Orizzonti giapponesi
ISBN 978-88-255-2118-4
DOI 10.4399/978882552118414
pp. 281-301 (dicembre 2018)

Riconoscimento fonetico della lingua giapponese attraverso gli strumenti digitali

Sviluppo di prototipi e analisi dei risultati

Alessandro Mantelli

La creazione di sistemi E-learning richiede non solo conoscenze di tipo informatico ma anche il possesso di competenze che consentano di organizzare il percorso che porta all'apprendimento. Tali competenze possono essere di tipo teorico (conoscenze di glottodidattica, psicologia dell'apprendimento) e di tipo progettuale quali le abilità nell'organizzazione dei contenuti e nella progettazione di interfacce (*User Experience Instructional Design*). Relativamente a questo ultimo punto, la tecnologia permette di realizzare, oltre a interfacce cinestetiche, rappresentate dalla creazione e l'organizzazione di elementi virtuali sullo schermo, interfacce che consentono l'acquisizione della voce e la successiva trasposizione in testo scritto. Queste interfacce vengono chiamate VUI (*Voice User Interface*). La scelta implementativa di una interfaccia VUI può avere diverse motivazioni: intrinseche necessità del discente (difficoltà di apprendimento quali la dislessia per esempio), esigenze di implementazione di esercizi di pronuncia o test di valutazione della produzione orale, creazione di sistemi di assistenti virtuali quali Siri per iOS e Mac OSX o Cortana per Windows 10, creazione di ambienti virtuali condivisi tipo *Second Life*[1] o i vari *Mmorpg*.[2]

[1] Si veda: Kaplan e Haenlein (2009).

[2] Massively Multiplayer Online Role-Playing Game. Per approfondimenti si veda: Feng, Brandt e Debanjan (2007)..

Nel momento in cui si prendono in considerazione le necessità su esposte e quindi si voglia implementare un'interfaccia che richiede l'acquisizione vocale, bisognerà indagarne i costi e i risultati effettivi. È dunque questa la domanda di ricerca a cui il presente studio intende rispondere.

Nel presente studio analizzeremo due tipologie di sistemi di acquisizione fonetica e successiva produzione di testo (*speech-to-text*) per la lingua giapponese e i rispettivi costi di implementazione all'interno di un'applicazione web. Riporteremo inoltre, i risultati di alcuni test di riconoscimento svolti su prototipi appositamente realizzati.

Ad oggi non si sono rilevate precedenti ricerche sull'analisi comparativa di sistemi di riconoscimento fonetico per la lingua giapponese. Il presente studio si focalizzerà quindi sull'analisi dei risultati di due sistemi di riconoscimento (Julius e Web Speech API), differenti sia per tecnologia utilizzata sia per l'approccio all'analisi e utilizzo delle informazioni. Questi due sistemi inoltre a differenza di software quali HTK[3] e CMU Sphinx,[4] posseggono di base un buon supporto per la lingua giapponese.

Applicazioni Web e riconoscimento fonetico

Sebbene funzionalità di riconoscimento fonetico siano ormai presenti all'interno dei più noti sistemi operativi sia per PC (Windows e Mac OSX) quanto per *tablet* e *smartphone* (Android e iOS), non è possibile utilizzare questi strumenti per la pianificazione di un applicativo Web. Innanzitutto, infatti, la capacità di riconoscimento fonetica di questi sistemi varia a seconda del sistema operativo utilizzato e non è quindi univoca, e in alcuni casi questa funzione può non essere presente, ma soprattutto, l'applicativo *E-Learning* che si va a creare deve necessariamente

[3] Si veda la pagina del progetto: http://htk.eng.cam.ac.uk/

[4] Si veda la pagina del progetto: https://cmusphinx.github.io/

possedere al proprio interno tali strutture funzionali per evitare di affidarsi alle risorse del computer dell'utilizzatore.

Un'applicazione Web, infatti non risiede fisicamente nel dispositivo dell'utente sia questo un PC, un *tablet* o uno *smartphone,* ma viene invocata dal *browser*[5] come una normale pagina web attraverso l'inserimento di un indirizzo URL HTTP.[6] La pagina web visualizza solo la componente dell'applicazione necessaria nel momento della richiesta.

Tutta l'applicazione risiede in un server[7] remoto il quale non solo ne contiene i file, ma permette anche l'esecuzione di altri programmi necessari all'applicazione per funzionare quali per esempio il servizio HTTP[8] che gestisce le pagine web, uno o più linguaggi di programmazione e un database[9] per la raccolta delle informazioni.

Lo schema in figura 1 riepiloga la logica classica su cui si basa una normale applicazione Web, ovvero la divisione di competenze *Client-Server*.

[5] Un browser è un'applicazione per il recupero e la navigazione di risorse sul Web. Il browser di base installato sui sistemi operativi Windows è Internet Explorer e più recentemente Edge. Sui sistemi operativi Mac OSX è invece presente di base Safari. Altri browser molto noti sono Google Chrome e Mozilla Firefox.

[6] La locuzione Uniform Resource Locator (in acronimo URL), nella terminologia delle telecomunicazioni e dell'informatica, è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet. L'HyperText Transfer Protocol (HTTP) (protocollo di trasferimento di un ipertesto) è un protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web. Un esempio di URL HTTP è: http://www.google.com.

[7] Un computer che fornisce i dati richiesti da altri elaboratori.

[8] Un software all'interno di un server in grado di gestire le pagine web e fornirle al browser quando vengono richieste. Si veda: Marshall, Brain (2000). How Web Servers Work. 1 Aprile. http://computer.howstuffworks.com/web-server.htm (29/06/2017).

[9] Un'applicazione che consente la creazione, gestione ed interrogazione efficiente dei dati.

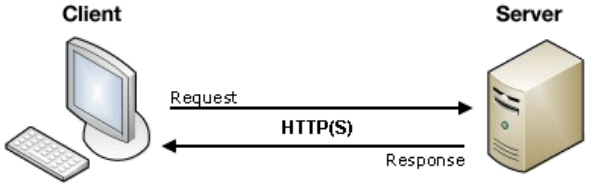

Figura 1. Schema dell'architettura Client-Server

La prima tipologia di sistema di acquisizione fonetica che ho analizzato è rappresentata da Julius (Lee e Kawahara, 2009 e Lee, 2010), un software installabile all'interno di un server. La seconda soluzione, invece, è rappresentata da Web Speech API (Julius, 2013) e non richiede alcuna installazione di programmi perché è già presente all'interno del linguaggio HTML5 (Pilgrim, 2010, pp.15-29).[10]

Julius, le basi

Julius è stato sviluppato principalmente da Lee Akinobu (Nagoya Institute of Technology) e Kawahara Tatsuya (Kyoto University), esiste fin dal 2004 e sebbene supporti molte lingue, gli sviluppatori hanno concentrato le loro energie soprattutto a implementazioni dedicate alla lingua giapponese. È rilasciato come software libero e scaricabile da internet gratuitamente.

Non è lo scopo di questo studio approfondire nel dettaglio il funzionamento dei sistemi di interpretazione del linguaggio naturale, ma possiamo considerarlo un sistema di riconoscimento delle parole pronunciate dall'utilizzatore attraverso la ricerca nella base dati di un suono simile, con successiva verifica del termine corrispondente. Per ottenere questo risultato è necessaria l'implementazione di due modelli: un modello linguistico e un modello acustico.

[10] Il linguaggio HTML5 permette di utilizzare molte periferiche attraverso il browser.

Il modello linguistico è caratterizzato principalmente da una lista ordinata alfabeticamente di tutte le parole che il sistema può riconoscere e le relative probabilità di utilizzo e da uno o più file di regole grammaticali.[11]

Il modello acustico identifica i segmenti che sono presenti nel file audio memorizzati nella base dati, prende una forma d'onda, la divide in segmenti più piccoli, fa un'analisi della frequenza e restituisce una distribuzione probabilistica.

Prendiamo ad esempio la frase «My dog ran away». Il sistema associa ai primi tre segmenti una lista di probabili interpretazioni. Maggiore è la quantità di dati fornita e maggiore è la probabilità di riconoscimento.

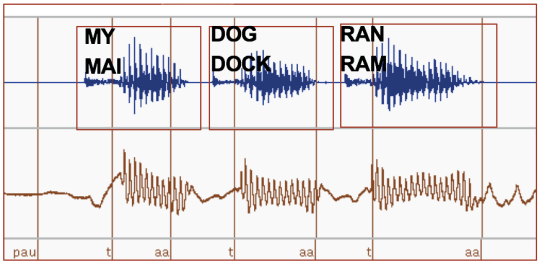

Figura 2. Analisi dei segmenti "My" "Dog" "Ran" attraverso il modello acustico

Julius, implementazione

La creazione del nostro prototipo basato su Julius è avvenuta nelle seguenti fasi e ha coperto un periodo di sviluppo di circa tre giorni.

[11] Per esempi di creazione di regole e di sintassi si veda Bringert, Bjorn (2009). Speech Recognition Grammar Compilation in Grammatical Framework. Department of Computer Science and Engineering Chalmers University of Technology and Goteborg University, Goteborg: Goteborgs Universitet.

1. Configurazione del server atto ad ospitare Julius e installazione dei servizi Web principali. Installazione di Julius e dei modelli acustici e linguistici forniti di base con l'applicativo
2. Creazione di un'applicazione che possa registrare la voce attraverso il *browser* e inviare la forma campionata risultante al *server* con Julius attraverso l'utilizzo della funzione HTML5 MediaDevices.getUserMedia.[12]
3. Esecuzione di Julius, acquisizione del testo risultante e invio dei dati al *browser*.

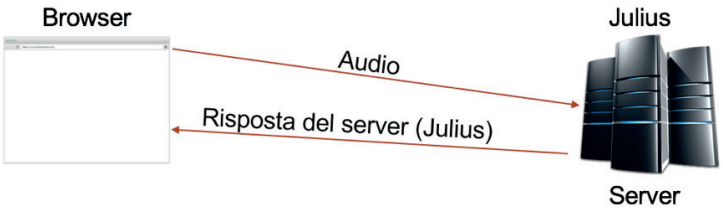

Figura 3. Schema di comunicazione dati Client-Server per il prototipo Julius

Il prototipo che ho creato con l'uso del metalinguaggio HTML e dei linguaggi PHP[13] e JavaScript,[14] ha un'interfaccia essenziale con la presenza di due soli tasti.

[12] Si veda: Bidelman, Eric (2012). Capturing Audio & Video in HTML5.https://www.html5rocks.com/en/tutorials/getusermedia/intro/. (28/06/2017).

[13] PHP è un linguaggio dinamico in grado di connettersi a database e generare pagine HTML potenzialmente differenti ogni volta che si richiama la pagina. Per una introduzione a PHP si veda: Nixon, Robin (2015). Learning PHP, MySQL & JavaScript.Quarta edizione. Sebastopol: O'Reilly, pp. 35-37.

[14] JavaScript è un linguaggio che viene eseguito nel browser e permette di agire sugli elementi della pagina modificandoli in tempo reale. Si veda: Nixon, op.cit, pp. 309-312

Figura 4. Interfaccia del prototipo Julius

Cliccando il tasto "record" è possibile parlare attraverso il microfono e acquisire l'audio. Cliccando il tasto "stop" l'acquisizione viene interrotta e viene visualizzata la trascrizione del testo pronunciato.

Sia per questo prototipo che per quello presentato nelle pagine successive ho preparato il seguente testo composto da frasi più o meno complesse da leggere in successione e da fare trascrivere al computer automaticamente. Considerando una possibile futura implementazione di una interfaccia di questo tipo in un E-learning destinato a studenti di giapponese, mi sono occupato personalmente, in quanto io stesso discente, della lettura del testo da far individuare dal sistema, invece di ricorrere alla collaborazione di un lettore madrelingua. Ho definito inoltre come regola base che se una frase del testo non fosse stata riconosciuta correttamente dal software Julius dopo cinque tentativi, il test sarebbe fallito.

1aこれは本です。	*Kore wa hon desu*	Questo è un libro.
1bこれは本ではありません。	*Kore wa hon dewa arimasen*	Questo non è un libro.
2aこれは誰の傘ですか	*Kore wa dare no kasa desu ka*	Di chi è questo ombrello?
2bこれは私の傘です。	*Kore wa watashi no kasa desu*	Questo è il mio ombrello.
3aこれを着てみてもいいですか	*Kore wo kite mite mo ii desu ka*	Posso provarlo?
3b はい、どうぞ。	*Hai, dōzo*	Si, prego.
3c いいえ、新品ですから。	*IIe, shinpin desu kara*	No, perché è nuovo.
4a 飼ってみたい動物はなんですか	*Katte mitai dōbutsu wa nan desu ka*	Qual è l'animale che vorresti avere?
4b 猫です。	*Neko desu*	Un gatto.
4c 犬です。	*Inu desu*	Un cane.
5経営学の勉強をしようと思っています。	*Keieigaku no benkyō o shiyō to omotte imasu*	Penso che andrò a studiare economia.
6これから授業で面接や小論文の書き方を練習します。	*Korekara jugyō de mensetsu ya shōronbun no kakikata o renshū shimasu*	Ora a lezione faremo esercizio di colloquio e di come si scrivono delle tesine.
7今日はお忙しいところお時間を頂戴しまして申し訳ございません。	*Kyō wa oisogashii tokoro ojikan o chōdai shimashite mōshi wake gozaimasen*	Grazie per avermi dedicato il suo tempo oggi benché (molto) impegnato.

Per gli esempi 3,5,6 si veda: Dalla Chiesa (2014), per l'esempio 3a: Yamamoto, Ueno, Mera (2014), per l'esempio 7: Yumi, Kamiya, Kitamura (2008).

Di seguito i risultati del test con la relativa trascrizione in caratteri latini.

Testo originale	Testo trascritto da Julius
これは本です*Kore wa hon desu*	1: これは今です*Kore wa kon desu* **ERR** 2: これは本です Kore wa hon desu **OK**
これは本ではありません。*Kore wa hon de wa arimasen*	1:これは建立の二つの線*Kore wa kon ryū no futatsu no sen* **ERR** 2: これは今度のあの戦*Kore wa kondo no ano sen* **ERR** 3: これは今ではありません *Kore wa kon de wa arimasen* **ERR** 4: これは本ではありません *Kore wa hon de wa arimasen* **OK**
これは誰の傘ですか*Kore wa dare no kasa desu ka*	1: 彼の差ですか*Kare no sa desu ka* **ERR** 2:これは誰もがそうですか *Kore wa dare mo gasō desu ka* **ERR** 3: これは誰の月ですから。*Kore wa dare no gatsu desu kara* **ERR** 4: これは誰の月ですか。*Kore wa dare no gatsu desu ka* **ERR** 5: これは彼の月ですか。*Kore wa dare no gatsu desu ka* **ERR**

La colonna di destra riporta il numero del tentativo seguito dalla trascrizione effettuata da Julius e il risultato (OK per trascrizione corretta, ERR per trascrizione non corretta). Non essendo il sistema stato in grado di riconoscere l'ultima frase inserita per cinque volte, il test è stato interrotto.

Web Speech API,[15] le basi

Web Speech APIè attualmente implementato nel browser[16] tramite il sistema di apprendimento profondo (*Deep Learning*) realizzato da Google. Questo tipo di sistema di apprendimento automatico da parte del computer si basa sull'utilizzo di reti neurali per l'elaborazione parallela dei dati. Il concetto di reti neurali era stato in parte ipotizzato da Von Neumann fin dagli anni Cinquanta (1958), come un'architettura d'elaborazione tradizionale che prevedeva una unità operativa singola di elaborazione (CPU)[17] e una memoria digitale. Al contrario, oggi l'approccio a reti neurali vede la presenza di diverse unità d'elaborazione che funzionano come strutture indipendenti e che si relazionano tra di loro (Spezziano e Talia, 1999). La distribuzione del carico di lavoro e la parallelizzazione delle attività[18] è effettuata attraverso un sistema *cloud*[19]di 16000 macchine (Hwang e Chen, 2017). In questo caso, il sistema *speech-to-text* si basa sempre sui modelli acustico e linguistico illustrati precedentemente, ma ha due grossi vantaggi: prima di tutto accede alla mole di dati audio e testuali di Youtube, Google Maps e Google.com e, in secondo luogo, il sistema analizzando questa enorme quantità di dati, è in grado di rintracciare al loro interno dei pattern ricorrenti in modo da creare automaticamente l'algoritmo necessario per completare il compito assegnato di riconoscimento dei termini. Maggiori sono i dati e il tempo di esecuzione del programma, maggiori sono i pattern rintracciabili dall'algoritmo. Interessante nota caratteristica di questo approccio è che il risultato finale dell'attività

[15] Acronimo per Application Programming Interface. Si indica una serie di funzionalità disponibili al programmatore, spesso fornite come un pacchetto di funzionalità aggiuntive.

[16] Al momento di produzione di questo studio, è implementato in modalità non sperimentale solo in Chrome.

[17] Central Process Unit, ovvero unità d'elaborazione centrale. Nei computer attuali sono presenti per la maggior parte quelle prodotte da Intel (Pentium, i3, i5, i7) e AMD.

[18] Ovvero l'esecuzione di più attività da unità di elaborazione autonome nello stesso momento.

[19] Cloud: Un sistema di risorse informatiche, solitamente geodistribuite che possono essere utilizzate per il calcolo parallelo e la condivisione di risorse e di carico.

di elaborazione può portare dei risultati differenti a seconda di quando avviene l'elaborazione. Solitamente, inoltre, questi risultati migliorano con il passaggio del tempo.

Web Speech Api, implementazione

L'implementazione in questo caso è avvenuta creando un prototipo senza utilizzare linguaggi presenti nel server come nel caso di Julius. Questo perché l'attività di riconoscimento e di trasformazione del parlato in testo scritto avviene tramite comandi definiti a priori dalle Web Speech API, come indicato nella pagina ufficiale del progetto.[20] È sufficiente quindi scrivere poche righe di JavaScript per invocare le API di riconoscimento.[21]

```
recognition =  new SpeechRecognition();

recognition.lang = "ja-JP";

recognition.continuous = true;

//logica di callback
recognition.onresult = function(e) {
        [...]
}
```

Figura 5. Comandi base per la creazione di un prototipo basato su Web Speech API

La prima riga, recognition = new Speech Recognition () attiva una connessione al motore di riconoscimento, la seconda, recognition.lang = "ja-JP" definisce che la lingua da riconoscere sarà

[20] Per dettagli sull'implementazione delle Web Speech API si veda Shires, Glen; Wennborg,Hans (2012). Web Speech API Specification. https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html (28/6/2017).

[21] L'attività di cattura del testo e rappresentazione a video è un più complessa ma non è oggetto di questo studio.

il giapponese e la terza, recognition.continuous = true indica che il riconoscimento fonetico dev'essere continuo (dettato).

Il codice successivo recognition.onresult = function(e) {} serve a raccogliere il testo convertito e gestirlo, ma non è scopo di questo studio descriverlo nel dettaglio.

Lo schema seguente sintetizza la relazione fra client e server e la relativa chiamata alle API di Google.

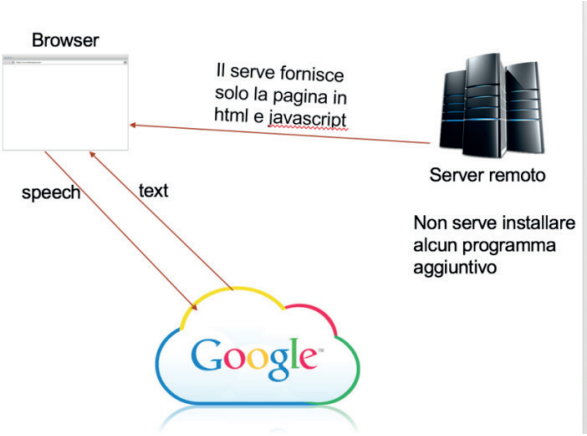

Figura 6. Schema della comunicazione dati Client-Server per il prototipo Web Speech API

Di seguito, una schermata del prototipo realizzato con WEB Speech API. Contrariamente al prototipo creato con Julius descritto in 1.1, in questo caso, poiché è stata attivata la modalità di riconoscimento continuo, non è stato necessario inserire i tasti di inizio e fine e operare su questi ad ogni singola lettura.

Figura 7. Interfaccia del prototipo basato su Web Speech API

Premendo il tasto "Attiva" è stato possibile leggere tutte le frasi in sequenza. Ho comunque atteso circa due secondi dopo ogni frase per attendere che il sistema convertisse la voce in testo e che il mio programma lo visualizzasse. Il testo viene visualizzato nell'area "Processo" durante la lettura, mentre ad ogni pausa il risultato definitivo viene visualizzato in successione nell'area "Risultato".

Il materiale per il test è identico a quello creato per il prototipo Julius e anche in questo caso ho definito un limite di cinque tentativi per frase per decretare il fallimento del test.

Di seguito i risultati del presente test:

Testo originale	Testo trascritto
これは本です*Kore wa hon desu*	1: これは本です。*Kore wa hon desu* OK
これは本ではありません。*Kore wa hon de wa arimasen*	1: これは本ではありません。*Kore wa hon de wa arimasen* OK
これは誰の傘ですか*Kore wa dare no kasa desu ka*	1: これは誰の傘ですか*Kore wa dare no kasa desu ka* OK

これは私の傘です。*Kore wa watashi no kasa desu*	1: これは私の傘です。*Kore wa watashi no kasa desu* OK
これを着てみてもいいですか。*Kore o kite mite mo ii desu ka*	1: これを着てみてもいいですか。*Kore o kite mite mo ii desu ka* OK
はい、どうぞ*Hai, dōzo*	1: はい、どうぞ*Hai, dōzo*OK
いいえ、新品ですから*Iie, shinpin desu kara*	1: いいえ、新品ですから*Iie, shinpin desu kara*OK
飼ってみたい動物はなんですか*Katte mitai dōbutsu wa nan desu ka*	1:行ってみたい動物はなんですか*Itte mitai dōbutsu wa nan desu ka*ERR 2: 買ってみたい動物はなんですか*Katte mitai dōbutsu wa nan desu ka*OK
猫です*Neko desu*	1:猫です*Neko desu* OK*
犬です*Inu desu*	1:犬です*Inu desu*OK
経営学の勉強をしようと思っています。*Keieigaku no benkyō o shiyō to omotte imasu*	1:経営学の勉強をしようと思っています。*Keieigaku no benkyō o shiyō to omotte imasu*OK
これから授業で面接や小論文の書き方を練習します。*Korekara jugyō de mensetsu ya shōronbun non kakikata o renshū shimasu*	1:これから授業で面接や小論文の書き方を練習します。*Korekara jugyō de mensetsu ya shōronbun non kakikata o renshū shimasu* OK
今日はお忙しいところお時間を頂戴しまして申し訳ございません。*Kyo wa oisogashii tokoro ojikan o chōdai shimashite mōshiwake gozaimasen*	1: 今日はお忙しいところお時間を頂戴しまして申し訳ございません。*Kyo wa oisogashii tokoro ojikan o chōdai shimashite mōshiwake gozaimasen*OK

Sebbene 飼ってみたい *katte mitai* (vuoi avere/allevare) venga riconosciuto come 買ってみたい *katte mitai* (vuoi comprare), ho

accettato questa interpretazione poiché la struttura segmentale è identica e dal contesto non era possibile capire a quale trascrizione il parlante facesse riferimento.

Analisi dei risultati

Dall'analisi dei dati suesposti, è evidente che l'implementazione del secondo prototipo WEB Speech API ha portato ai risultati migliori. Mentre il test con Julius, infatti, è fallito alla terza frase, non essendo riuscito il prototipo a riconoscerne per cinque volte la lettura, il prototipo sviluppato con le Web Speech API ha completato il test con un solo errore di riconoscimento.

Ci sono però altri parametri da considerare per poter rispondere adeguatamente alla nostra domanda di ricerca e quindi vagliare quale applicazione sia meglio utilizzare per il riconoscimento vocale: costo tecnologico di implementazione e produzione del software, compatibilità del software con i browser più noti, controllo sulla produzione del testo e privacy.[22]

Il costo di implementazione e manutenzione è la somma dei costi in termini di tempo e denaro per produrre un sistema e continuare ad usarlo nel tempo.[23] Nel caso dell'implementazione di un sistema di riconoscimento vocale basto su Julius, è necessario predisporre un server e installare il programma relativo. Inoltre, visto che il test basato sui modelli linguistico e acustico di base ha dato risultati insoddisfacenti si riterrebbe necessaria una rielaborazione personalizzata dei due modelli: i tempi di implementazione e i costi aumenterebbero in modo esponenziale. Sono poi da considerare i costi dell'infrastruttura di base e il tempo per la creazione dei modelli ex novo che variano in base alla quantità

[22] I costi di implementazione e produzione sono una componente fondamentale del Total Cost of Ownership TCO dell'applicativo, si veda: Doig, Chris (2015). Calculating the total cost of ownership for enterprise software. http://www.cio.com/article/3005705/software/calculating-the-total-cost-of-ownership-for-enterprise-software.html (4/07/2017).

[23] Si veda il sito governativo per l'Italia digitale http://www.agit.gov.it. http://www.agid.gov.it/sites/default/files/documenti_indirizzo/guida_alla_valutazione_dei_costi.pdf

di dati inseriti e della qualità di riconoscimento che il progetto intende offrire.

Utilizzando invece WEB Speech API, tutti questi costi sono praticamente annullati in quanto l'analisi fonetica e la conversione in testo avviene in modo trasparente per il server, non c'è nessun server da installare e nessun modello da creare.

A livello di compatibilità di browser, ci sono due componenti da considerare: il sistema di ricezione della voce e il sistema di trasformazione in testo scritto. Nel caso di Web Speech API, entrambe le attività sono eseguite dal comando Speech Recognition, nel caso di Julius invece l'audio deve essere catturato attraverso la funzionalità html5 MediaDevices.getUserMedia e poi inviato al server in cui è stato preinstallato il programma Julius.

Ad oggi, la funzionalità getUserMedia indispensabile per Julius può essere attivata in tutti i browser più recenti,[24] mentre per utilizzare Web Speech API attraverso il comando SpeechRecognition il supporto completo è offerto solo per il browser Chrome, mentre per Firefox, Safari e Edge è ancora allo stato sperimentale.[25]

Analizziamo ora il problema del controllo sulla produzione. Nel caso di Julius, essendo in grado di selezionare i modelli linguistico e acustico e di crearli in autonomia,è certo che la trascrizione in testo di un determinato input fonetico e un determinato modello acustico/linguistico, sarà sempre uguale. Questo non si può dire per la tecnologia Web SpeechAPI; non è possibile definire un modello acustico o linguistico personale da utilizzare o analizzare il contenuto dei modelli utilizzati. L'attività di conversione viene eseguita in modo totalmente autonomo dal sistema cloud di Google. Inoltre, come già evidenziato in precedenza, il sistema tende a migliorare con il tempo grazie all'analisi continua di flussi di dati sempre crescenti e a riconoscere parole anche se pronunciate con una intonazione non corretta, in quanto anco-

[24] Si faccia riferimento a: "Can I use", Support table for HTML5 CSS3, etc.,http://caniuse.com/#search=getUserMedia per i dettagli sulla compatibilità dei browser.

[25] Si faccia riferimento a: "Can I use", Support table for HTML5 CSS3, etc.: "Ca http://caniuse.com/#search=speechRecognition.

ra privo di configurazioni che consentano l'adeguato riconoscimento dei diversi parametri acustici. Nel caso della frase, *Neko desu* per esempio, durante il mio test ho utilizzato una configurazione tonale scorretta enfatizzando la seconda sillaba *ko,* laddove di solito la configurazione tonale tende ad enfatizzare la prima sillaba *Ne*, ma il sistema ha comunque restituito una trascrizione corretta.

Figura 8. Pronunce della parola Neko

Una tecnologia che non permette il controllo sul flusso di trasformazione voce testo e dell'evoluzione del sistema nel tempo è adeguata per implementazioni di tipo generalizzato, ovvero applicazioni che devono riconoscere più voci e un grande numero di parole quali sistemi di E-learning, esercitazioni automatiche, simulazione di robot vocali/assistenti personali quali Siri o Cortana. Laddove al contrario, ci sia la necessità di riconoscere un limitato numero di comandi vocali, che appartengono a un ristretto numero di operatori e che non necessariamente richiedano una connessione a Internet, una tecnologia sul modello di Julius può rappresentare una scelta più adeguata (esempio: sistemi di antifurto, domotica).

Al problema del mancato controllo del flusso di dati riscontrato per Web Speech API, si collega anche il problema della privacy. Ogni tecnologia di tipo *speech-to-text* che raccoglie dati in un sistema remoto che non possiamo controllare, è un potenziale bacino di dati per agenzie governative e enti di controllo. Queste strutture hanno infatti sia la possibilità di associare un determi-

nato modello di "comportamenti" fonetici a una o più tipologie di persone e sia di utilizzare tecnologie di geolocalizzazione e di *data retention*[26]. L'utilizzo combinato di queste tecnologie può essere estremamente invasivo non solo a livello di privacy ma anche a livello di diritti umani e libertà di parola (Pastukhov e Kindt, 2016).

Al contrario, un modello tradizionale tipo Julius, proprio perché permette di mantenere il controllo dei dati in un server può garantire un maggiore rispetto dei dati personali.

Conclusioni

In risposta alla nostra domanda di ricerca, lo studio appena svolto dimostra come una soluzione tecnologica *speech-to-text* basata su tecnologia a modello di reti neurali (Web Speech API) offra buoni risultati ad un costo di implementazione software relativamente basso.

Sebbene esistano molte aziende che propongono soluzioni di riconoscimento vocale sotto forma di API per l'implementazione in applicazioni web,[27] essendo la soluzione WEBSpeechAPI proposta da Google gratuita e implementabile direttamente con l'utilizzo di HTML5, essa rappresenta sicuramente un'opzione interessante.

Nell'analisi di fattibilità dell'applicativo è necessario però valutare in quale tipo di software possa essere accettata una soluzione come questa, che ammette più configurazioni tonali come ugualmente valide favorendo l'integrazione, mettendo però in discussione qualsiasi concetto di 'unica lingua standard'.

[26] Si intende la possibilità di raccogliere e immagazzinare un grandissimo numero di dati sulle inclinazioni linguistiche dei parlanti e incrociare questi dati con quelli relativi alla loro posizione geografica

[27] Si veda per esempio la pagina web IBM Watson Cloud Service: https://www.ibm.com/watson/developercloud/speech-to-text.html oppure i servizi LEX di Amazon AWS: https://aws.amazon.com/it/blogs/aws/amazon-lex-build-conversational-voice-text-interfaces/

Riferimenti Bibliografici

Bunka Institute of Language; Simone Dalla Chiesa (a cura di) (2004). *Shinbunka Shokyū Nihongo 1*. Venezia: Libreria Editrice Cafoscarina.

Bunka Institute of Language; Simone Dalla Chiesa (a cura di) (2004). *Shinbunka Shokyū Nihongo 2*.Venezia: Libreria Editrice Cafoscarina.

Feng, Wu Chang; Brandt, David; Debanjan, Saha (2007). "A long-term study of a popular MMORPG". *NetGames* 07, pp. 19-24.

Hwang, Kai; Chen, Min (2017). *Big-Data Analytics for Cloud, IoT and Cognitive Computing*. Chichester: John Wiley & Sons.

Julius, Adorf. 2013. *Web Speech API*. KTH Royal Institute of Technology, Stockholm: KTH Royal Institute of Technology, pp 1-5. http://www.juliusadorf.com/pub/web-speech-api.pdf.

Kaplan, Andreas M.; Haenlein, Michael (2009). "The fairyland of Second Life: Virtual social worlds and how to use them". *Business Horizons*, 52, 6, pp. 563-572.

Lee, Akinobu (2010). "The Julius Book". *Julius*. Disponibile online all'indirizzo https://julius.osdn.jp/juliusbook/en/ (2/06/2017).

Lee, Akinobu; Kawahara Tatsuya (2009). "Recent Development of Open-Source Speech Recognition Engine Julius". *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* . Sapporo: APSIPA ASC.

Pastukhov, Oleksandr; Kindt Els (2016). *Voice Recognition: Risks To Our Privacy.* Disponibile online all'indirizzo https://www.forbes.com/sites/realspin/2016/10/06/voice-recognition-every-single-day-every-word-you-say/#3d11ab98786d (25/06/2017).

Pilgrim, Mark. 2010. *HTML5 Up and Running.* Sebastopol, CA: O'Reilly.

Segawa, Yumi; Kamiya, Sachiko; KitamuraSadayuki (2008). *Hito o ugokasu! Jissenbijinesunihongokaiwajōkyū,* Tokyo, Surīēnettowāku.

Spezziano, Giandomenico; Talia, Domenico (1999). *Calcolo Parallelo, Automi cellulari e modelli per sistemi complessi.* Settima edizione. Franco Angeli.

Yamamoto, Emi; Ueno, Junko; Mera, Yoshie (2014). *Ohisama*. Tokyo: Kuroshio shuppan.

Von Neumann, John (1958). *The Computer and the Brain.* London: Yale Univesity Press.

Phonetic recognition of Japanese language through digital tools: prototype development and results analysis

Nowadays technology permits to develop interfaces that allow speech recognition and its transformation into text. This type of solution can be very interesting for the deployment of E-learning systems for Japanese language exercises or self-evaluation tests.

In the present study, we will analyze the result in testing two prototypes developed with two different approaches: one based on a traditional client server communication implementation called Julius and one based on a neural network implementation in a cloud architecture mainly managed by Google. The analysis phase through the application of the same test on both prototypes will allow to understand the strengths and weaknesses of the two different solutions and the possible application scope.

デジタル技術による日本語音声認識。
プロトタイプの開発と結果分析

マンテッリ・アレッサンドロ

現在、情報処理技術発展により、音声認識とテキスト変換（スピーチ・テキスト）ができるインタフェースの開発が可能である。会話の練習や自己評価テストなどを含む日本語のイーラーニングの開発を目指すのに、幅広く役立つソリューションである。本論ではJuliusという音声認識ソフトを組み込む一般的なクライアント・サーバアプローチと、グーグル社のクラウドの元で機能するディープラーニングシステムのアプローチで開発されたプロトタイプに与えた音声認識テストの結果を分析する。これと共に、用いた技術の長所、弱点、利便性と適用分野について考察する。