da un ASR addestrato per l'italiano). Si pensi poi a tutte

quelle peculiarità che contraddistinguono alcuni parlanti come la r moscia o la lisca. In questo caso i sistemi addestrati al riconoscimento di un singolo speaker sono sicuramente facilitati; in un sistema *speaker independent* come SpeechScribe (v. 3.2.1) in cui il motore deve essere in grado di trascrivere l'audio indipendentemente dallo speaker che lo produce, le difficoltà aumentano notevolmente. I modelli acustici (v. 3.5.1) dei sistemi di ASR infatti vengono addestrati con un corpus di dati audio in cui i parlanti hanno una dizione conforme a quella che è la lingua per cui tale modello acustico viene creato. Il motore di SpeechScribe, essendo stato addestrato con le registrazioni dei telegiornali RAI in cui gli speaker sono perlopiù giornalisti aventi una dizione corretta, sarà facilitato nel trascrivere parlanti aventi una buona dizione e troverà maggiori difficoltà in presenza di parlanti con accento regionale o straniero (v. 8.2.2).

- Spesso le condizioni e il contesto entro il quale un parlante produce un discorso, presentano dei rumori di sottofondo, della musica, un brusio di voci, due voci che si sovrappongono e così via. L'orecchio umano riesce a filtrare questo *rumore* e captare soltanto il messaggio che chi parla sta producendo; questa operazione diventa non banale per l'ASR che, tramite tecniche di signal processing, dovrà filtrare l'audio trascrivendo soltanto ciò che non è rumore (v. 3.2.1.2). È presumibile dunque che l'ASR in presenza di un audio *pulito*, produca risultati migliori (v. 8.2.3).
- Oltre a filtrare il messaggio da eventuali rumori di sottofondo, l'orecchio umano riesce anche piuttosto bene a percepire il messaggio anche se il suono non è di ottima qualità. La voce che giunge al nostro orecchio da un interlocutore al telefono ha una qualità inferiore rispetto alla voce di un interlocutore con cui parliamo dal vivo. Nonostante questo, a parte conversazioni telefoniche particolarmente disturbate, riusciamo comunque a percepire e interpretare il messaggio. Per un ASR non è così facile e una buona qualità dell'audio che deve trascrivere è essenziale ai fini di un buon risultato. SpeechScribe ad esempio richiede una frequenza di campionamento di almeno 22Khz e la frequenza di campionamento della linea telefonica è di soli 8Khz. È presumibile quindi che il motore, in tali trascrizioni, non sia troppo performante (v. 8.2.1). Per poter trascrivere correttamente materiale audio telefonico, o comunque di bassa qualità audio, si dovrebbe ricreare il modello acustico del motore a partire da un corpus di audio aventi tali caratteristiche.

- Quando ascoltiamo un discorso in una lingua a noi sconosciuta, non riusciamo ad identificare le singole parole e, alle nostre orecchie, tutto appare come un'unica produzione di suoni; questo è ciò che *sente* il computer. Quindi, dopo aver riconosciuto la sequenza di foni, il motore deve raggrupparli in modo ragionevole individuando i limiti di ogni parola. Questa operazione viene eseguita spesso ricorrendo ad un approccio statistico che integra un modello linguistico (v. 3.5.1). Dato però che la lingua è immensa e in continua evoluzione, è probabile che il sistema si trovi a dover riconoscere parole non comprese nel suo vocabolario perché rare, nuove, o arcaiche (v. 8.2.4).

Dunque la qualità della trascrizione e, conseguentemente, dell'annotazione, è strettamente dipendente dal contenuto dell'audio e dalla sua qualità. Nella prossima sezione verranno confrontate alcune tipologie di audiovisivi per individuare i requisiti che permettano di ottenere una trascrizione (e annotazione) attendibile e dunque utilizzabile, senza un eccessivo intervento di revisione, a scopo didattico.

8.2 Analisi delle performance

Come si è detto nella sezione precedente, gli aspetti che possono mettere in difficoltà un motore di trascrizione sono molteplici. Per verificare quanto detto, vediamo di esaminarne alcuni mettendo a confronto un audiovisivo che presenta un certo aspetto negativo con un audiovisivo che invece ne è privo.

Le seguenti valutazioni verranno effettuate mediante il calcolo del WER, una misura che indica la percentuale media di errore per quella trascrizione. Per calcolare tale misura è necessaria la trascrizione elaborata dal sistema e una trascrizione di riferimento fedele all'audio (*gold*) elaborata, in parte o completamente, dall'uomo. L'algoritmo per il calcolo del WER allinea e confronta la sequenza di parole delle due trascrizioni calcolando quante parole il sistema ha sostituito, quante parole non ha trascritto e quante ne ha trascritte laddove non erano presenti. Questi tre valori vengono detti rispettivamente *substitutions* (S), *deletion* (D) e *insertions* (I) e vengono utilizzati nella seguente formula:

$$WER = \frac{S + D + I}{N}$$

- dove *N* è il numero totale di parole presenti nel *gold.*

Lo script utilizzato per il calcolo del WER è stato anch'esso elaborato durante il progetto SAVAS (v. 3.2) e consiste in un eseguibile invocabile da un prompt tramite la sintassi:

```
>evaluate.exe –ref [path_alla_trascrizione_di_riferimento] –hyp
[path_alla_trascrizione_del_sistema]
```

L'output di tale chiamata genera una stringa che riporta il WER e specifica i valori con cui è stato calcolato (S, D, I, N).

Poiché il WER non tiene in considerazione punteggiatura e *capitalization*, sia le trascrizioni di riferimento sia quelle elaborate dal sistema devono esserne prive. Poiché SpeechScribe genera automaticamente entrambe, ho realizzato uno script che, prima di invocare l'eseguibile, normalizza i testi eliminando punteggiatura e rendendo tutti i caratteri minuscoli

8.2.1 Qualità dell'audio

Per verificare quanto la qualità della registrazione influisca sui risultati della trascrizione, è stato preso in esame un audio[30] dal sito del consorzio ICoN che contiene un'intervista in cui colui che risponde è in collegamento telefonico. La qualità dell'audio in corrispondenza dell'intervistatrice è molto buona e nelle porzioni di audio in cui parla l'intervistato è ovviamente peggiore (ma comunque comprensibile dall'orecchio umano). Dopo aver realizzato il gold di questo audio ne è stato calcolato il WER; dopodiché l'audio è stato scisso in due parti: la prima contenente la sola intervistatrice e la seconda contenente il solo intervistato (allo stesso modo è stato suddiviso il testo del gold). Nei due audio le condizioni acustiche, la dizione dei due parlanti e il linguaggio utilizzato, sono perlopiù equivalenti e dunque non vi sono altre variabili che possano incidere sui risultati. Il WER dei tre audio è mostrato nella tabella seguente.

[30] Lettura *Bosco di San Francesco*, http://corsilingua.italicon.it/services/media/11/Audio_Salviamo_Ambiente/C1_MSA_ULettura_S6_A2-A3-A4_boscoSfrancesco.mp3

	Totale parole nel gold	Inserimenti	Cancellazioni	Sostituzioni	**WER (%)**
Audio integrale	255	10	9	64	**32,55**
Intervistatrice (buona qualità)	87	1	0	3	**4,59**
Intervistato (bassa qualità)	168	8	10	55	**43,45**

Come è possibile constatare dalla tabella, il divario tra i WER dei due parlanti (4,59% contro 43,45%) è decisamente molto consistente, prova del fatto che la bassa qualità incide notevolmente sui risultati della trascrizione. Dunque, al fine di ottenere una trascrizione attendibile è consigliabile caricare in SubItalian audiovisivi di buona qualità acustica[31], possibilmente non provenienti da telefonico.

8.2.2 Accento

Per verificare quanto un accento che si discosta dall'italiano standard influisca sulla trascrizione, è stato preso in esame un video[32] contenente un'intervista all'attore e regista Luca Zingaretti la cui parlata è caratterizzata da un accento romano. L'intervista è realizzata da Rosaria Renna, conduttrice radiofonica con studi di dizione alle spalle[33]. Oltre ad una miglior dizione, da parte dell'intervistatrice si avvertono parole meglio scandite.

Analogamente al caso precedente è stato dapprima realizzato il *gold* dell'audio integrale[34]; l'audio è stato poi scisso in modo da ottenere due file audio contenenti rispettivamente la sola intervistatrice e il solo intervistato. I WER sono mostrati nella tabella seguente.

[31] Precisamente si consiglia una frequenza di campionamento di almeno 22KHz.
[32] Intervista RDS: Penelope Cruz protagonista di "Venuto al Mondo" https://www.youtube.com/watch?v=KItQFteVJc0we
[33] http://www.art-roma.it/scheda-docente.php?id=2
[34] In realtà la parte dell'intervistato, assai più lunga di quella dell'intervistatrice, è stata scorciata per cercare di avere una quantità di audio il più possibile equivalente.

	Totale parole nel gold	Inserimenti	Cancellazioni	Sostituzioni	**WER (%)**
Audio integrale	408	8	41	102	**37,01**
Rosaria Renna (dizione corretta)	168	0	0	20	**11,90**
Luca Zingaretti (accento romano)	240	2	35	77	**47,50**

La tabella mostra che le porzioni di audio relative a Rosaria Renna ottengono una trascrizione con un WER molto più basso (11,90%) delle porzioni di audio relative a Luca Zingaretti (47,50%). Quindi, nonostante a orecchio umano i discorsi di Zingaretti siano assolutamente comprensibili, il motore di trascrizione è poco performante. Ai fini dell'insegnamento della lingua italiana sarebbe pertanto opportuno caricare in SubItalian video che contengano parlanti con una buona dizione.

8.2.3 Rumori e/o musica di sottofondo

Per verificare quanto dei rumori o della musica di sottofondo che all'orecchio umano non ostacolerebbero la corretta interpretazione del parlato, influiscano invece sui risultati dell'ASR, si è preso in esame un audio[35] prelevato dal sito del consorzio ICoN riportante una notizia di cronaca. I parlanti hanno una buona dizione e non vi sono rumori o musica di sottofondo. Sono stati poi creati due ulteriori audio, sovrapponendo all'originale, tramite il software *Audacity*[36],una traccia audio contenente il brano *Divenire* di Ludovico Einaudi. Nel primo audio creato il volume della nuova traccia è molto basso e non disturba l'ascolto; nel secondo audio il volume è più alto e, nonostante l'orecchio umano riesca comunque a comprendere ciò che viene detto nella traccia audio principale, il

[35] Lettura della sezione *Sacro Profano* http://corsilingua.italicon.it/services/media/11/Audio_Sacro_Profano/C1_SP_Lettura_S6_A2-A3-A4.mp3
[36] Tramite il software Audacity (http://audacityteam.org/download/?lang=it)

sottofondo è piuttosto invadente. Dopo aver creato il gold dell'audio è stato calcolato il WER delle tre versioni.

	Totale parole nel gold	Inserimenti	Cancellazioni	Sostituzioni	**WER (%)**
Audio originale	239	3	13	36	**21,76**
Audio con musica di sottofondo (volume più basso)	239	3	23	43	**28,45**
Audio con musica di sottofondo (volume più alto)	239	6	40	48	**39,33**

Come ci si poteva aspettare, i risultati della trascrizione con un sottofondo musicale peggiorano rispetto all'audio pulito e si aggravano con l'aumentare del volume del sottofondo, passando da un WER del 21,76% ad uno del 28,45% fino al 39,33%. Minore è il volume dei rumori/musica di sottofondo, maggiore è la probabilità che l'*audio pre-processing block* di SpeechScribe (v. 3.2.1.2) riesca a filtrare l'audio, identificando e trascrivendo soltanto il parlato. Di conseguenza, è consigliabile caricare in SubItalian del materiale video che contenga un rumore e/o musica di sottofondo ai minimi livelli.

8.2.4 Termini estranei al dizionario

In un audio che presenti un dominio molto distante da quello con cui è stato addestrato il sistema (quello delle news) è molto probabile che vi siano parole sconosciute all'ASR (rare, nuove, arcaiche). Inoltre in alcuni domini, come quello poetico, la struttura sintattica è ben diversa da quella cui il modello del linguaggio dell'ASR fa riferimento. Per verificare, almeno in parte, questo aspetto, si sono presi in esame i primi 30 versi del primo canto dell'Inferno della Divina Commedia e la loro parafrasi[37]. La parafrasi in questione mantiene una

[37] Sito *La Divina Commedia* http://divinacommedia.weebly.com/inferno-canto-i.html

struttura piuttosto simile al poema originale ma il linguaggio utilizzato è ovviamente quello contemporaneo. I versi originali e la loro parafrasi sono stati da me letti e registrati (cercando di mantenere una dizione più corretta possibile!) creando due differenti file audio. I gold sono stati creati semplicemente recuperando il testo dettato per la registrazione.

	Totale parole nel gold	Inserimenti	Cancellazioni	Sostituzioni	**WER (%)**
30 Versi, I Canto Inferno **Testo**	211	10	11	98	**56,40**
30 Versi, I Canto Inferno **Parafrasi**	234	10	5	27	**17,95**

Come è possibile constatare dalla tabella, il WER ottenuto per la trascrizione della Divina Commedia originale è molto più alto di quello ottenuto per la sua versione parafrasata.

Vi sono molti termini che non rientrano nel dizionario di SpeechScribe e che dunque vengono sostituiti con parole a lui conosciute:

esta→ resta
queta → questa
compunto → confronto

Le corrispondenti parole della parafrasi (*quella, [si] placò* e *rattristato*) vengono invece trascritte correttamente.

Anche dal punto di vista sintattico possiamo constatare che alcune costruzioni non in uso, a differenza della loro parafrasi, mettono in difficoltà l'ASR, che non riesce ad elaborare una trascrizione corretta.

	Gold	Trascrizione ASR
Testo originale	*tant'è amara che poco è più morte*	**tante amare che può per più molte**
Parafrasi	*così spaventosa che la morte lo è poco di più*	**così spaventosa che la morte lo è poco di più**
Testo originale	*cose ch'i v'ho scorte*	**cose chi vasco arte**
Parafrasi	*che vi trovai dentro*	**che vi trovai dentro**

Dunque, al fine di ottenere una trascrizione corretta, è preferibile caricare in SubItalian degli audiovisivi il cui dominio non si discosti troppo da quello con cui è stato addestrato il sistema.

9 Casi d'uso

In questo capitolo verranno presentati tre possibili casi d'uso di SubItalian: la sua integrazione in un LMOOC, il suo utilizzo da parte di un singolo utente straniero e il suo impiego come strumento ausiliare alla didattica nelle scuole elementari.

9.1 LMOOC

Il termine MOOC (Massive Open Online Courses) è stato utilizzato per la prima volta da George Siemens, un ricercatore canadese che nel 2008 architettò una strategia di *diffusione dell'apprendimento* su larga scala[33]. Infatti, come il nome suggerisce, un MOOC è un corso di formazione online che ha come obiettivo quello di coinvolgere un vastissimo numero di utenti rimuovendo le barriere d'accesso all'informazione. Una particolare tipologia di MOOC sono i Language MOOC (LMOOC) che prevedono appunto l'insegnamento a distanza di una lingua straniera. Dopo il 2008 i MOOC hanno visto una diffusione piuttosto rapida sia presso le università che presso le aziende e il 2012, con più di un milione e mezzo di utenti iscritti a un MOOC, è stato definito "l'anno dei MOOC". Negli anni successivi però si è assistito ad una forte diminuzione della diffusione di tali corsi a causa di un alto tasso di abbandono da parte dei partecipanti iscritti. Si è infatti registrato che la percentuale media di utenti che portano a termine un corso è molto bassa e si aggira tra il 7 e il 10%[61]. I partecipanti sono spesso poco motivati, possono riscontrare difficoltà in termini di auto gestione, oppure possono trovare il corso poco attrattivo. Per cercare di ridurre al minimo il tasso di abbandono, il docente può cercare di intervenire su questi aspetti (in modo particolare sull'ultimo) cercando di realizzare un corso che risulti ben organizzato, interattivo, accattivante, facile da usare e che preveda interazione, sia tra pari che con il docente stesso. La creazione di un MOOC richiede quindi al docente competenze aggiuntive rispetto a quelle standard, che riguardano la gestione di risorse tecnologiche e *web-based*. SubItalian può dunque rivelarsi utile in tale contesto per:

- Alleggerire il carico di lavoro necessario alla realizzazione del corso: il docente potrà caricare in modo agevole i contenuti audiovisivi (v. 4.4) e potrà creare esercitazioni in modo semplice e intuitivo (v. 4.7). Potrà inoltre indagare circa la partecipazione degli iscritti attraverso il forum (v. 4.9) e attraverso la sezione dedicata alle statistiche (v. 4.8).
- Introdurre una fonte di attrazione: come si è visto nel Capitolo 2, l'utilizzo di materiale didattico interattivo, e in particolare l'introduzione di contenuti

audiovisivi, costituisce un approccio stimolante per l'apprendente che pertanto potrebbe essere maggiormente invogliato a seguire il corso.

9.2 Singolo utente straniero

Spesso uno studente alle prime armi con una LS può sentirsi in imbarazzo nello *sfoggiare* in pubblico una lingua che conosce a malapena. Il verificarsi di situazioni in cui l'apprendente non riesce a comprendere ciò che l'interlocutore gli sta comunicando o in cui non riesce a trovare le giuste parole per rispondere a domande che gli vengono poste può risultare estremamente frustrante e sicuramente demotivante. Questa è una delle regioni per cui molti studenti dichiarano che le attività di apprendimento che preferiscono sono quelle individuali (v. 1.1). SubItalian potrebbe quindi essere utilizzato per tutti gli apprendenti di italiano che desiderino adottare un approccio comunicativo (v. 1.1), facendo pratica, sia orale che scritta, senza dover necessariamente ricorrere a una comunicazione faccia a faccia. Si è detto però che per poter intraprendere un percorso di apprendimento di una lingua straniera mediante fruizione di audiovisivi è necessaria una delle seguenti condizioni:

- l'apprendente ha una minima competenza iniziale della lingua target;
- la lingua madre dell'apprendente è affine a quella target.

Detto questo, possiamo delineare due possibili profili di un singolo apprendente che utilizzi SubItalian:

- studente di lingua cinese iscritto ad un corso universitario di lingua italiana. Uno studente con una lingua così distante dalla nostra, per poter ottenere risultati dall'apprendimento mediante audiovisivi, necessita di acquisire una competenza di base. Dunque se tale studente sta apprendendo (o ha appreso) le basi della lingua italiana in un contesto strutturato, potrebbe fare uso di SubItalian per far pratica, velocizzare, potenziare e rendere più stimolante il proprio processo di apprendimento.
- utente spagnolo che non stia apprendendo la lingua italiana in un contesto strutturato. La somiglianza lessicale tra lo spagnolo e l'italiano è dell'82% [62] e, stando a quelli che sono i fatti verificatisi a Malta e in Albania (v. 1.2.1 e 1.2.2), è del tutto probabile che un apprendente spagnolo, che sia determinato nell'apprendere la lingua italiana, riesca ad acquisire delle ottime competenze attraverso il solo utilizzo di SubItalian. Un tutor iscritto a SubItalian potrebbe

verificare tale ipotesi attraverso la sezione dedicata alle statistiche sulle performance (v. 4.7).

9.3 Ausilio alla didattica nelle scuole elementari

Sebbene SubItalian sia nato pensando prevalentemente ad utenti stranieri, niente vieta di utilizzarlo come strumento ausiliario all'insegnamento dell'italiano per utenti italiani, ovviamente, nelle scuole elementari. Nonostante l'introduzione di strumenti informatici già nelle scuole elementari sia un aspetto piuttosto controverso, sono molte le opinioni a favore di tale approccio[43]. Nelle scuole elementari dotate di laboratori informatici, sarebbe dunque possibile pensare di utilizzare SubItalian per agevolare l'insegnamento dell'italiano da un punto di vista sintattico, ortografico e grammaticale. La sottotitolazione aiuterebbe gli studenti a sviluppare la capacità di associare forma orale e forma scritta, mentre l'annotazione permetterebbe di studiare la grammatica in modo del tutto alternativo. I maestri dovranno iscriversi a SubItalian come utenti *tutor* e gli studenti come utenti *learner*. I primi potranno creare esercitazioni (v. 4.7) più o meno complesse in funzione dell'attuale livello della classe e potranno monitorare lo sviluppo delle competenze dei secondi, seguendoli nello svolgimento di tali esercizi. Inoltre potranno selezionare il materiale audiovisivo da caricare in SubItalian che, per catturare l'attenzione degli studenti, potrà essere sulla falsariga del palinsesto di Disney Channel!

10 Conclusioni

In questo capitolo verranno dapprima affrontati i maggiori ostacoli ad un approccio didattico via web. Verranno poi discussi, entrando nella specificità di SubItalian, gli ostacoli relativi alla trascrizione automatica. Infine, verranno descritte le possibili migliorie e i possibili sviluppi futuri di SubItalian.

10.1 Digital Divide

Nella Sezione 2.2 si è detto che l'invenzione del web ha rappresentato una svolta decisiva per la formazione a distanza e per gli strumenti CALL in generale.

Sebbene la diffusione del web sia stata rapidissima (dalla sua nascita in soli sette anni ha raggiunto il 30% della popolazione americana) è stata al tempo stesso estremamente disomogenea, e oggi si assiste a quello che viene chiamato *digital divide* o divario digitale. Il termine era nato negli anni novanta per indicare la disparità presenti in America tra i possessori, e non, del personal computer. L'espressione è stata poi adattata per designare il divario presente tra i soggetti che hanno pieno accesso alla rete e alle tecnologie afferenti e coloro che invece non ne hanno le possibilità/capacità.

Il digital divide più evidente è sicuramente quello che esiste tra i paesi sviluppati e quelli in via di sviluppo. La penetrazione di Internet in Africa, ad esempio, è a livelli minimi mondiali (si stima che nel 2008 soltanto il 4% della popolazione africana avesse accesso a internet). Le cause di tale isolamento sono principalmente tre:

- basso livello d'istruzione: nei paesi più arretrati si assiste spesso al fenomeno del cosiddetto *analfabetismo informatico* per cui molti soggetti non sono capaci di interagire con la tecnologia ed il computer;
- bassa qualità delle infrastrutture: in queste zone la larghezza di banda è estremamente limitata, il numero di host è molto ridotto (1.800.000, contro i 120 milioni dell'Europa) e la maggior parte delle comunità rurali non è raggiunta persino dalla rete telefonica fissa;
- difficoltà economiche: la maggior parte delle famiglie africane non può sostenere la spesa di un computer e, tanto meno, quella ad un abbonamento Internet.

Per questi paesi l'introduzione di internet sarebbe determinante per migliorare la qualità della vita attraverso l'istruzione a distanza, compensando magari la carenza di altre fonti di informazione come biblioteche[1][58].

Le cause che impediscono un regolare accesso al web non riguardano però soltanto i paesi sottosviluppati. La Cina dal 2003, con l'introduzione del sistema di sorveglianza internet Great Firewall, blocca l'accesso a una serie di siti considerati tabù dal Governo e monitora costantemente i dati in entrata e in uscita dell'utenza. Più di 600 milioni di utenti internet si nascondono dietro questa barriera [47].

Questo importante divario non lascia certo indifferenti e le iniziative prese da vari paesi per farvi fronte sono molte. Il piano *eEurope 2005* (v. 2.1) è incentrato sull'obiettivo di una connettività Internet trasversale che garantisca a *tutti* l'accesso alle risorse. Per far fronte a tale esigenza e garantire dunque a chiunque la possibilità di accedere a strumenti di e-learning, tale piano di azione proponeva una serie di obiettivi da perseguire, tra cui l'accesso a banda larga (entro il 2005) da parte di tutti gli istituti di insegnamento e università e la messa a disposizione di un accesso alla rete per studenti e ricercatori [49].

10.2 Limiti dell'e-learning

Nonostante, come si è visto nella Sezione 2.1, l'e-learning apporti numerosi vantaggi e apra nuove frontiere alla didattica, vi sono alcuni limiti di cui si deve tener conto. Il primo grande ostacolo è stato appena affrontato e riguarda quindi l'attuale impossibilità di un accesso trasversale ad Internet. Gli altri potenziali limiti sono di varia natura e gravità.

Uno dei problemi maggiori è correlato al discente che spesso non è sufficientemente motivato, o è vittima di troppe distrazioni domestiche, per riuscire a completare un corso *individualmente* senza l'ausilio delle stimolazioni *faccia a faccia* di un insegnante. Sebbene questo aspetto dipenda strettamente dall'atteggiamento del singolo discente, la realizzazione di un'adeguata applicazione di e-learning è estremamente importante per minimizzare tale rischio. La mancata interazione *faccia a faccia*, e la mediazione di un computer, possono essere causa di un altro rischio: l'isolamento sociale[38].

Un ulteriore ovvio svantaggio dell'e-learning è l'impossibilità di trattare in modo completo qualsiasi ambito didattico. Per alcuni campi, specialmente quelli che riguardano l'informatica e il web (web-design, linguaggi di programmazione ecc.), la corretta formazione di un discente è senz'altro un obiettivo realizzabile. Per altri ambiti una buona formazione a distanza è più complessa da mettere in piedi, ma comunque attuabile; un esempio è proprio quello dell'apprendimento di una lingua straniera per cui si deve prestare particolare attenzione all'oralità, aspetto ovviamente trattato in un insegnamento in presenza, ma che rischia di essere trascurato o di non essere affrontato correttamente,

in un insegnamento a distanza. Vi sono infine degli ambiti per cui non è possibile adottare una formazione (solo) a distanza; è il caso del campo della medicina che, ovviamente, richiede anche un addestramento pratico. Un domani forse questo limite potrà essere superato da applicazioni di realtà virtuale.

11 Bibliografia e sitografia

[1] Ajuwon, Grace A. "Internet Accessibility and Use of Online Health Information Resources by Doctors in Training Healthcare Institutions in Nigeria." Health Information & Libraries Journal 09/2015; 32(3):241-6.

[2] Alvarez, Aitor, et al. "Automating live and batch subtitling of multimedia contents for several European languages." Multimedia Tools and Applications: 1-31.

[3] Baltova, Iva. "Multisensory language teaching in a multidimensional curriculum: The use of authentic bimodal video in core French." Canadian Modern Language Review 56.1 (1999): 31-48.

Ademi, Esmeralda; Bulija, Mirjiana; Longo, Maurizio. 2010. “Una quantificazione della penetrazione della lingua italiana in Albania tramite la televisione”. Èducation et Sociétés Plurilingues. Numero 28, 2010

[4] Bosco, Cristina, et al. Harmonization and Development of Resources and Tools for Italian Natural Language Processing Within the PARLI Project. Springer International Publishing, 2015.

[5] Bosma, Wauter, et al. "KAF: a generic semantic annotation format." Proceedings of the GL2009 workshop on semantic annotation. 2009.

[6] Bravo, Maria Conceição. "Foreign language learning made simpler by reading subtitles." Studies in teacher education (2005): 105-114.

[7] Caimi, Annamaria. "Audiovisual translation and language learning: The promotion of intralingual subtitles." The journal of specialized translation 6 (2006): 85-98.

[8] Caruana, Sandro. "Italian in Malta: a socio-educational perspective." International Journal of Bilingual Education and Bilingualism 16.5 (2013): 602-614.

[9] Caruana, Sandro. "Trilingualism in Malta: Maltese, English and ‘Italiano Televisivo’." International Journal of Multilingualism 3.3 (2006): 159-172.

[10] Cavaliere, Salvatore. "Motivazione e apprendimento dell’italiano LS."

[11] Chang, J. Y. "Captioned movies and vocabulary acquisition: Learning English through movies." MA proposal. University of Southern California (2004).

[12] Danan, Martine. "Captioning and subtitling: Undervalued language learning strategies." Meta: Journal des traducteursMeta:/Translators' Journal 49.1 (2004): 67-77.

[13] Daniel, Jurafsky, and H. M. James. "Speech and Language processing: An introduction to natural language processing." Computational Linguistics and Speech Recognition. Prentice Hall, NJ, USA (2000).

[14] Davies, G. "EXPODISC: an interactive videodisc package for learners of Spanish." Audiovisual librarian 17.1 (1991): 31-37.

[15] De Mauro, Tullio. Storia linguistica dell'Italia unita. Laterza, 1970.

[16] Drakouli, Athanasia, and Georgia Milioni-Bertinelli. "Una proposta per l'insegnamento dell'italiano come LS in Grecia: la prospettiva dell'utilizzo della tv satellitare nella scuola statale greca."

[17] Forsberg, Markus. "Why is speech recognition difficult." Chalmers University of Technology (2003).

[18] Freeman, Diane E. Larsen. "The acquisition of grammatical morphemes by adult ESL students." TESOL quarterly (1975): 409-419.

[19] Gambier, Yves, and Henrik Gottlieb, eds. (Multi) media translation: concepts, practices, and research. Vol. 34. John Benjamins Publishing, 2001.

[20] Ganino, Giovanni. Immagini per la didattica: metodologie e tecnologie dell'audiovisivo digitale. Anicia, 2009.

[21] Gieve, Simon, and Rose Clark. "'The Chinese approach to learning': Cultural trait or situated response? The case of a self-directed learning programme." System 33.2 (2005): 261-276.

[22] Gold, Ben, Nelson Morgan, and Dan Ellis. Speech and audio signal processing: processing and perception of speech and music. John Wiley & Sons, 2011.

[23] Grixti, Joe. "Symbiotic transformations: youth, global media and indigenous culture in Malta." Media, Culture & Society 28.1 (2006): 105-122.

[24] Huang, X.H. & Van Naerssen, M. (1987). Learning strategies for oral communication. Applied Linguistics, 8/3.

[25] Hyland, Fiona. "Learning autonomously: Contextualising out-of-class English language learning." Language Awareness 13.3 (2004): 180-202.

[26] Ina, Lekkai. "Incidental Foreign-Language Acquisition by Children Watching Subtitled Television Programs." Turkish Online Journal of Educational Technology-TOJET 13.4 (2014): 81-87.

[27] Koolstra, Cees M., and Jonannes WJ Beentjes. "Children's vocabulary acquisition in a foreign language through watching subtitled television programs at home." Educational Technology Research and Development 47.1 (1999): 51-60.

[28] Krashen, Stephen. Principles and practice in second language acquisition. Pergamon: Oxford, 1982.

[29] La Noce, Filippo. E-learning. La nuova frontiera della formazione. Vol. 236. FrancoAngeli, 2002.

[30] Laçej, Eliana. "L'acquisizione dell'italiano da parte di apprendenti albanesi in contesti non guidati. L'errore linguistico." Analele Universităţii din Craiova. Seria Ştiinţe Filologice. Lingvistică 1-2 (2010): 306-314.

[31] Leogrande, Alessandro. Il naufragio: morte nel Mediterraneo. Feltrinelli Editore, 2011.

[32] Longo, Maurizio.Ademi, Esmeralda; Bulija, Mirjiana; 2010. Una quantificazione della penetrazione della lingua italiana in Albania tramite la televisione. Èducation et Sociétés Plurilingues. Numero 28, 2010

[33] McAuley, Alexander, et al. "The MOOC model for digital practice." (2010).

[34] Mezzadri, M. "Utilizzo delle nuove tecnologie nella didattica dell'italiano L2: implicazioni metodologiche." Maddii L.(a cura di), Insegnamento e apprendimento dell'italiano L2 in età adulta, Edilingua, Atene (2004): 153-161.

[35] Neves, Josélia. "10 fallacies about subtitling for the D/deaf and the hard of hearing." JosTrans 10 (2008).

[36] Perego, Elisa, 2005, La traduzione audiovisiva, Carocci, Roma.

[37] Pickard, Nigel. "Out-of-class language learning strategies." ELT journal 50.2 (1996): 150-159.

[38] Ranieri, Maria. E-learning: modelli e strategie didattiche. Vol. 3. Edizioni Erickson, 2005.

[39] Rapaport, William J. "In defense of contextual vocabulary acquisition." Modeling and using context. Springer Berlin Heidelberg, 2005. 396-409.

[40] Schneider, Edward W., and Junius L. Bennion. "Veni, vidi, vici via videodisc: a simulator for instructional conversations." System 11.1 (1983): 41-46.

[41] Simone, R., Una “trama” italiana, in “La crusca per voi”, 4, 1992

[42] Trentin, Guglielmo. Dalla formazione a distanza all'apprendimento in rete. Vol. 4. FrancoAngeli, 2001.

[43] Van Braak, Johan, Jo Tondeur, and Martin Valcke. "Explaining different types of computer use among primary school teachers." European Journal of Psychology of Education 19.4 (2004): 407-422.

[44] Vitale, Rosario, 2009. L’italiano a Malta: quale lingua? Supplemento alla rivista EL.LE - ISSN: 2280-6792 http://www.itals.it/l%E2%80%99italiano-malta-quale-lingua

[45] Willing, K., 1994. Learning Strategies in Adult Migrant Education. National Centre for English Language Teaching and Research, Sydney

[46] Wong, Lillian LC, and David Nunan. "The learning styles and strategies of effective language learners." System 39.2 (2011): 144-163.

[47] Yang, Qinghua, and Yu Liu. "What’s on the other side of the great firewall? Chinese Web users’ motivations for bypassing the Internet censorship." Computers in Human Behavior 37 (2014): 249-257.

[48] Zarei, Abbas Ali, and Zohreh Rashvand. "The effect of interlingual and intralingual, verbatim and nonverbatim subtitles on L2 vocabulary comprehension and production." Journal of language teaching and research 2.3 (2011): 618-625.

[49] Eur-lex, Programma eEurope 2005, http://eur-lex.europa.eu/legal-content/IT/TXT/HTML/?uri=URISERV:l24226&from=IT

[50] OpeNER Project, http://www.opener-project.eu/

[51] The Subdubexperience. Sottotitolazione intralinguistica, https://thesubdubexperience.wordpress.com/tag/sottotitolazione-intralinguistica/

[52] Treccani, voce Mediterraneo e lingua italiana,
http://www.treccani.it/enciclopedia/mediterraneo-e-linguaitaliana_%

[53] Unipa. Sorce. Tokenizzazione,
http://www1.unipa.it/sorce/didattica/sei1213/SEI1213_05_tokenizzazione.pdf

[54] Univeritual. Tecnhice didattiche. Cloze: tecnica metacognitiva,
http://www.univirtual.it/red/Tecniche-didattiche/CLOZE_tecnica_metacognitiva_Licia_LANDI.pdf

[55] Wikipedia, voce Cloze test, https://en.wikipedia.org/wiki/Cloze_test

[56] Wikipedia, voce E-learning, https://it.wikipedia.org/wiki/E-learning

[57] Wikipedia, voce Gestalt psychology
https://en.wikipedia.org/wiki/Gestalt_psychology

[58] Wikipedia, voce Internet in Africa,
https://it.wikipedia.org/wiki/Internet_in_Africa#Situazione_attuale

[59] Wikipedia, voce Irredentismo italiano a Malta
https://it.wikipedia.org/wiki/Irredentismo_italiano_a_Malta#L.27irredentismo_fascista

[60] Wikipedia, voce Lingua maltese, https://it.wikipedia.org/wiki/Lingua_maltese

[61] Wikipedia, voce Massive open online course,
https://en.wikipedia.org/wiki/Massive_open_online_course

[62] Wikipedia, voce Somiglianza lessicale,
https://it.wikipedia.org/wiki/Somiglianza_lessicale

[63] Wikipedia, voce SubRip
https://en.wikipedia.org/wiki/SubRip#SubRip_.28.srt.29_structure_examples

12 Appendice

12.1 Indice delle figure