Interfacce Vocali

Riconoscimento Comprensione e Sintesi
Il caso Mycroft

Alessandro Falaschi

Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni - DIET
Università La Sapienza di Roma

Pubblicato su TeoriadeiSegnali.it

Contributo al corso di Tecniche Audio Video - Prof. Aurelio Uncini

Dicembre 2020

Introduzione

Con il pretesto di analizzare i componenti che permettono
il funzionamento di un assistente vocale completamente open source
viene sviluppato un percorso divulgativo nel settore
del trattamento del segnale vocale ai fini dell'interfaccia uomo-macchina

Sebbene tale scopo sia stato perseguito da diversi decenni
solo di recente la potenza di calcolo e le tecniche neurali
hanno reso possibile avvicinarsi realisticamente al suo compimento

La presentazione si sviluppa pertanto in forma di un excursus storico
tentando di palesare le basi teoriche e conoscitive
che le reti neurali si dimostrano in grado di fare proprie
sulla scorta della disponibilità di adeguate basi di dati

Una particolare attenzione viene riservata ai collegamenti web
che permettono di approfondire gli argomenti esposti

Indice dei contenuti

Ora parliamo di...

Da tastiera e schermo, alla voce

L'interfaccia uomo-macchina si fa più umana

- In 50 anni siamo passati da schede perforate e stampanti, a
  - terminali ('70), GUI con mouse e finestre ('80), touchscreen ('00)
- Prima si impara a parlare, poi a leggere e scrivere
- Il genere umano si distingue da altri animali per il linguaggio
  - una manifestazione di *intelligenza*
- La comunicazione vocale è applicabile a tutte le attività assistite da macchine ed automi, come
  - ricerche vocali e geografiche, immissione dati (già realtà)
  - IoT e domotica
  - autoveicoli anche a guida autonoma
  - ... badante robotica/o... (ci arriveremo?)
- Sussistono barriere psicologiche alle interfacce vocali
  - per compiti semplici appare preferibile agire manualmente
- Gli assistenti vocali possono fare da *apripista*

Ora parliamo di...

Le applicazioni esistenti

- *Google* e *Apple* già presenti su telefoni, altoparlanti e auto
- *Amazon* e *Spotify* oltre all'audio entrano in auto
- *Microsoft* continua su PC con *Cortana*
- *Samsung* parte da televisori e frigoriferi con *Bixby*
- Nella mischia anche Russi e Cinesi con **Alice** (*Yandex*), **Xiaowei** (*Tencent*), **AliGenie** (*Alibaba*), **Duer** (*Baidu*)

Mycroft

Nella tabella di Wikipedia, **Mycroft**[a] è l'unico assistente vocale completamente *open source* sia per HW che per SW, il sogno di developers, makers and hackers

[a] Ispirato all'uso del nome dal fratello di Sherlock Holmes in un film

Ora parliamo di...

Il caso Mycroft

Gira su Linux, Raspberry Pi e Android. Video

Mark I (2015)

Fuori produzione. Monta Raspberry Pi 3, speaker, uscite audio RCA, display LED 8×32, due «occhi» NeoPixel, WiFi, 10/100 Eth, porta HDMI debug, 4 porte USB, connettore GPIO 40 pin, Arduino Mini

Mark II (2021)

Previsto per il 2021, consegue lo stato dell'arte per array di microfoni, beamforming e cancellazione attiva del rumore. Provvisto di schermo e di un buon altoparlante, *filling your room with vivid highs and deep lows*

Ora parliamo di...

Open Source, Cloud e Privacy

Prima di iniziare il viaggio tra le tecnologie abilitanti, alcune riflessioni

- Nucleo e componenti di Mycroft sono distribuiti via GitHub
- Il codice rilasciato con licenza *aperta* permette di studiarne il funzionamento ed adattarlo a nuove idee
- Con quasi 7000 diverse lingue parlate sul pianeta, ognuno deve essere in grado di adattare la tecnologia al proprio idioma
- **Privacy:** con l'assistente in salotto od in camera da letto, siamo sicuri di non avere problemi a lasciare che *nel Cloud* si ascolti tutto ciò che viene detto? ...non solo per la pubblicità che potrebbero mandarci..
- **Sicurezza:** ... ma chi ci può assicurare che i nostri dialoghi non siano ascoltati da terze parti, e che le relative registrazioni non possano essere hackerate?

Alternative a Mycroft

Una ricerca svela un numero inatteso di iniziative simili!

- Picovoice
- Jasper
- Naomi
- Rhasspy
- LinTo
- Almond
- OpenAssistant
- Aimybox
- Olivia
- Stephanie
- Jarvis
- Leon
- Alice
- Jovo
- Python-ai-assistant
- Voice engine
- OpenVoiceOS
- Kalliope
- Dragonfire
- Hey Athena

Ora parliamo di...

Lo Stack Vocale

ovvero le tecnologie che abilitano l'operatività di un assistente vocale

Tratto dal whitepaper di Mycroft

- **Wake Word:** la *parola magica* che *risveglia* l'assistente. E' essenziale che la sua detezione avvenga *tra le mura di casa*
- **Speech to Text (STT):** la fase *più impegnativa*, in cui dalla frase pronunciata dopo la **WW** si risale alla sua trascrizione ortografica
- **Intent parsing:** la selezione dell'azione da svolgere, individuando *keyword* (**KW**) (verbi, sostantivi) presenti nella frase, o mediante tecniche neurali
- **Skill invocation:** noto l'intento si esegue l'agente che può esaudirlo, e che in base alle **KW** (eventualmente) interroga basi di dati esterne, e formula una risposta
- **Text to Speech (TTS):** la pronuncia della risposta dello skill

Aspetti sistemistici

Cos'altro serve per poter funzionare

- Comunicazione tra processi
  - i diversi componenti sono processi indipendenti, che comunicano in modo asincrono attraverso messaggi JSON scambiati su di un message bus implementato mediante websocket con un processo core
  - Ad esempio, alla fine del bucato la lavatrice invia un messaggio sul bus, producendo una notifica *consumata* dalla sintesi vocale
- Internazionalizzazione
  - oltre alla lingua usata dal TTS e dal riconoscitore vocale, occorre adattare le regole e i dialoghi degli skill, e la pronuncia dei numeri
- Registrazione del *device* presso una Home, che
  - svolge il ruolo di proxy anonimizzante per le interazioni con la rete
  - gestisce la configurazione del device
  - permette l'installazione degli skill disponibili

Ora parliamo di...

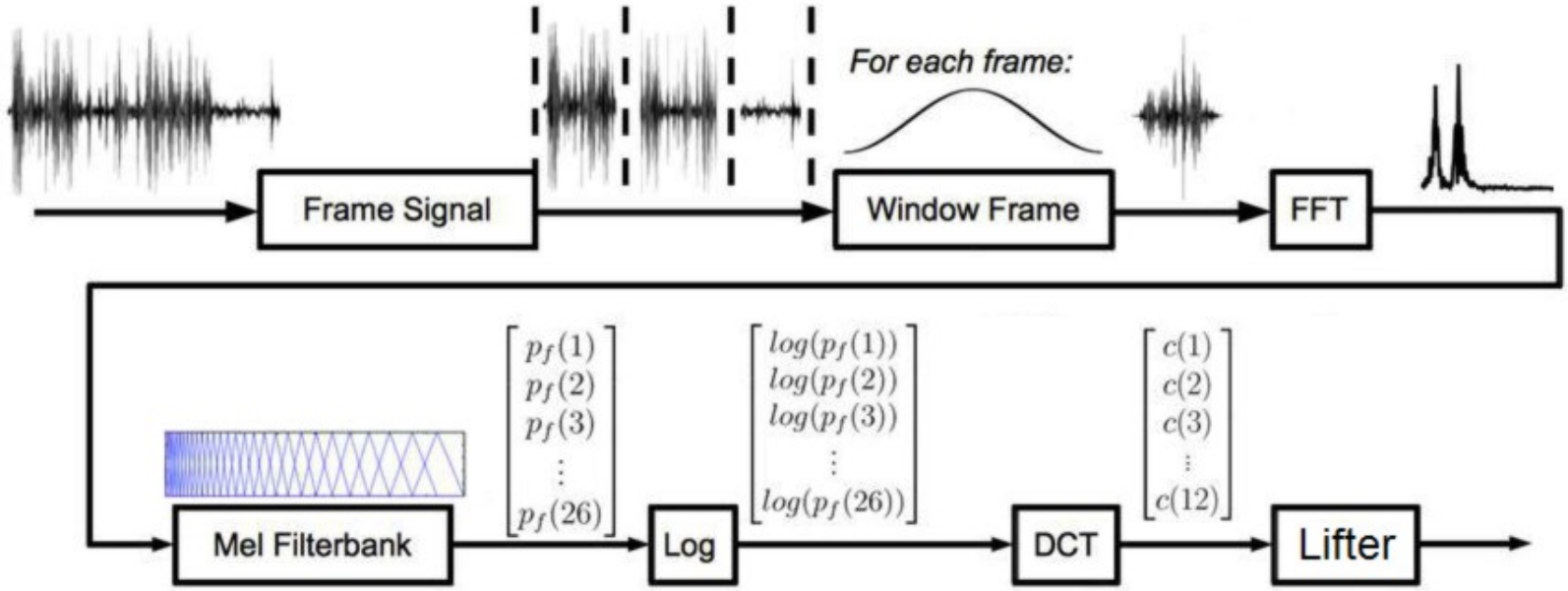
Detezione della parola di risveglio o WakeWord

- Processo eseguito di continuo *in locale*, mediante uno tra
  - **PocketSphinx**: basato su **HMM**, in cui il modello di parola è la concatenazione di modelli fonetici pre-costituiti, *oppure*
  - **Precise**: adotta una **NN** con una singola *Gated Recurrent Unit* (**GRU**) che codifica i **MFCC** della pronuncia in un vettore di 20 features, che alimentano uno strato *denso* il cui unico neurone indica la detezione della **WW**, ed i cui parametri sono *appresi* a partire da un *dataset*
- E' una classificazione *aperta* tra una parola ed *il resto del mondo*
- Il dataset di apprendimento per una nuova **WW** *di Precise* contempla sia ripetizioni della **WW** che di non-**WW** (es. audiolibri)
- E' in corso la raccolta volontaria di **WW** italiane pronunciate da diversi locutori - *partecipa!*

Allenamento di Mycroft Precise - I

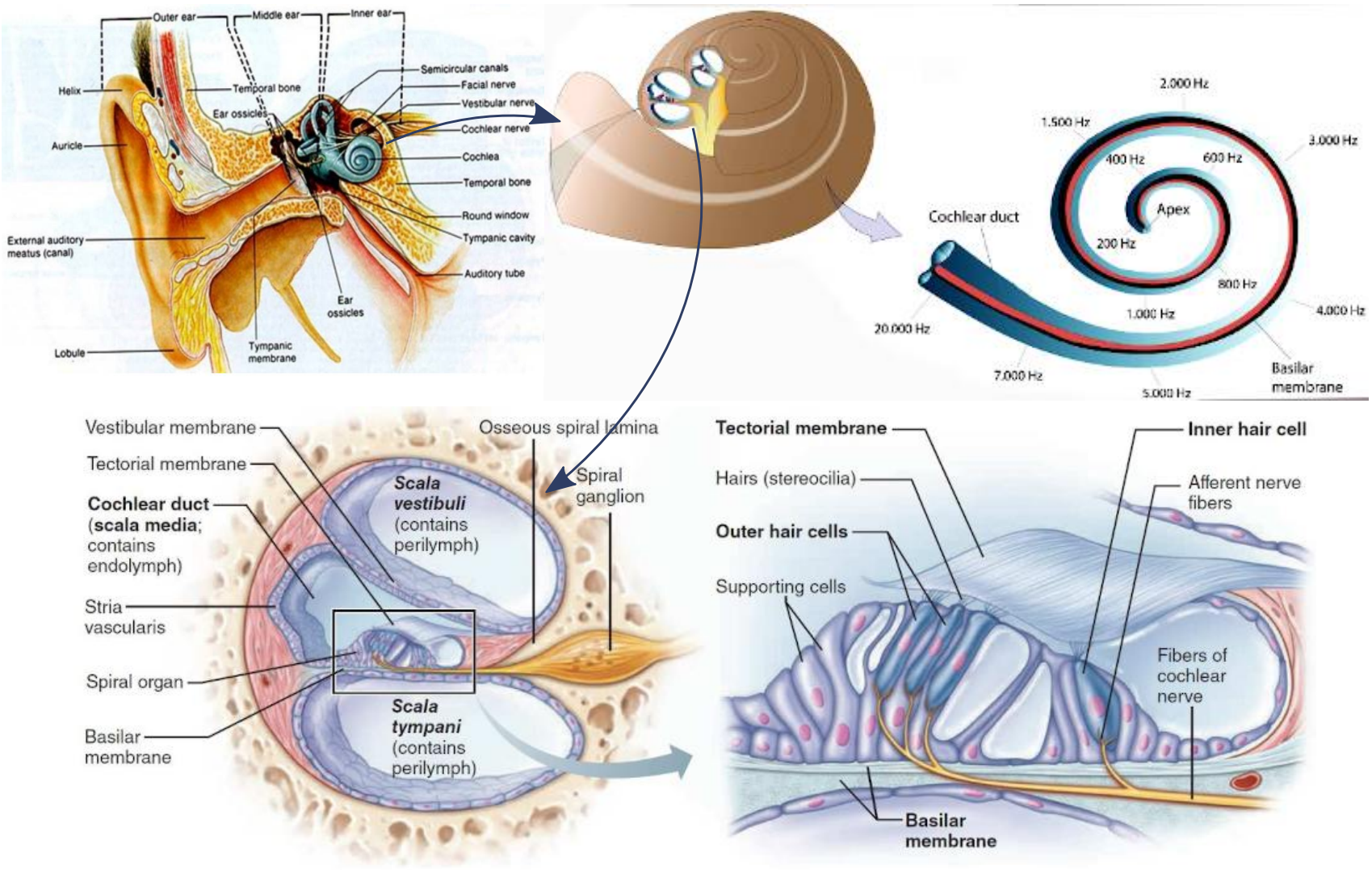
Pre-processing

- Ogni 50 ms una finestra di 100 ms di segnale vocale è parametrizzata con 13 coefficienti spettrali MEL-Cepstrum (MFCC)
  - anni di precedenti ricerche hanno promosso gli MFCC come la rappresentazione più performante agli scopi del riconoscimento vocale



Sistema uditivo

La scala di frequenze MEL trova motivazione nella struttura della coclea



la sensazione uditiva è prodotta dalle cellule ciliate interne

Allenamento di Precise - II

Il codice di *Precise* si basa su *Tensorflow/Keras*, individuando i passi

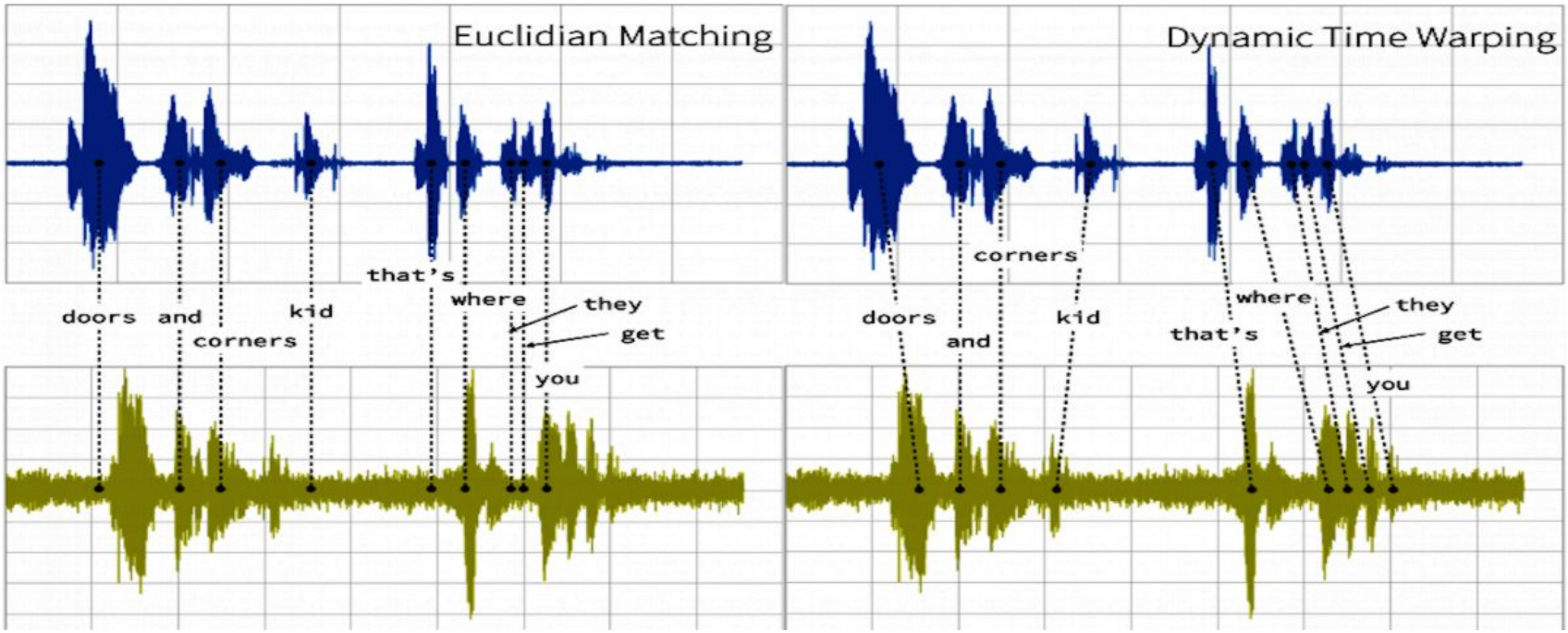
1. ripartizione del dataset **WW** e non-**WW** nei set *train* e *test*
2. `precise-train` $\rightarrow$ uso del set *train* (**WW** e non) che usa solamente l'ultimo secondo e $\frac{1}{2}$ dei files audio
3. `precise-test` $\rightarrow$ valuta l'accuratezza sul set *test* (**WW** e non)
   - esiti possibili: vero e falso positivo, vero e falso negativo
   - si osserva una sensibilità eccessiva, con troppi falsi positivi
4. `precise-train-incremental`
   - qui i files non-**WW**/*train* sono usati *per intero* come *test*, ed ogni volta che si verifica un falso positivo il *corrispondente* secondo e $\frac{1}{2}$ di audio viene *ritagliato* e salvato nel set non-**WW**/*train*
   - viene quindi lanciata una nuova fase di training
5. `precise-test` $\rightarrow$ verifica dei miglioramenti, in assenza dei quali occorre includere ulteriori dati di apprendimento e riprendere da 4)

Ora parliamo di...

La natura *elastica* della voce

In principio fu il Dynamic Time Warping - per parole *isolate*, fine anni 70

- Distanza euclidea tra vettori MFCC: $d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\Sigma_i (x_i - y_i)^2}$
- Ma la voce è *elastica*, e la pronuncia può accelerare e rallentare



- Il DTW individua l'allineamento *ottimo* a cui corrisponde la minima distanza complessiva $\rightarrow$ programmazione dinamica (Bellam) o percorso di minimo costo (Dijkstra)

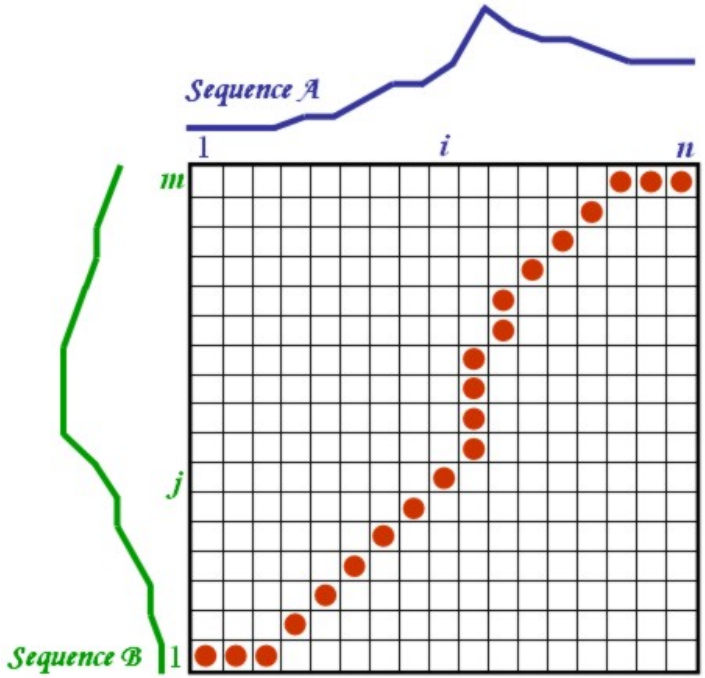
Il Dynamic Time Warping

come migliore allineamento tra sequenze

- La distanza tra pronuncia $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^n\}$ e *template* $\boldsymbol{Z} = \{\boldsymbol{z}^1, \boldsymbol{z}^2, \cdots, \boldsymbol{z}^m\}$ è pari alla minima tra gli allineamenti possibili

$$d_{DTW}(\boldsymbol{X}, \boldsymbol{Z}) = \min_{warp} \left\{ \sum_{j=1}^{\max(n,m)} d\left(\boldsymbol{x}^{i=warp(j)}, \boldsymbol{z}^{i=warp(j)}\right) \right\}$$

- calcolate le $n \times m$ distanze $d_{i,j} = d(\boldsymbol{x}^i, \boldsymbol{z}^j)$ la dist. minima e la funzione *warp* sono ottenute iterando $d^m_{i,j} = d_{i,j} + \min(d^m_{i,j-1}, d^m_{i-1,j-1}, d^m_{i-1,j})$
- si termina con $d_{DTW} = d^m_{n,m}$
- Il template $\boldsymbol{Z}$ *(ipotesi)* con la d_{DTW} più piccola *vince*
- per approfondimenti, qui

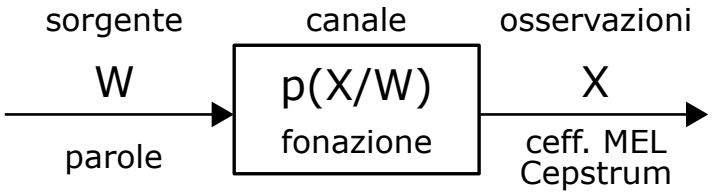


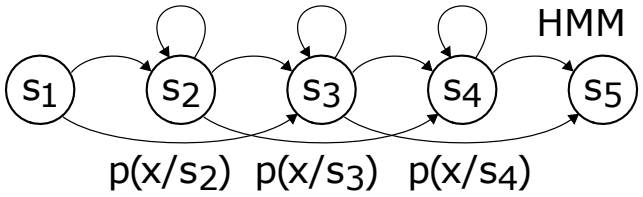
Gli Hidden Markov Model

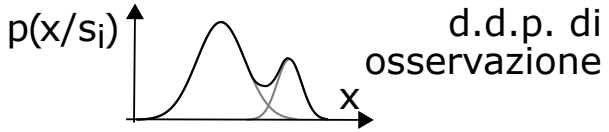
verso il *Machine Learning* (anni 80)

Per gestire un aumento di parlatori e condizioni acustiche, il DTW non può fare altro che aumentare il numero di template

- subentra una impostazione *comunicazionistica*, ossia una *verifica di ipotesi* che una sequenza di *unità linguistiche* $\boldsymbol{W}$ generi le osservazioni $\boldsymbol{X}$
- le probabilità *in avanti* $p(\boldsymbol{X}/\boldsymbol{W})$ sono valutate mediante *Hidden Markov Models* (HMM) - ad es. *left-to-rigth* come in fig
- per ogni stato s_i di $\boldsymbol{W}$ una d.d.p *mistura gaussiana* $p(x/s_i)$ fa da *modello acustico* delle osservazioni $\boldsymbol{x}$ per lo stato







- le transizioni tra stati modellano le variabilità *temporale* e di *pronuncia*

I parametri $\boldsymbol{\theta}^*$ *ottimi* degli HMM (prob. di transizione e densità di osservazione) si ottengono come $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{p(\boldsymbol{X}_{train}/\boldsymbol{W}_{train}, \boldsymbol{\theta})\}$ *(stima di ML)*

Riconoscimento con HMM

Unità fonetiche, modello di linguaggio e decodifica di Viterbi

- Un HMM_W rappresenta *parole intere*; al crescere del vocabolario si ottiene *concatenando* gli HMM_{subw} di *sottounità* (sillabe o fonemi)
- la prob. *a priori* $p(\boldsymbol{W})$ delle frasi è data da un *modello di linguaggio* di tipo *automa markoviano*, ottenendo $p(\boldsymbol{W}) = \prod_i p(w_i/w_{i-1})$, con memoria più estesa, o definita da una grammatica
- Il riconoscimento vuole trovare $\boldsymbol{W}^* = \arg\max_W \{p(\boldsymbol{W}/\boldsymbol{X})\}$ *(MAP)* ma applicando il teorema di Bayes possiamo scrivere

$$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \{p(\boldsymbol{X}/\boldsymbol{W}) \cdot p(\boldsymbol{W})\}$$

- Gli HMM sono *compilati* assieme al modello di linguaggio in un unico grafo, ed il riconoscimento diviene trovare la sequenza di stati

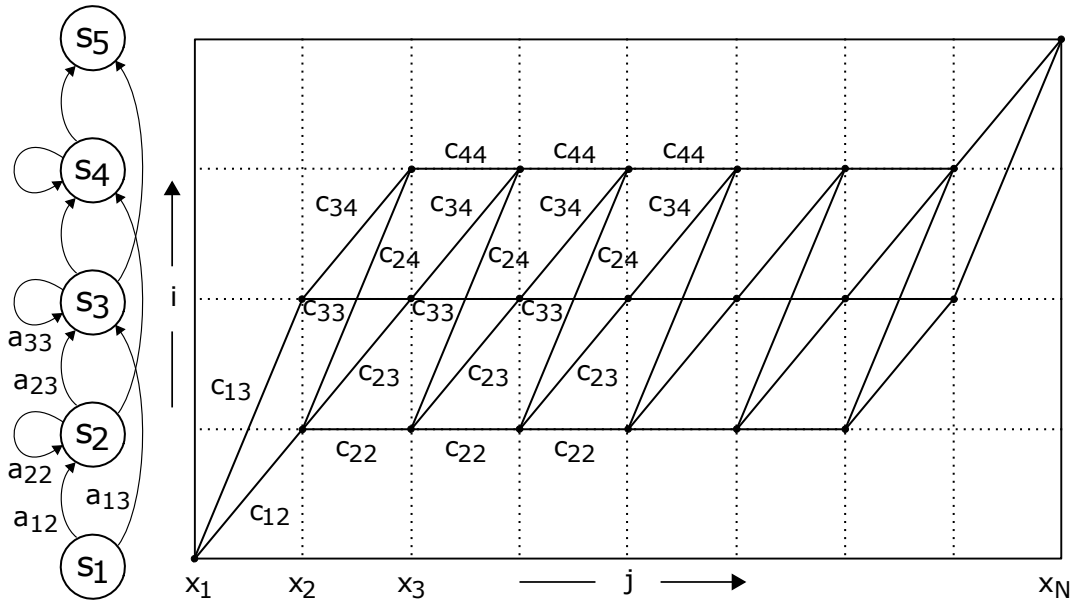
$$\boldsymbol{S}^* = \arg\max_{\boldsymbol{S}} \{p(\boldsymbol{X}, \boldsymbol{S}/\boldsymbol{W}(\boldsymbol{S})) \cdot p(\boldsymbol{W}(\boldsymbol{S}))\}$$

- Passando ai logaritmi la produttoria di prob. diviene una sommatoria, e la massimizzazione diviene una minimizzazione, ottenuta mediante l'algoritmo di Viterbi eseguito sul *traliccio* associato al grafo

Algoritmo di Viterbi

Percorso ottimo in un grafo con costi

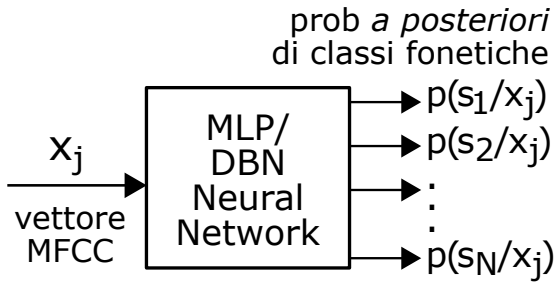
- Il *traliccio* ha righe come gli stati dell'HMM, colonne come i vettori $\boldsymbol{x}^i$, e collegamenti tra nodi con costo $c_{ij} = -\log a_{ij}$ in cui $a_{ij} = p\left(s_j / s_i\right)$
- attraversare un nodo ha un costo $d_{i,j} = -\log p\left(x_j / s_i\right)$
- per ogni nodo il costo minimo è $C_{i,j} = d_{i,j} + \min_k \left\{C_{k,j-1} + c_{ki}\right\}$ da calcolare iterando sulle colonne e quindi sulle righe
- la sequenza di stati lungo il percorso *migliore* si ottiene seguendo *a ritroso* le scelte ottime che portano all'ultimo nodo (in alto a destra)
- si ottiene così una *segmentazione* di $\boldsymbol{x}$ nei termini di $\boldsymbol{S}$ (e $\boldsymbol{W}$)



L'approccio HMM-DNN

Anni '90, il ponte tra HMM e NN

- L'uso degli HMM soffre dei problemi
  - le prob. $p(x_j/s_i)$ dipendono solo dallo stato s_i e non dal contesto $x_{k\neq j}$
  - la fase di training non è *discriminativa*, in quanto la stima ML di $\boldsymbol{\theta}^*$ *non tiene conto* dei modelli che *competono* con quello allenato
  - la scelta delle sub-unità fonetiche, della topologia, del numero di stati e di modi gaussiani possono introdurre *forzature* arbitrarie
- Si fa strada l'idea di sostituire la mistura di gaussiane del *modello acustico* $p(x_j/s_i)$ (HMM-GMM) con una rete neurale (HMM-DNN)
- La NN viene allenata in modo supervisionato in base alla segmentazione di $\boldsymbol{X}$ nei termini della seq. di *classi fonetiche* associate agli stati degli HMM - stima MAP e discriminativa



- Il risultato è la stima delle prob. *a posteriori* $p(s_i/x_j)$, da cui ottenere

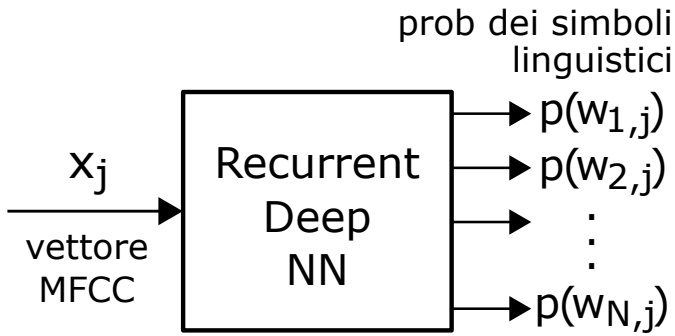
$$p(x_j/s_i) = \frac{p(s_i/x_j)p(x_j)}{p(s_j)} \;\alpha\; \frac{p(s_i/x_j)}{p(s_j)}$$

utilizzata negli HMM (le $p(x_j)$ non contribuiscono al riconoscimento)

L'approccio end-to-end

Fa tutto la rete neurale

- Si evita di postulare un modello di pronuncia, e la NN viene allenata ponendo come obiettivo direttamente la trascrizione ortografica $\boldsymbol{W}$
- L'adozione di unità *ricorrenti* consente di modellare gli aspetti di correlazione temporale del segnale $\boldsymbol{x}$
- All'uscita della RNN si hanno le *probabilità* dei simboli linguistici w, parole o... simboli ortografici!

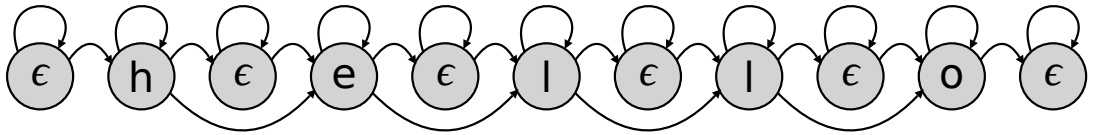


- La sequenza $\boldsymbol{X}$ produce una matrice $p(w_{i,j})$, usata per valutare
  - la loss function da usare per l'apprendimento
  - il risultato del riconoscimento, in accordo ad un *modello di linguaggio*
- Sembra incredibile, ma il problema a questo punto sono *"le doppie"*
  - es. $W = pollo \longrightarrow pppooooollllloooooo \longrightarrow polo$ (!)
  - per questo si introduce il simbolo opzionale ϵ tra ogni lettera
  - $\epsilon\epsilon\epsilon\epsilon ppp\epsilon\epsilon ooooo\epsilon\epsilon ll\epsilon lll\epsilon ooooo \longrightarrow \epsilon p\epsilon o\epsilon l\epsilon l\epsilon o \longrightarrow pollo$
  - prima fondiamo le ripetizioni, poi rimuoviamo le ϵ

Connectionist Temporal Classification - CTC

Il successore di DTW, Viterbi e Baum-Welsh

- Il CTC allinea sequenze $\boldsymbol{X}$ e $\boldsymbol{W}$ di diversa lunghezza nei modi possibili, contemplando il simbolo opzionale ϵ tra coppie di caratteri w

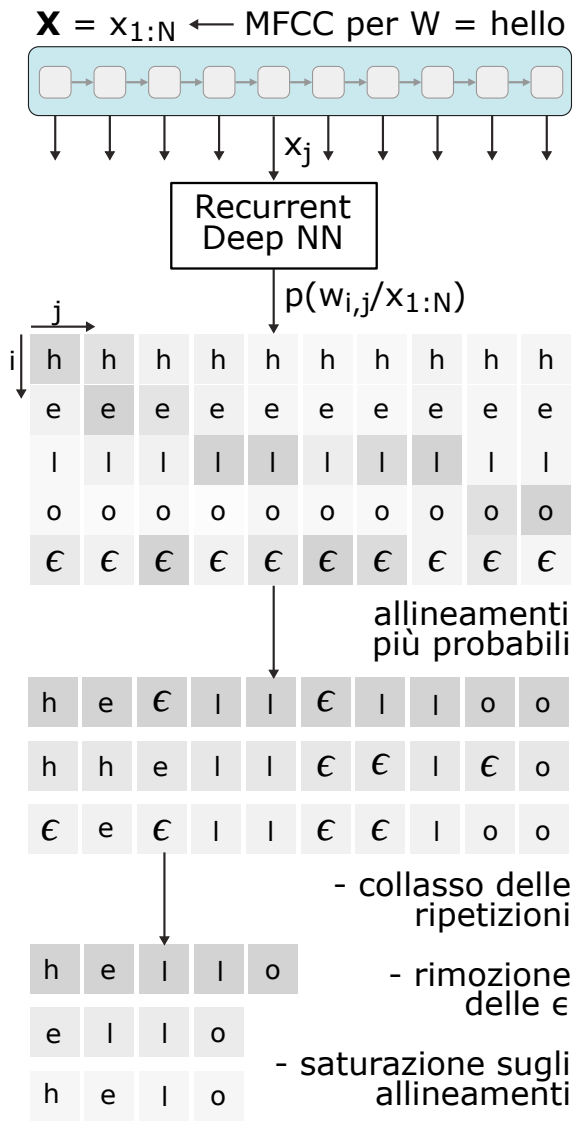


possibili sequenze w per la parola *"hello"* con ϵ opzionale tra lettere

- ciò permette di allenare la NN su materiale *non segmentato*
- La $p\left(W_{train}/X_{train}\right)$ è la *saturazione* di $\prod_{j=1}^{N} p\left(w_{i,j}/X\right)$ sui possibili allineamenti $\mathcal{A}$

$$p\left(W/X\right) = \sum_{a\in\mathcal{A}} \prod_{j=1}^{N} p\left(w_{i(a),j}/X\right)$$

  - calcolata via programmazione dinamica
  - è differenziabile, ed usata per la backpropagation
  - il gradiente è calcolato applicando **CTC** ad uno schema forward-backward



Inferenza con CTC

- Terminato il training, vogliamo trovare $\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} \{p(\boldsymbol{W}/\boldsymbol{X})\}$
- Diversi allineamenti producono la stessa sequenza $\boldsymbol{W}$: occorre sommare le rispettive probabilità
- Riduzione di complessità: si effettua
  - una ricerca *beam search* estendendo ad ogni istante solo le ipotesi più promettenti
  - la ricombinazione delle ipotesi che condividono lo stesso prefisso dopo semplificazione CTC
- Modello di linguaggio e compensazione della lunghezza
  - le seq. $\boldsymbol{W}$ che non esistono nel lessico o nella sintassi vanno scartate
    $$\boldsymbol{W}^* = \arg\max_{W} \{p(\boldsymbol{W}/\boldsymbol{X}) \cdot p(\boldsymbol{W})^{\alpha}\}$$
  - ora le ipotesi con più parole sono penalizzate, occorre una correzione
    $$\boldsymbol{W}^* = \arg\max_{W} \left\{p(\boldsymbol{W}/\boldsymbol{X}) \cdot p(\boldsymbol{W})^{\alpha} \cdot \mathit{Length}(\boldsymbol{W})^{\beta}\right\}$$
  - α e β determinati a tentativi
- CTC è stato adottato in altri ambiti, come lettura labiale e riconoscimento di azioni da video, detezione di wakeword audio, riconoscimento calligrafico

Riconoscimento vocale di Mycroft

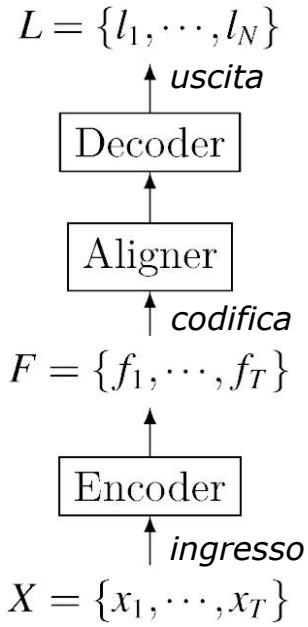
- A fronte della accuratezza necessaria, Mycroft usa di default il servizio di Google, ma anonimizza i files audio attraverso un suo proxy dal quale passano tutte le richieste dei devices accoppiati
- In alternativa sono configurabili altri servizi *on-line*: GoVivace, Google Cloud, Houndify, IBM Cloud, Microsoft Azure, Yandex (tutti più o meno a pagamento), Wit.ai di Facebook - free
- Oppure possono essere installati server STT per un uso *off-line*:
  - Mozilla DeepSpeech che usa il CTC prima discusso. Vedi anche deepspeech-server project su *PyPI*, il modello per l'italiano, ed il sito per contribuire alla raccolta del dataset di apprendimento
  - Kaldi (GitHubGitHub) - ma non è chiaro se supporta l'italiano (forse qui?)

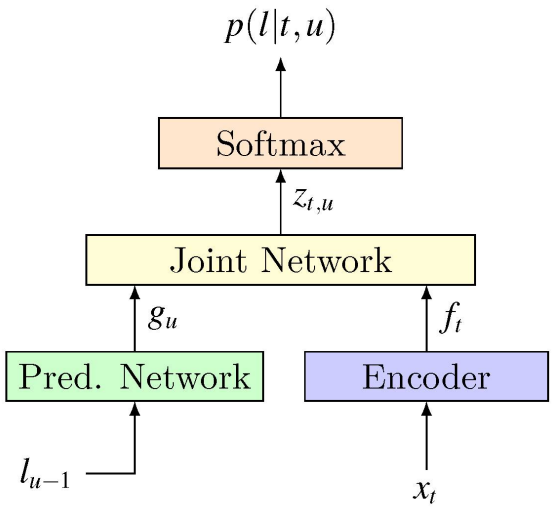
Di meglio, si può

L'influenza di seq2seq e dei transformer

Sebbene l'approccio R(C,D)NN-CTC adottato da *DeepSpeech* dia ottimi risultati, non è esente da critiche

- le probabilità $p(w_{i,j}/X)$ non tengono conto dei caratteri adiacenti: il CTC non può apprendere un modello di linguaggio
- i vincoli lessicali e sintattici subentrano a giochi fatti
- la sequenza $\boldsymbol{W}$ deve essere lunga $\leq$ di $\boldsymbol{X}$





L'alternativa RNN-Transducer è composta da

- un encoder $\boldsymbol{f}_t = \mathcal{F}(\boldsymbol{x}_t)$ *(mod. acustico)*
- un predittore $\boldsymbol{g}_u = \mathcal{P}(\boldsymbol{h}_u)$, di tipo ricorsivo con stato interno $h_u(h_{u-1}, l_{u-1})$ dove u segue gli indici di uscita *(mod. di linguaggio)*
- un *combinatore* $\boldsymbol{z}_{u,t} = \mathcal{J}(\boldsymbol{f}_t, \boldsymbol{g}_u)$ che produce la probabilità $p\left(l^k/t,u\right)$ dei simboli l^k per ogni istante di ingresso t e di uscita u

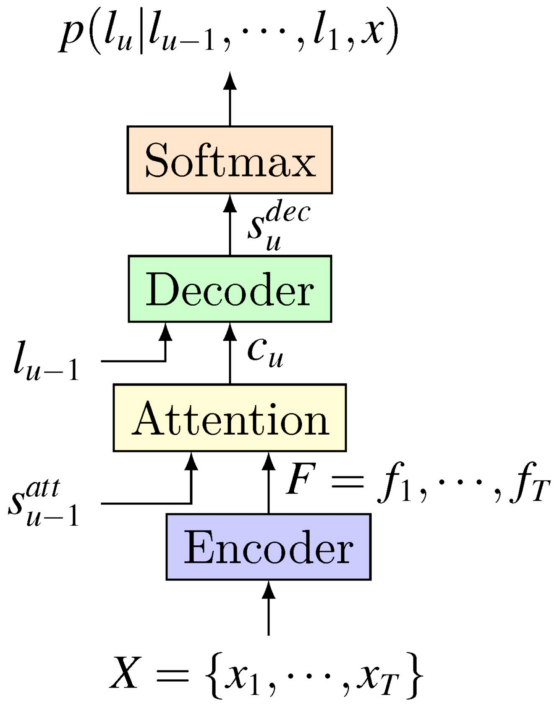
RNN-T e le reti con attenzione

L'RNN-Transducer

- fa uso anch'esso di CTC per apprendimento ed inferenza
- adotta come encoder una RNN *bidirezionale*, che osserva tutta la seq. $\boldsymbol{X}$, perciò non funziona in *real time* (ma sono in corso esperimenti)
- presenta difficoltà di apprendimento, e produce molte ipotesi di sequenze irragionevoli

Un ulteriore approccio si basa su di una tecnica nata per la traduzione automatica, in cui

- la sequenza di partenza è codificata in una *sequenza* di stati anziché in uno solo
- un componente focalizza *(attenziona)* il decoder su quale elemento della codifica sia più *attinente* al simbolo di uscita



...ma ne riparliamo nella parte sul TTS, per ora limitiamoci al confronto

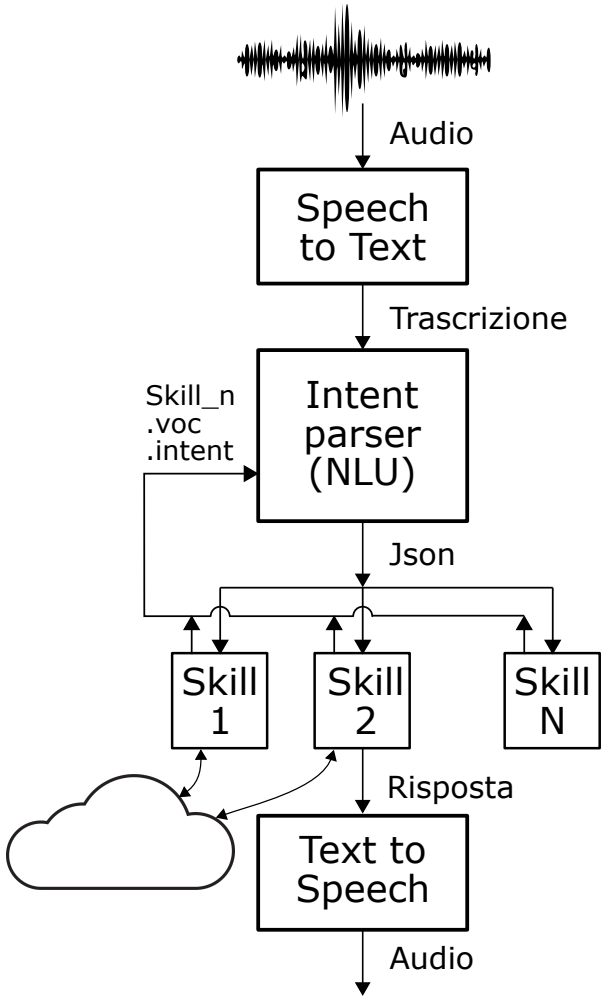
Model	Delay	Computation Complexity	Language Model Ability	Training Difficulty	Recognition Accuracy
CTC-based	●○○○○	●○○○○	×	●○○○○	●○○○○
RNN-Transducer	●●●○○	●●●●●	√	●●●●●	●●●●○
Attention-based	●●●●●	●●●○○	√	●●●○○	●●●●●

Ora parliamo di...

Il pipeline linguistico

La dove risiede l'intelligenza

- La trascrizione prodotta dallo STT è analizzata dal componente di Natural Language Understanding (NLU) che ne produce una descrizione codificata in formato JSON (esempio)
- L'oggetto JSON è inviato su di un message bus su cui sono in ascolto gli agenti (Skill) in grado di eseguire (o meno) il comando
- Ogni Skill istruisce il parser relativamente a quale intento si ritiene competente, a mezzo di *parole chiave* o *frasi di esempio*
- Uno Skill può accedere (o meno) a risorse di rete per poter elaborare una *risposta* da passare al TTS
- Mycroft adotta due diverse implementazioni di Intent parser, *Adapt* e *Padatious*



Adapt

- Idoneo a sistemi con risorse limitate e non connessi alla rete
- Individua un intento in base ad un insieme di parole chiave e di espressioni regolari, che trova in
  `/opt/mycroft/skills/`*`skillname`*`/(vocab|regex)/`*`lingua`*`/`*`insieme`*`.(voc|rx)`
  - Qui la guida su come descrivere un intento da parte di uno skill
- Adapt ha capacità contestuali, ovvero può mantenere *uno stato* associato alle precedenti interazioni, in modo da poter dare luogo a *conversazioni* che si estendono su di una serie di frasi
- Restituisce un oggetto JSON in cui compare
  - l'intento
  - un livello di confidenza relativo al match
  - una lista di parole etichettate *(entità)*, usate dagli Skill per espletare le funzioni relative all'intento

Padatious

- Permette di descrivere gli intenti mediante un insieme di frasi-tipo
- Le frasi di esempio sono usate per allenare una *rete neurale* implementata a mezzo della libreria FANN. Le frasi si trovano presso

  `/opt/mycroft/skills/`*`skillname`*`/vocab/`*`lingua`*`/`*`insieme`*`.(intent|type)`

  in cui *insieme* corrisponde al *metodo* dello skill che esegue il match
  - questi files sono caricati a seguito dell'installazione dei relativi skill
  - la NN li esamina all'avvio per *apprendere* la propria logica
- Le attivazioni (della NN) relative ai diversi intenti sono mutuamente indipendenti, permettendo di installare nuovi skill senza *apprendere* di nuovo i precedenti
- E' facile estrarre *le entità* e quindi darle in pasto agli skill. Ad esempio "Find the nearest gas station" $\rightarrow$ *{ "place":"gas station"}*

Ora parliamo di...

1. Scenario
   - Passato, presente e (?) futuro
   - Le applicazioni esistenti
   - Il caso Mycroft
   - Open Source, Cloud e Privacy
   - Lo stack vocale
     - Aspetti sistemistici
2. **Tecnologie**
   - La parola di risveglio
   - Riconoscimento del parlato
   - Comprensione del linguaggio naturale
   - Agenti attuatori (Skills)
   - Sintesi vocale
3. Riferimenti e materiale utile

Gli Skills esistenti

In pratica, le *app* di un assistente vocale

Sono elencati presso il Marketplace. Possiamo raggrupparli come

- *di sistema*: registratore, controllo volume, configurazione, suggerimento ed installazione di altri skill
- *quotidiane*: data e ora, podcast delle news, Mycroft dormiente, allarmi e timer, ricordami di, meteo
- *divertenti*: cantare, ridere, barzellette, abbaiare, farsi ripetere qualcosa, numeri casuali, lancio di dadi, indovinare un numero, la storia di Mycroft, insegnare a gestire richieste sconosciute, raccontare fiabe, risposte casuali binarie, dove si trova la stazione spaziale, un chatbot, un adventure game
- *informazione*: Wikipedia, Wolfram Apha, DuckDuckGo, arbitro di quale skill risponde, ricette cocktail, parchi, interroga TMDB su trame attori registi e date, dizionario, spelling, accadde oggi, scelta del WiFi
- *IoT*: interfacce per Wink, Home assistant, Mozilla, Lifx
- *produttività*: email, ricerca files, gestione liste di cose da fare, speedtest, collegamento ad un Bot di Telegram

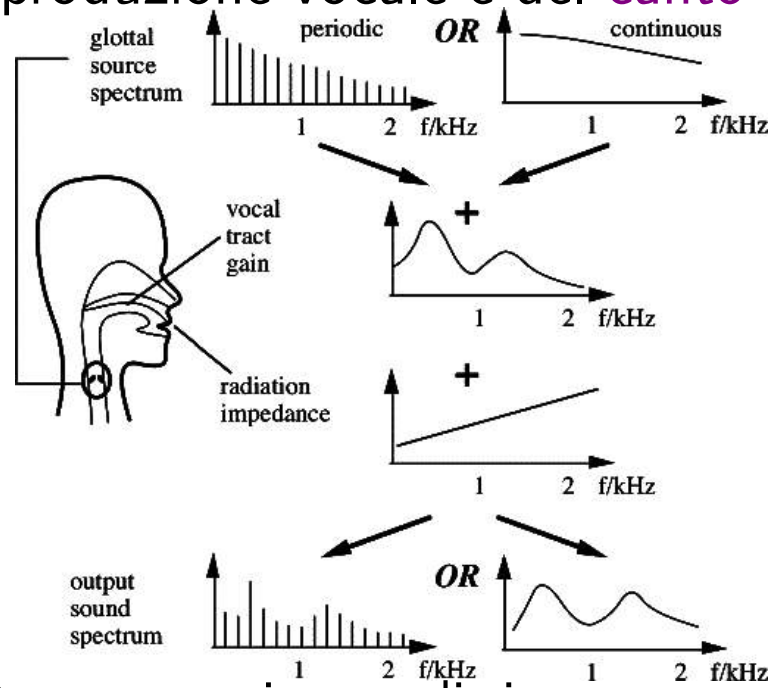
Ora parliamo di...

Tipologie di Sintesi Vocale

1. Parametrica (migliore intellegibilità)
   - codifica le conoscenze acustico-fonetiche nella forma di *parametri di controllo* con cui pilotare un *vocoder*
   - prevede trascrizione fonetica, intonazione (pitch intensità e durate), timbrica (formanti e transizioni), ed un modello di onda glottale
   - i parametri sono definiti tramite *regole* oppure appresi per via *statistica* mediante HMM
2. Concatenativa (migliore naturalezza)
   - vengono usati segmenti di segnale reale, distorti temporalmente e sovrapposti
   - sono adottate unità linguistiche semplici come i fonemi, o composite come difoni, sillabe o parole intere (se molto frequenti)
3. Neurale
   - un approccio segue quello della trasformazione sequenza-sequenza come per i traduttori automatici
   - un altro lavora direttamente su di un modello autoregressivo del segnale

Sintesi Parametrica

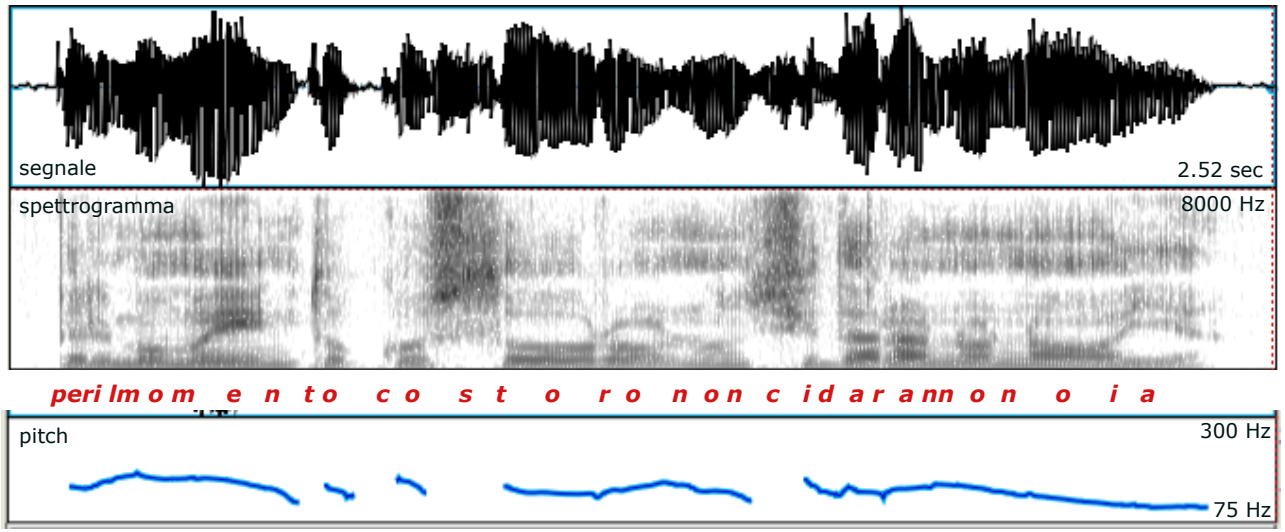
- Questo video illustra la *fisiologia* della produzione vocale e del canto
- Coinvolge diversi *meccanismi articolatori* ossia corde vocali, glottide, velo, mascella, dorso e punta delle lingua, labbra
- Fisici ed ingegneri hanno acquisito le competenze dei fonetisti (PRAAT) e definito un *modello di produzione*
- Ne è derivato uno schema a blocchi di sintetizzatore programmabile (Klatt
- pag. 24). Due approcci:
  - sintesi *per formanti*, di dubbia qualità ma con esigenza di risorse (calcolo e memoria) trascurabile (es. Espeak)
  - sintesi *articolatoria*, basata su modelli fisici (es. Gnuspeech)
- Prima della *sintesi* il testo deve essere
  - *normalizzato* (sigle e numeri), individuate le classi sintattiche, svolta l'analisi prosodica, svolta la trascrizione fonetica, assegnate durate a fonemi e pause



Lo spettrogramma

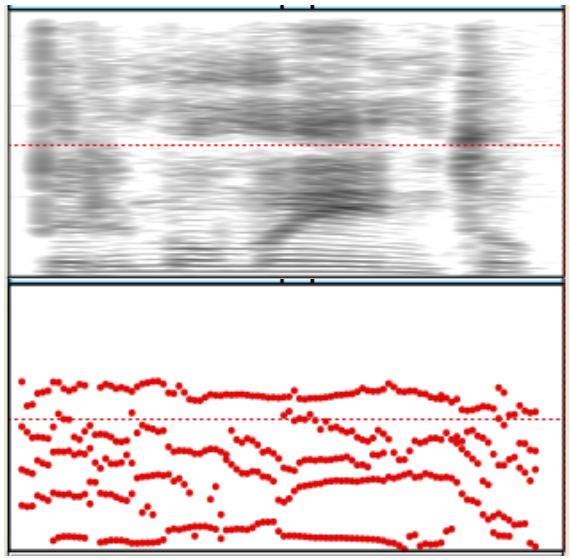
Lo strumento di lavoro del fonetista

Forma d'onda, spettro e pitch per la frase
per il momento costoro non ci daranno noia



Una finestra di analisi di 5 msec (meno di un periodo di pitch) produce linee *verticali* distanziate da tale periodo

Un finestra di 50 msec (a maggior risoluzione spettrale) applicata a *per il momento* mostra le striature *orizzontali* che rappresentano le *armoniche* del periodo di pitch

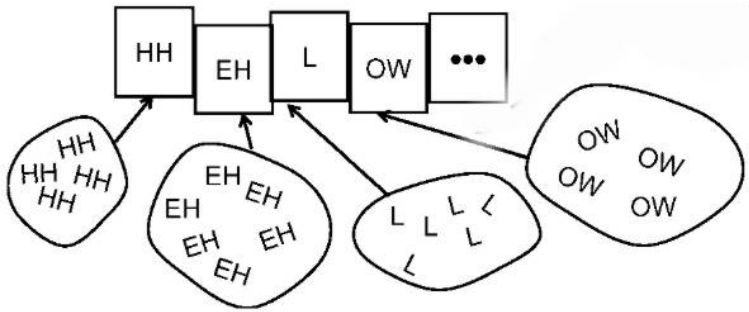


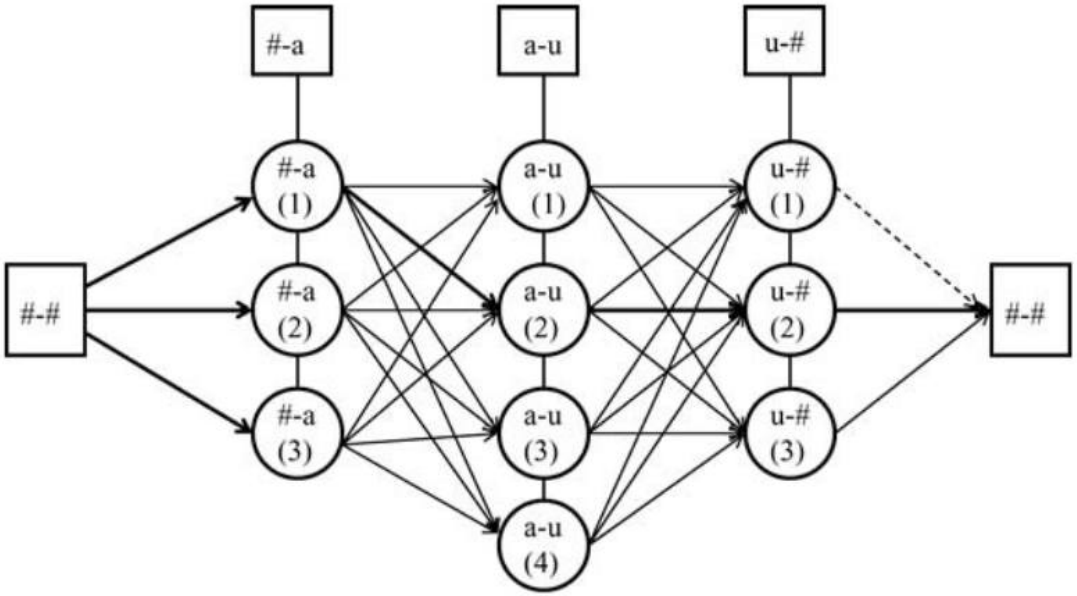
Le zone più scure corrispondono ai picchi spettrali delle *risonanze* del tratto vocale, denominate *formanti*, di cui le due inferiori già permettono di distinguere tra le vocali

Sintesi concatenativa

Quali sotto-unità utilizzare

- Richiede la stessa pre-elaborazione linguistica → durate, pitch, intensità
- Il risultato complessivo si ottiene come una sequenza di frammenti audio
  - presenti in una base di dati audio registrata da un attore
-
  - etichettati foneticamente e prosodicamente mediante le tecniche del riconoscimento
  - occorre *ricercare* quelli più *simili* al target prosodico





• ogni segmento ha un *costo* legato allo *scostamento* dei parametri prosodici rispetto al target

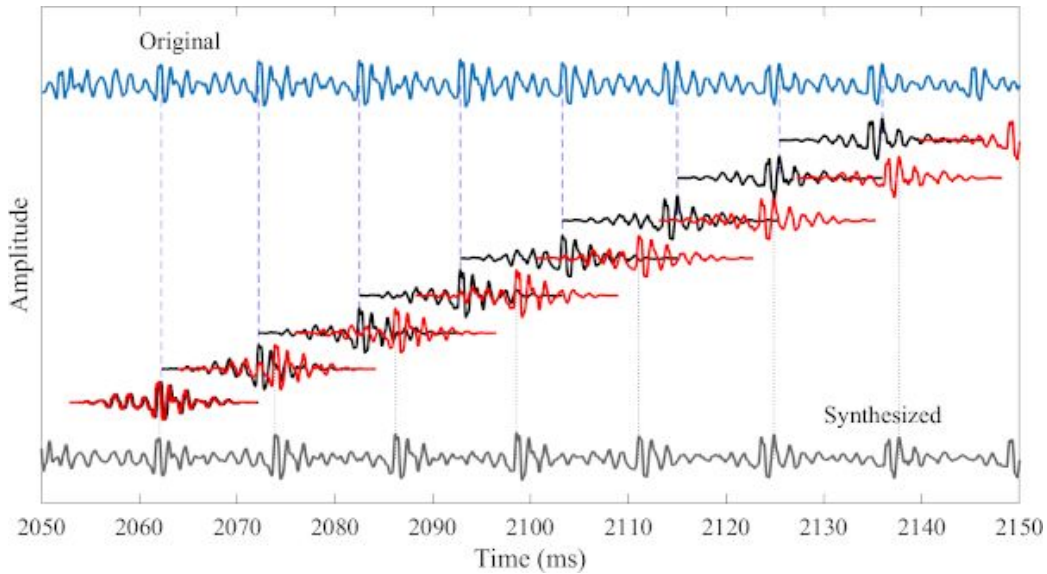
• ogni coppia ha un costo di *compatibilità contestuale*

• la migliore sequenza è individuata via *programmazione dinamica*

Concatenare, in che senso?

L'operazione di ricucitura

- Individuate le sotto-unità, sono necessarie *due* distorsioni temporali
  - modificare la durata del segmento in accordo al target prosodico
  - modificare con continuità il periodo di pitch
  - senza per questo modificare lo spettro (le formanti)
- La tecnica *Pitch Synchronous Overlap and Add* (PSOLA) suddivide il segnale in finestre apposite centrate sui massimi
- le finestre si ri-posizionano con il nuovo periodo di pitch
- sono aggiunti o tolti periodi per avere la durata richiesta, e si risommano le finestre



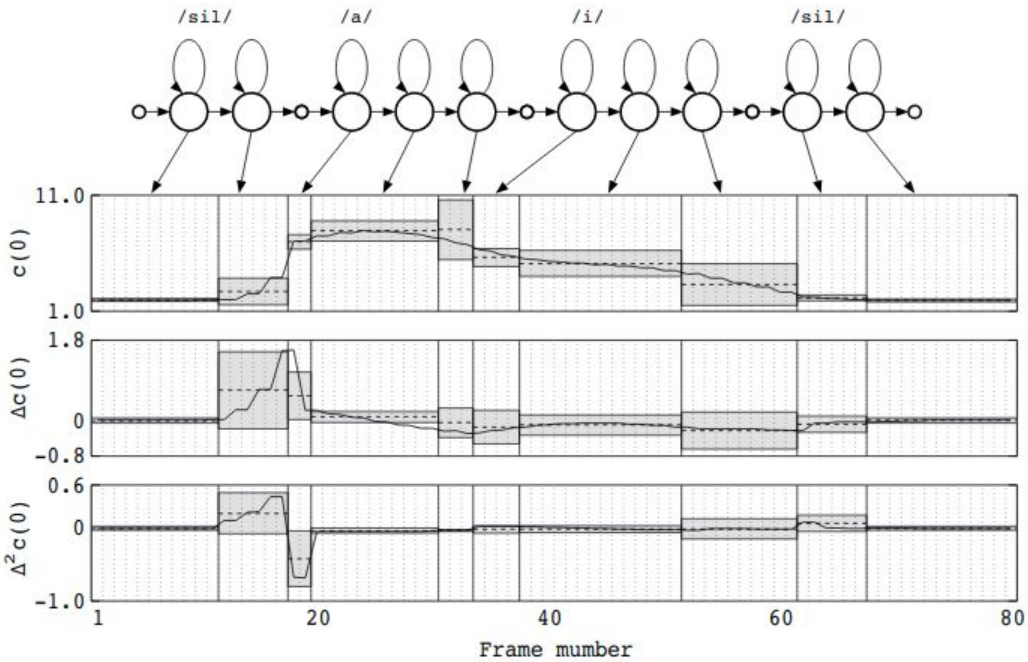
- MBROLA è un motore di sintesi per lo sviluppo collaborativo di nuove voci, che si appoggia a trascrittori fonetici esterni come Espeak e Festival
- MaryTTS implementa miglioramenti al metodo di ricerca delle unità

Sintesi parametrica statistica

Gli HMM, a volte ritornano

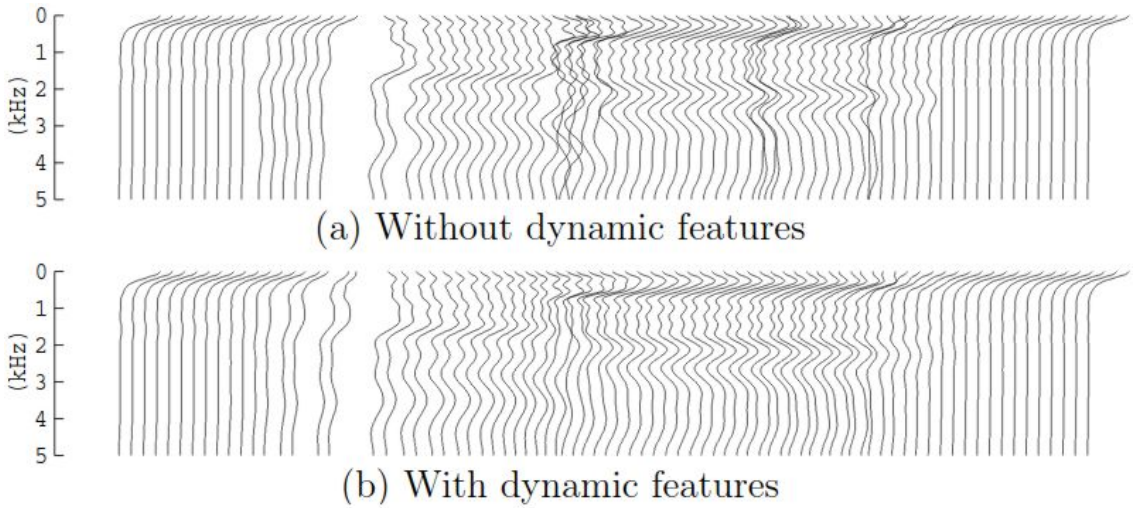
Problema: il valore delle durate e della prosodia da utilizzare nelle tecniche di sintesi discusse viene deciso in base a regole linguistiche rigide

- al contrario gli HMM delle unità fonetiche usati per il riconoscimento *apprendono* tali valori nella fase di training $\rightarrow$ *da usare per la sintesi!*
- si adottano HMM fonetici con tre stati, condizionati ai contesti destro e sinistro, allenati su sequenze $\boldsymbol{x}$ *arricchite* di derivata prima e seconda
- ogni stato $s_j \, \epsilon \, \mathrm{HMM}_{W_i}$ individua vettore medio $m_x(j)$ e varianza $\sigma_x^2(j)$ per le osservazioni $\mathbf{x}$, e viene stimata una d.d.p. del tempo di permanenza
- si costruisce la catena di HMM della frase di pronunciare, ottenendo la seq x più probabile ($\boldsymbol{c}$ in fig.)



Sintesi parametrica statistica - 2

Includendo nel vettore di osservazione (e dunque nel modello statistico) le prime due derivate si ottengono transizioni spettrali realistiche



(a) Without dynamic features

(b) With dynamic features

Il modello viene poi reso capace di modellare anche il contorno prosodico - ma la trattazione si complica e non approfondiamo

Da segnalare anche le capacità di trasformare una voce in un'altra

Demo: presso il sito dei primi proponenti (HTS), ma anche ad es. in quelli di Festvox e di MaryTTS

Speech Synthesis Markup Language (SSML)

Come spiegare alla sintesi in che modo parlare

Una *raccomandazione* del W3C per *guidare* la sintesi vocale negli aspetti di

pronuncia, volume, pitch, velocità, enfasi, pause,
genere, età, idioma, prosodia, espressività, emozioni,
audio di sottofondo

Ne troviamo un esempio presso IBM Watson.

Altre voci *commerciali* in italiano e non solo:
Responsive Voice, ReadSpeaker, Harpo, TTSMP3

La sintesi vocale di Mycroft

- Può usare sia servizi in *cloud* (di default GoogleTranslate via gTTS), sia in esecuzione *locale* come *Espeak*, *MaryTTS* e *Mozilla*
- La voce *nativa* di Mycroft è chiamata Mimic
  - basata su Flite, una versione *leggera* di Festival (github)
  - di tipo concatenativo
    - ★ difoni codificati come coefficienti LPC e residuo, con calcoli in virgola fissa (ma non solo)
    - ★ ufficialmente parla solo inglese... ma forse anche italiano (?)
  - una pagina del Blog riporta confronti tra Mimic ed altre voci, ma..
- dal 2018 si passa al *Deep Learning*, con nome in codice Mimic II
  - implementazione OpenSource di ***Tacotron*** di Google (github)
  - approccio *end-to-end* da grafema a segnale audio *(!!)*
    - ★ non serve più *riversare* competenze linguistiche nel codice
  - generazione di nuove voci a partire da una dataset testo + voce
    - ★ buoni risultati già con 16 ore di materiale audio
    - ★ è disponibile un tool di creazione del dataset
  - veloce perché produce *spettrogrammi* e non campioni

Tacotron

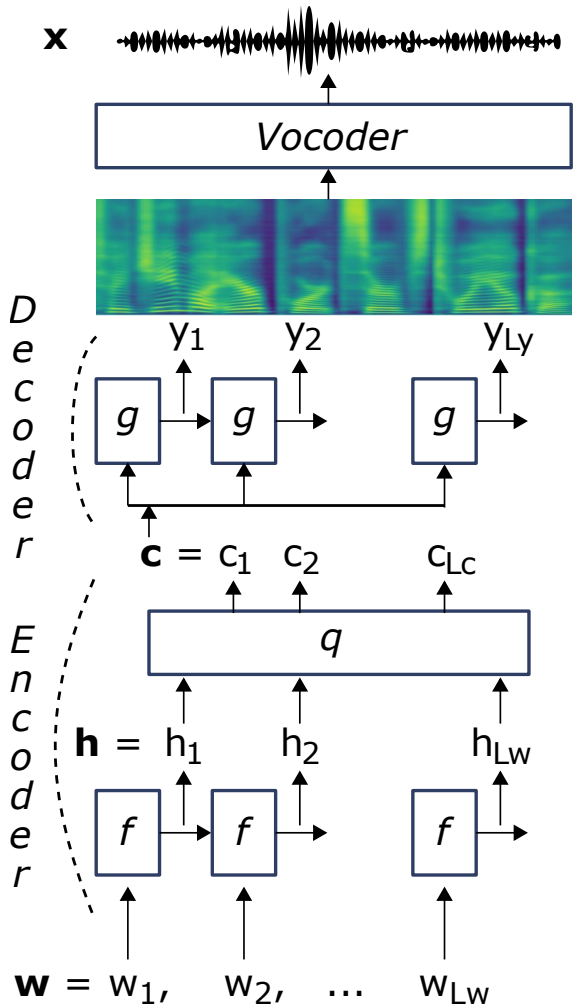
L'idea in generale

Approccio derivato da quello alla traduzione automatica (seq2seq)

- un *encoder* RNN produce una sequenza $\boldsymbol{h}$ di stati interni a partire dalla sequenza di caratteri $\boldsymbol{w}$ come $h_j = f\left(w_j, h_{j-1}\right)$, ed un *contesto* $\boldsymbol{c} = q\left(\boldsymbol{h}\right)$
- un *decoder* RNN con stato interno $\boldsymbol{s}$ *viene allenato* per predire la sequenza di spettri y_i in base a $p\left(y_i / y_1^{i-1}, \boldsymbol{c}\right) = g\left(y_{i-1}, s_i, \boldsymbol{c}\right)$
- dallo spettrogramma $\boldsymbol{y}$ si ottiene una forma d'onda $\boldsymbol{x}$ mediante un *vocoder* (Griffin-Lim)

La cosa si complica in quanto

- va fatto *l'embedding* dei caratteri $\boldsymbol{w}$ in uno spazio metrico
- occorre considerare il contesto destro e sinistro di ciascuna w_j
- $\boldsymbol{w}$ ed $\boldsymbol{h}$ hanno durata differente da $\boldsymbol{c}$, $\boldsymbol{y}$ e $\boldsymbol{x}$
- uno spettro y_i dipende da pochi caratteri w_j

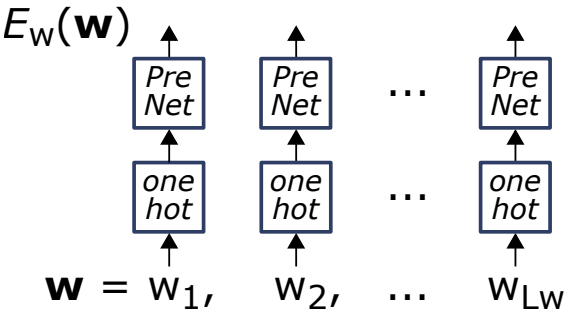


Tacotron (2)

Qualche dettaglio in più

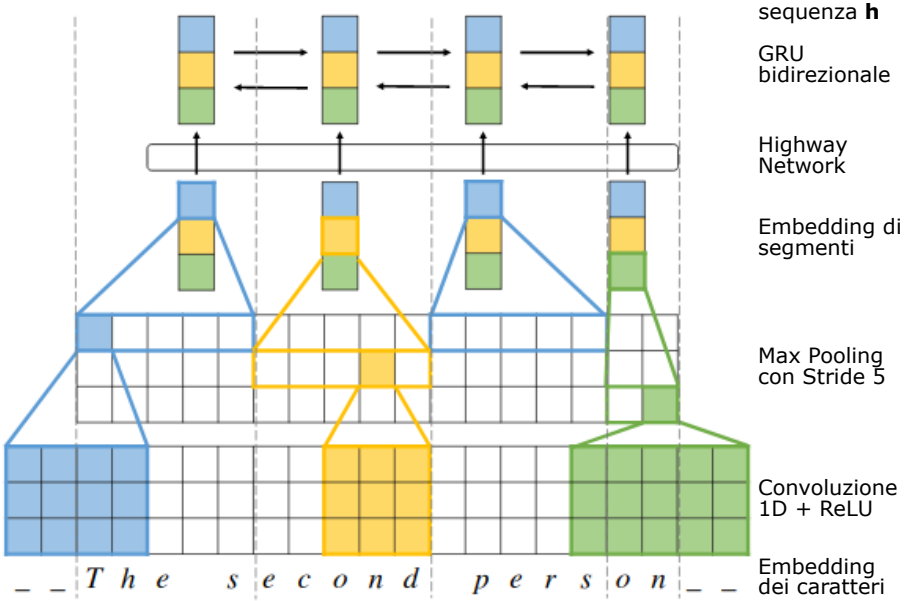
Embedding dei caratteri

- Ad ogni w_j corrisponde un vettore *one-hot*, elaborato da una *pre-net* di tipo FF che ne varia la dimensionalità, ottenendo la seq. $E_w(w_i)$



Encoder

- le attivazioni $\boldsymbol{h}$ sono prodotte da uno stadio **CBHG** ovvero *Convoluzione*, *Banco di filtri*, *Highway* e *GRU*
- filtri con diversa lunghezza riassumono informazioni *contestuali*
- *max-pool* riduce la durata
- la HighWay NN realizza una forma di *residual learning*
- la RNN dell'encoder è *bidirezionale* dando luogo ad $\boldsymbol{h} = \left[\overrightarrow{h}; \overleftarrow{h}\right]$
- sono adottate anche altre tecniche *residuali* e di *dropout*

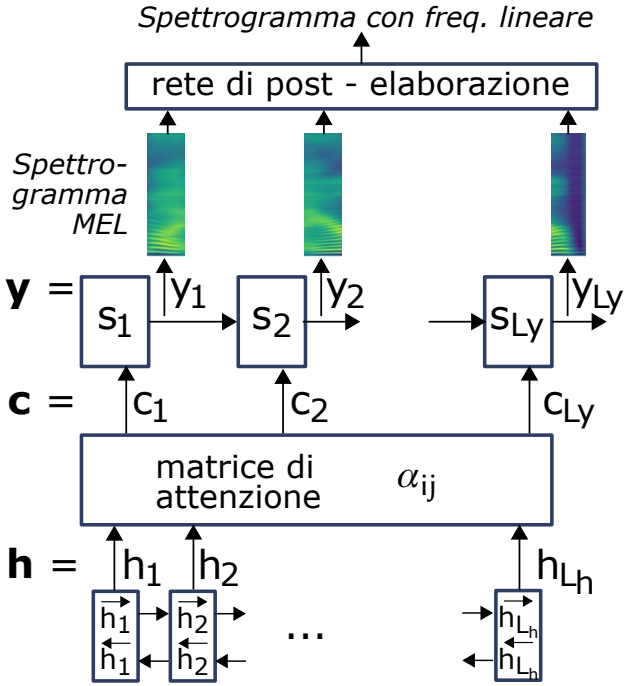


Tacotron (3)

Il decoder con attenzione ed il vocoder

Decoder

- Una *matrice di attenzione* valuta un diverso contesto $\boldsymbol{c}_i$ per ogni istante i di uscita dando *più peso* a certi stati $\boldsymbol{h}_j$ piuttosto che ad altri, ottenendo $\boldsymbol{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j$ in cui $\sum_j \alpha_{ij} = 1$
- la RNN di uscita aggiorna il proprio stato come $\boldsymbol{s}_i = f\left(\boldsymbol{y}_{i-1}, \boldsymbol{s}_{i-1}, \boldsymbol{c}_i\right)$ e valuta una predizione $p\left(\boldsymbol{y}_i / \boldsymbol{y}_{1:i-1}, \boldsymbol{c}\right) = g\left(\boldsymbol{y}_{i-1}, \boldsymbol{s}_i, \boldsymbol{c}_i\right)$
- lo spettrogramma MEL viene convertito in frequenze lineari da una ulteriore rete *CBHG* di post-elaborazione



Vocoder

- L'algoritmo *iterativo* di Griffin-Lim sintetizza una forma d'onda a partire da una sequenza di |STFT| *(modulo di short time fourier transform)*
- Questa la pagina di demo audio con diverse alternative
- Qui i progressi fatti dal 2017

Calcolo dell'attenzione

- La *matrice di attenzione* della slide precedente in realtà è solo *un modo di dire*: essendo le sequenze $\boldsymbol{h}$ e $\boldsymbol{c}$ di lunghezza *arbitraria*, non è infatti possibile definirne i coefficienti α_{ij} in fase di training
- Piuttosto, gli α_{ij} necessari al calcolo di $\boldsymbol{c}_i$ si ottengono da una ulteriore rete neurale, in base *all'attinenza* $a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right)$ tra lo stato $\boldsymbol{s}_{i-1}$ della RNN di uscita all'istante precedente, e tutti i valori $\boldsymbol{h}_j$, $j = 1, 2, \cdots, L_h$ della RNN dell'encoder, come
$$\alpha_{i,j} = \text{soft}\max_{\boldsymbol{h}}\left(a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right)\right) = \frac{\exp(a(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j))}{\sum_{j'} \exp\left(a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_{j'}\right)\right)}$$
- l'attinenza $a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right)$ è una misura di quanto $\boldsymbol{h}_j$ arrechi informazione *aggiuntiva* rispetto a quella codificata da $\boldsymbol{s}_{i-1}$ ai fini della predizione della *fetta di spettrogramma* corrente, e la NN che la valuta viene allenata assieme a tutto il resto
- Sono state proposte diverse alternative per il calcolo di $a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right)$, come ad esempio $a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right) = \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a \boldsymbol{s}_{i-1} + \boldsymbol{U}_a \boldsymbol{h}_j + \boldsymbol{b}\right)$ in cui $\boldsymbol{v}_a$, $\boldsymbol{W}_a$, $\boldsymbol{U}_a$ e $\boldsymbol{b}$ sono *appresi*

Emozioni ed alternative

Valutazione di Tacotron, ed il resto del mondo

- L'evoluzione della matrice di attenzione ricorda l'esito del DTW
- Perdiamo il controllo tipico dei metodi parametrici?
  - No, anzi. Si possono apprendere stili di pronuncia, sia espressivi che prosodici che emotivi. Si possono parlare lingue sconosciute.
- Quali altre tecniche si sono affermate?
  - Wavenet genera campioni vocali PCM legge μ con una NN convoluzionale *dilatata* che implementa un modello autoregressivo. Realizza un TTS condizionando la rete con una sequenza prosodico - fonetica, e dunque è *un vocoder*.
  - Char2Wav simile a Tacotron, usa SampleNN come vocoder
  - DeepVoice (Baidu) usa solo convoluzioni senza unità con memoria
  - FastSpeech (Microsoft) si basa su di un input fonetico, a cui aggiunge la prosodia; rivendica un codice particolarmente efficiente
  - anche Voiceloop (Facebook) rivendica un codice efficiente
  - citiamo infine ClariNet e WaveGlow

Conclusioni

L'idea da lungo tempo inseguita di poter interagire verbalmente con le macchine sembra potersi finalmente realizzare grazie ai progressi tecnologici che rendono possibile un approccio a riconoscimento e sintesi basato su reti neurali

Le competenze linguistico-fonetiche relative alla produzione del segnale vocale, su cui si basano le tecniche fin qui adottate come HMM e sintesi parametrica, appaiono poter essere acquisite in modo *autonomo* a partire dall'analisi automatizzata di grandi quantità di dati

L'approccio che si sta rivelando vincente è quello di trattare riconoscimento e sintesi del segnale vocale come un problema di trascrizione da un linguaggio ad un altro, mutuando i metodi sviluppati per affrontare problemi di traduzione automatica

Il controllo della sintesi vocale, almeno per come è sempre stato possibile fare con approcci parametrici, sembra poter essere ottenuto mediante un opportuno condizionamento degli ingressi alla rete

Riferimenti e materiale utile

I link forniti sono validi al momento della stesura

Riconoscimento vocale

- HMM-GMM - L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, 1989
- HMM-DNN - H. Bourlard, N Morgan, Connectionist Speech Recognition - A Hybrid Approach, 1994
  - G.Hinton et al, Deep Neural Networks for Acoustic Modeling in Speech Recognition, 2012
  - A.L. Maas et al, Building DNN Acoustic Models for Large Vocabulary Speech Recognition, 2012
- CTC-RNN - A.Graves et al, Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks, 2006
  - D.Amodei et al, Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, 2015
  - A.Hannun, Sequence Modeling with CTC, 2017

Riferimenti e materiale utile - 2

Riconoscimento Vocale (cont.)

- RNN-T - A.Graves, Sequence Transduction with Recurrent Neural Networks, 2012
  - A.Graves et al, Speech Recognition with Deep Recurrent Neural Networks, 2013
  - Y.He et al, Streaming End-to-end Speech Recognition For Mobile Devices, 2018 *(blog post)*
  - Y. Li et al, Improving RNN Transducer Modeling for End-to-End Speech Recognition, 2019
- *Attention Based* - J.Chorowski et al, Attention-Based Models for Speech Recognition, 2015
  - W.Chan et al, Listen, Attend and Spell, 2015 *(blog post)*
  - C.Olah et al, Attention and Augmented Recurrent Neural Networks, 2016
- *Wav2Letter (Facebook)* - https://github.com/facebookresearch/wav2letter
- *Confronto tra gli approcci End-to-End* - D.Wang et al, An Overview of End-to-End Automatic Speech Recognition, 2019

Riferimenti e materiale utile - 3

Sintesi vocale

- *Parametrica* - D.H. Klatt, L.C. Klatt, Analysis, synthesis and perception of voice quality variations among male and female talkers, 1990
- *Concatenativa* - A.J. Hunt, A.W. Black, Unit selection in concatenative speech synthesis system using a large speech database, 1996
  - E.Moulines, F.Charpentier, Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones, 1990
  - T.Dutoit, MBR-PSOLA: Text-to-Speech Synthesis Based on An MBE Re-Synthesis of the Segments Database, 1993
  - A.W. Black, K,A. Lenzo, Flite: a small fast run-time synthesis engine, 2001
  - S. Le Maguer, I. Steiner, The MaryTTS entry for the Blizzard Challenge 2016
- *Sintesi via HMM* - K.Tokuda, T.Masuko et al, HMM/DNN-based Speech Synthesis System (HTS), 1996
  - J.Yamagishi, An Introduction to HMM-Based Speech Synthesis, 2006
  - H.Zen, K.Tokuda, A.W.Black, Statistical Parametric Speech Synthesis, 2009
- *Analisi storica* - S.King, Measuring a decade of progress in Text-to-Speech, 2014

Riferimenti e materiale utile - 4

Sintesi vocale (cont.)

- *Tacotron* - Y.Wang et al, Tacotron: Toward End-to-End Speech Synthesis, 2017
  - D.W. Griffin, J.S. Lim, Signal Estimation from Modified Short-Time Fourier Transform, 1984
  - D. Bahdanau et al, Neural Machine Translation by Jointly Learning to Align and Translate, 2015
  - J.Lee et al, Fully Character-Level Neural Machine Translation without Explicit Segmentation, 2017
  - J.Shen et al, Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, 2018
- *End-to-End* - J.Sotelo et al, Char2wav: End-to-end speech synthesis, 2017
  - Y.Taigman et al, VoiceLoop: voice fitting and synthesis via a phonological loop, 2018
  - W.Ping et al, Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning, 2018
  - Y.Ren et al, FastSpeech: Fast, Robust and Controllable Text to Speech, 2019
  - W.Ping et al, ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech, 2019

Riferimenti e materiale utile - 5

Sintesi vocale (cont.)

- *Vocoder* - A.van den Oord et al, WaveNet: A generative model for raw audio, 2016
  - M.Morise et al, WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications, 2016
  - S.Mehri et al, SampleRNN: An Unconditional End-to-End Neural Audio Generation Model, 2017
  - N. Kalchbrenner et al, WaveRNN: Efficient Neural Audio Synthesis, 2018
  - Z. Jin et al, FFTnet: A Real-Time Speaker-Dependent Neural Vocoder, 2018
  - R.. Prenger et al, WaveGlow: A Flow-based Generative Network for Speech Synthesis, 2018
  - Z.Kons et al, High quality, lightweight and adaptable TTS using LPCNet, 2019 (vedi anche qui e qui)
  - N.Chen et al, WaveGrad: Estimating Gradients for Waveform Generation, 2020

Riferimenti e materiale utile - 6

Cultura generale nel settore della voce

- L.R.Rabiner, R.W.Shafer, Introduction to Digital Speech Processing, 2007

Avanzamenti più recenti

- Interspeech 2020 - 25 - 29 October 2020, Shanghai

Mycroft

- *Introduzione* - A secure and private open source alternative to Alexa
  - Get started with open source voice assistant software
- Skills - How to prepare to write your first Mycroft AI skill using Python
  - Use intent parsers for your open source home automation project
  - Bring your Mycroft AI voice assistant skill to life with Python