Principles of ASR

Gabriele Filosofi

Aprile, 2005

Abstract

In the first part of this document, we will explore the core components of modern ASR systems. In the second part we will give design guidelines and review available resources to support the development of a ASR-enabled Radio Control system for avionics.

Contents

1 ASR definition

An **utterance** is the vocalization of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, or even a sentence.

Automatic Speech Recognition (ASR) is technology that allows a computer to identify the utterances that a person speaks into a microphone or telephone. In other words ASR is the ability to convert a speech waveform into text (the so called **transcription**).

We use ASR to communicate and control technology when a keyboard would be inconvenient because: (a) Too big, (b) Hands busy, (c) Via phone, (d) In the dark, (e) Moving around.

The "holy grail" of ASR research is to allow a computer to recognize in real-time with 100% accuracy all utterances that are intelligibly and fluently spoken by any person, independent of vocabulary size, noise, speaker characteristics, accent, speaking speed or channel conditions. Despite several decades of research in this area, accuracy greater than 90% is only attained when the task is constrained in some way.

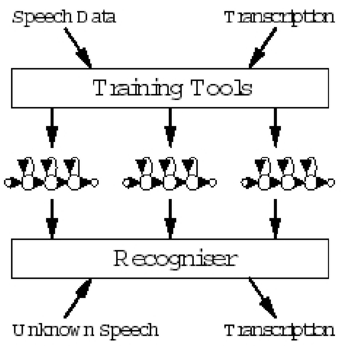


Figure 1: A ASR conceptual diagram

The waveform of a utterance varies a lot between different speakers (or even the same speaker). The adopted practice is to extract parameters (or features) from the speech waveform that are more consistent than the waveform. The role of the recognizer is to effect a mapping between sequences of speech parameter vectors and the original utterance. For each utterance, the recognizer has to find the most probable utterance among those legal that would have produced the parameter vectors observed.

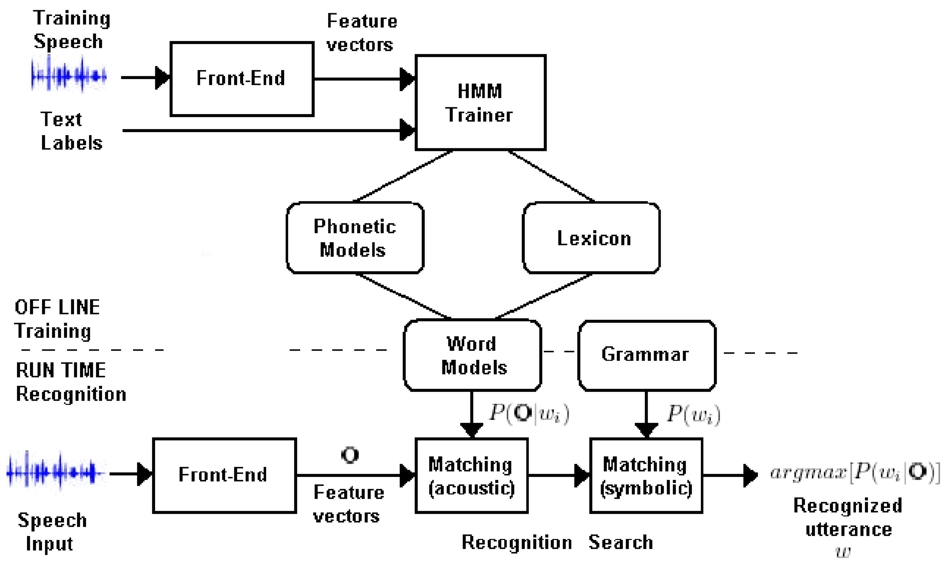


Figure 2: A ASR block diagram

2 System specifications

The desirable objectives for a recognizer can be broken into two categories, functional and technical.

2.1 Functional specifications

1. Speaker-Dependent/Independent recognition
2. Discontinuous/Continuous/Word-Spotting speech recognition
3. Word/Phoneme-based recognition
4. Mono/Di/Three-phone based recognition
5. Vocabulary size, flexibility and confuseability
6. Grammar and language constraints
7. Level of accuracy
8. Real time performance
9. Gender/Age-dependent/independent recognition

10. Language Dependent/Independent
11. Microphone Dependent/Independent
12. Speaker adaptation
13. Triggering recognition
14. N-Best output
15. Barge-in capability
16. Out-of-vocabulary detection

2.2 Technical specifications

1. Memory footprint (RAM/ROM words)
2. Computing requirements (MIPS)
3. fixed-point/floating-point arithmetic
4. Processing Delay (ms)
5. Power consumption

2.3 Speaker-Dependent/Independent recognition

A Speaker Dependent (**SD**) system is developed to operate for a single speaker. SD systems requires samples of speech from individual speakers. The process of providing these samples is called **training** or **enrollment**.

A Speaker Independent (**SI**) system is developed to operate for any speaker of a particular type (e.g. US English). These systems are the most difficult to develop, they are most expensive and accuracy is 3-5 times lower than SD systems. However, they are more flexible.

Intermediate between SD and SI systems there are also Multi-Speaker (**MS**) systems intended for use by a small group of people, and Speaker-Adaptive (**SA**) systems which tune themselves to any new speaker given a small amount of their speech as enrollment data.

2.4 Discontinuous/Continuous/Word-Spotting speech recognition

Three main types of ASR systems exist:

Isolated word They recognize single words beginning and ending with silence

Discontinuous speech They recognize full sentences in which words are artificially separated by silence

Word-Spotting speech They recognize a single word within a continuous stream of spoken speech or noise

Continuous speech They recognize naturally spoken sentences

Continuous ASR is more difficult, because word boundaries are unclear and their pronunciations are more corrupted by **coarticulation**, or the slurring of speech sounds, which for example causes a phrase like “could you” to sound like “could jou”. For example, while the sound /n/ of English normally has an alveolar place of articulation, in the word tenth it is pronounced with a dental place of articulation because the following sound, /t/, is dental.

Recognizers with continuous speech capabilities must utilize special methods to determine utterance boundaries. In a typical evaluation, the word error rates for isolated and continuous speech were 3% and 9%, respectively (Bahl et al 1981).

Word-Spotting is the ability to recognize a word within a continuous stream of spoken speech or noise. Assuming somebody wants to go to the next page or item, then he might say ”next, please” or ”next item, please” or just ”next”. Probably only the word ”next” belongs to the active vocabulary. Now, either one can instruct the users to use single word commands or one can deal in a gracious way with the irrelevant additional words. The first option is not a viable one as it has been found that in commercial systems 20-40% of all commands contain out of vocabulary words, whatever instructions are given to the naive users! Hence Word-Spotting is a must. There is no use for a 99.99% single word recognizer if it can not extract words from sentences. The cost of Word-Spotting is a slightly more complex recognition algorithm, but offers a more responsive, more natural interface for spoken commands than Isolated Word recognition, in which pauses are required before and after words.

2.5 Word/Phoneme-based recognition

Speech recognition systems use acoustic models that represent the variability of speech sounds across utterances, speakers, and environments. Acoustic models are created, trained with real voice samples, and finally tested and modified on modern servers using advanced tools.

Acoustic models vary in their granularity and context sensitivity. Three main acoustic models exist:

Word-based Used in applications for small vocabulary recognition. For example, for voice dialing, some keywords would be "call", "phone", "dial", etc.

Syllable-based

Phoneme-based Used when large and/or flexible vocabulary is needed, For example, when recognizing proper names from a private phonebook. Phoneme-based recognition allows for a relatively small and fixed amount of basic units. But it is less accurate.

Models with larger granularity (such as words) tend to have greater context sensitivity. Models with the greatest context sensitivity would give the best word recognition accuracy, but the larger the granularity, the poorer it will be trained, because fewer samples will be available for training it. For this reason, word and syllable models are rarely used in high performance systems.

2.6 Mono/Di/Three-phone-based recognition

Coarticulation is a variation in a phoneme due to the influence of neighboring phoneme. Context dependent phoneme-based models take into account coarticulation because a phone may be voiced differently depending on the other phones surrounding it. Phone-based recognition accuracy can be improved by using context-dependent parameters. Example: the phoneme /ee/ has different spectral features in the words "Wheel" and "Peat".

Three main acoustic phone-based models exist:

Monophone Many systems also use monophone models (sometimes simply called phone models), because of their relative simplicity

Biphone It models left or right context

Triphone Takes into consideration both the left and the right neightbouring phones. If two phones have the same identity, but different left or right contexts, they are considered different triphones. This is more indicative of a state-of-the-art modelling approach

If N is the total number of phonemes, the maximum number of triphones is N^3. Further, triphones can be grouped in classes depending on the left/right-context phone being a fricative or front vowel.

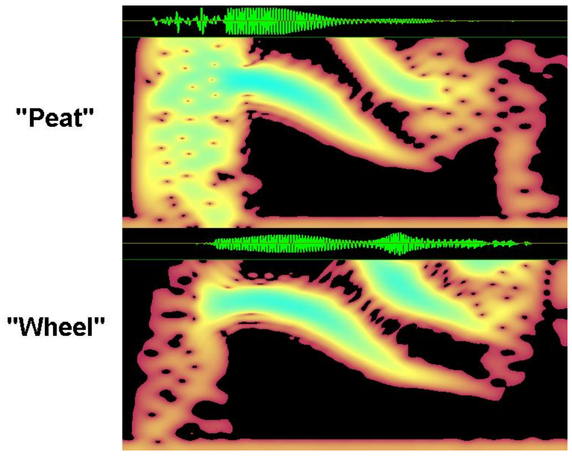


Figure 3: Different spectrograms of phoneme /ee/ in different word contexts.

Example: In a typical phoneme based implementation a simple monophone modelling of $N = 42$ phones was used, each having 3 states and 5 mixtures per state (see below). In experiments a bigram language model was used.

Example: In a typical phoneme based implementation a context dependent triphone modelling was performed, resulting in 1708 legal triphones. Each triphone is represented by 3 states, 5 mixtures per state. In experiments a trigram language model was used.

Cross-word refers to a triphone model that extends across word boundaries. It is important in continous ASR. **Word-internal** refers to a triphone model that remains within word boundaries.

2.7 Vocabulary size, flexibility and confuseability

A **vocabulary** (or **dictionary**) $\Omega = \{w_i\}$ is the list of utterances that can be recognized by a ASR system. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g."Wake Up"), while very large vocabularies can have a hundred thousand or more.

With a fixed vocabulary the user is constrained to a particular set of words, while a flexible vocabulary allows the addition of user-defined words simply recompiling a text file.

As a general rule, it is easy to discriminate among a small set of words, but error rates naturally increase as the vocabulary size grows. The 10 digits "zero" to "nine" can be recognized essentially perfectly, but vocabulary sizes of 200, 5000, or 100000 may have error rates of 3%, 7%, or 45%.

Even a small vocabulary can be hard to recognize if it contains confusable words. For example, the 26 letters of the English alphabet (treated as 26 "words") are very difficult to discriminate because they contain so many confusable words (most notoriously "B, C, D, E, G, P, T, V, Z"); an 8% error rate is considered good for this vocabulary. Let's you have 80-85% raw phoneme accuracy at 30db SNR. So if your grammar has a pair of word paths which are distinct in one phoneme only then you should expect one error in five, or so, when trying to recognize one of those word sequences. If a pair of paths are distinct in two phonemes, then you can get 1/25 errors, which is 96%, which isn't bad. But if you can make sure they're all distinct by three phonemes or more, then your recognition results should be 1 such error in 125, which is suddenly very very reliable (given 30db SNR). If you have a very simple small grammar including things like "turn it on" and "turn it off", then you cannot expect reliable results. So design your grammar using words that are different by multiple phonemes, and you should have good results.

The **active vocabulary** Ω_t is a set of words that can be recognized at any given moment t. Often the recognition of a specific utterance $w \in \Omega_1$ triggers the loading of a new active vocabulary Ω_2. This alleviates the task and reduce the search time. The complete set of words that can be recognized by a given ASR system is called **total vocabulary** $\Omega = \bigcup_k \Omega_k$.

2.8 Grammar and language constraints

While acoustic phone models enable the recognizer to decode phonemes that comprise words, **language models** (or **grammars**) specify the order in which a sequence of words (those listed in the dictionary) is likely to occur.

Even with a fixed vocabulary, performance will vary with the nature of constraints on the word sequences that are allowed during recognition. Some constraints may be task-dependent. Other constraints may be semantic (rejecting "The apple is angry"), or syntactic (rejecting "Red is apple the"). Language constraints are often represented by a grammar G.

Grammars are usually rated by their **perplexity**, a number that indicates the grammar's average branching factor (i.e., the average number of words that can follow any given word). The difficulty of a task is more reliably measured by its perplexity than by its vocabulary size.

N-Gram is a general term for a set of statistical constructs used to represent the probability of a word appearing alone (unigram) or in sequence with other words (bigram, trigram, etc.). N-grams are used to build better language models, which contribute to better recognition accuracy in dictation applications by favoring words that are more likely to occur in a particular context.

2.9 Level of accuracy

The ability of a recognizer can be examined by measuring its accuracy, or how well it recognizes utterances. This includes not only correctly identifying an utterance but also identifying if the spoken utterance is not in its vocabulary. Good ASR systems have an accuracy of 98% or more. The acceptable accuracy of a system really depends on the application. The typical achievable recognition rate as of 2005 for large-vocabulary SI systems is about 80-90% for a clear environment, but can be as low as 50% for scenarios like cellular phone with background noise.

Three types of error exist:

Deletion the type of error in which the recognizer's hypothesis does not contain a word in the reference transcription.

Insertion the type of error in which the recognizer produces a word (or symbol) hypothesis that does not correspond to any word in the reference transcription. Insertion errors are also common when noise is mistakenly recognized as speech.

Substitution the type of error in which a word in the reference transcription is replaced by an incorrect word in the recgonizer's hypothesis. Technically, the start and stop times of the hypothesis must overlap with the reference string for an error to be counted as a substitution.

Example: Correct: "Did mob mission area of the Copeland ever go to m4 in nineteen eighty one?"; Recognized: "Did mob mission area the copy land ever go to m4 in nineteen east one?". Insertion errors are also common when noise is mistakenly recognized as speech.

Word Error Rate is defined as

$$WER = 100\% \cdot \frac{del + subs + ins}{words - in - the - correct - sentence} \tag{2.9.1}$$

Other accuracy measurements are

- Word Recognition Rate (WRR = 100% - WER)
- Sequence (or Sentence) Error Rate (SER)
- Sequence Recognition Rate (SRR = 100% - SER)

Typical performance today:

- Continuous digits (0-9) over a noiseless channel: $SRR > 99\%$
- Speaker-Dependent and larger vocabulary: $WRR = 90 - 95\%$

Example: *Sentence Recognition Performance*
Total number of sentences 2272
Total sentence errors 133 (5.9%)
With substitutions 73 (3.2%)
Deletions 41 (1.8%)
Insertions 19 (0.8%)

Example: *Word Recognition Performance*
Total number of words 12785
Total word errors 148 (1.2%)
Substitutions 84 (0.7%)
Deletions 45 (0.4%)
Insertions 19 (0.1%)

Scoring refers to the process by which a recognition system's output is compared to reference transcriptions containing the "correct" answer. Errors are tabulated and presented in a format that can help a user understand the deficiencies of the system. NIST distributes a scoring package that is widely used within the community.

2.10 Out-of-vocabulary detection

This refers to the ability of the system to detect speech sounds that do not match any word in the active vocabulary. An application developer can take advantage of this feature to construct a more helpful speech interface. Rather than simply do nothing (reject an utterance) when a user tries, but fails, to say a valid command, the application can prompt the user to try again, or offer alternatives.

2.11 N-Best output

This refers to the ability of the system to suggest the closest N words recognized.

2.12 Triggering recognition

The trigger problem refers to the ability of the system to start the speech processing at the correct time. Three main techniques to trigger recognition exist:

Continuous Listening The speech detector is always active and continuously determines the start and end of speech

Push-to-Talk (PTT) A button push is required to initiate recognition. The speech detector determines the start and end of speech after the button push

Wake-Up Command the speech recognizer remains in a stand-by state, until it hears a specific "trigger" word (or keyword), which then activates normal recognition. This feature eliminates the need for a PTT button and is ideal for applications that require hands-free operation

In a typical Wake-Up Command implementation, two grammars exist: G_1 has a garbage model and silence in a loop, plus the keyword. G_2 has the rest of the command/control grammar along with optional silence, in a loop. The system starts recognizing with G_1 enabled, gets a traceback of the best results so far every $100ms$ (10 frames or observations) or so, and checks to see if the keyword has been recognized. Once the keyword is recognized, then immediately G_2 is enabled and G_1 is disabled. Finally, after a second threshold duration of silence has been recognized, then G_1 is switched back in, since the user isn't saying anything. This approach enables the system to continuously recognize without any PTT requirement.

2.13 Pruning capability

A Pruning level trades off recognition response time for accuracy.

2.14 Barge-in capability

This refers to the ability of the system to return intermediate results in real-time.

2.15 Processing delay

Processing Delay is measured in CRT units (*Computational Real Time*). 1 CRT equals 1 times the duration of the speech input.

3 Speech signal processing

3.1 Speech signal characteristics

Speech consists of vibrations produced in the vocal tract. The audible frequency range in human beings extends from $20Hz$ to $20kHz$. Speech sounds contain energies at all frequencies in the audible range, although it is thought that most phonetic information is concentrated below $8000Hz$.

The human vocal apparatus has mechanical limitations that prevent rapid changes to sound generated by the vocal tract. As a result, speech signals may be considered stationary, i.e., their spectral characteristics remain relatively unchanged for several milliseconds at a time.

3.2 Physiology of speech production

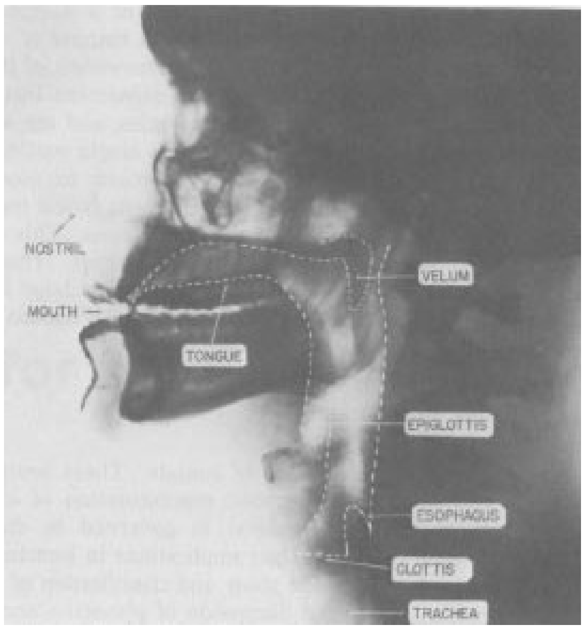


Figure 4: Analtomy of vocal apparatus in a sagittal view

The **tongue** is used to alter the vocal tract shape; the **velum** closes off nose cavity for all sounds except /m/, /n/ and /ng/; the **epiglottis** closes off larynx during eating; **vocal folds** vibrate during voiced sounds. Sources of Sound Energy are represented by:

Turbulence Air moving quickly through a small hole (e.g./s/ in ”size”)

Explosion Pressure built up behind a blockage is suddenly released (e.g. /p/ in ”pop”)

Vocal Fold Vibration Like the neck of a balloon (e.g. /a/ in ”hard”)

The frequency of vibration is determined by tension in vocal folds and pressure from lungs. For normal breathing and voiceless sounds (e.g. /s/) the vocal folds are held wide open and don’t vibrate.

3.3 Speech sound categories

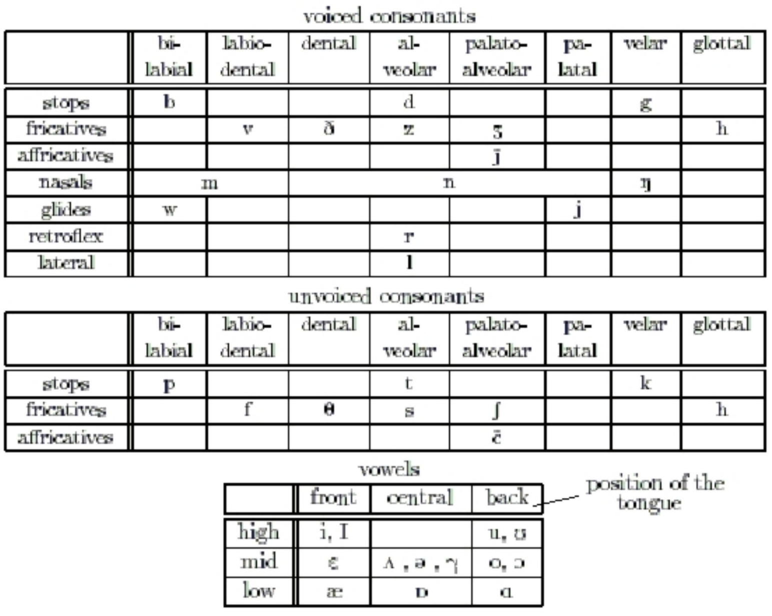
Speech sounds fall into four main categories:

Voiced Speech sounds where the vocal folds vibrate

Vowels Voiced sounds involving no blockage of the vocal tract and no turbulence

Consonants Non-vowels

Plosives Consonants invovling an explosion



voiced consonants

	bi-labial	labio-dental	dental	al-veolar	palato-alveolar	pa-latal	velar	glottal
stops	b			d			g	
fricatives		v	ð	z	ʒ			h
affricatives					ǰ			
nasals	m		n				ŋ	
glides	w					j		
retroflex				r				
lateral				l				

unvoiced consonants

	bi-labial	labio-dental	dental	al-veolar	palato-alveolar	pa-latal	velar	glottal
stops	p			t			k	
fricatives		f	θ	s	ʃ			h
affricatives					č			

vowels

	front	central	back
high	i, I		u, ʊ
mid	ɛ	ʌ , ə , ɤ	o, ɔ
low	æ	ɒ	ɑ

position of the tongue

Figure 5: Speech sounds in english

3.4 Vocal tract filter

The sound spectrum is modified by the shape of the vocal tract. This is determined by movements of the jaw, tongue and lips. The resonant frequencies of the vocal tract cause peaks in the spectrum called **formants**. The first two formant frequencies are roughly determined by the distances from the tongue hump to the larynx and to the lips respectively.

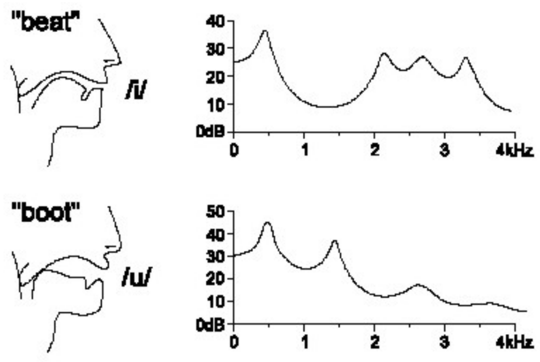


Figure 6:

3.5 Speech signal representation

Speech can be represented by speech waveforms, i.e. recorded amplitude as a function of time.

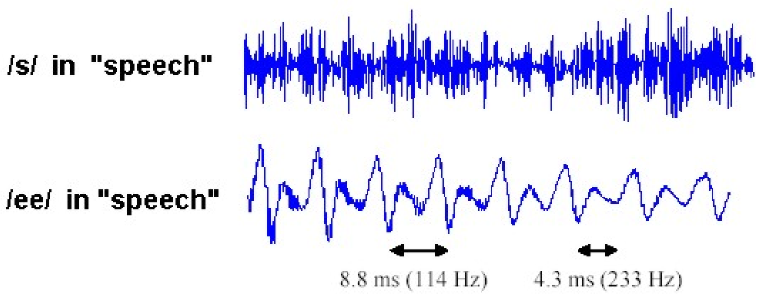


Figure 7: Speech waveforms

It is not possible to read the phonemes in a waveform, but if we analyze the waveform into its frequency components, we obtain a **spectrogram** which can be deciphered. For a given spectrogram S, the strength of a given frequency component f

at a given time t in the speech signal is represented by the darkness or color of the corresponding point $S(t, f)$.

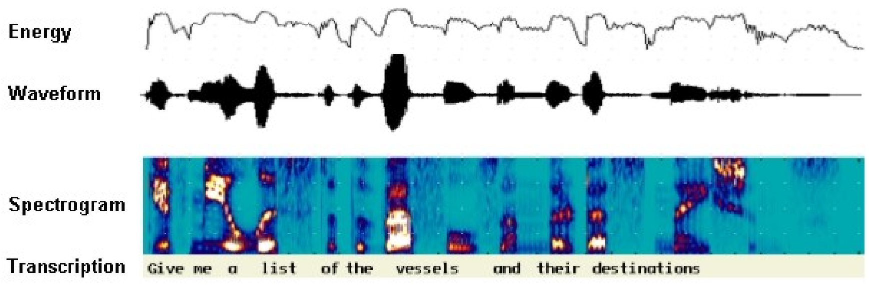


Figure 8: Speech representations

In a BW spectrogram, darker areas show high intensity. Voiced segments are much louder than unvoiced. Horizontal dark bands are the formant peaks.

3.6 Linear acoustic models of speech production

A detailed acoustic theory must consider the effects of the following:

- Time variation of the vocal tract shape
- Losses due to heat conduction and viscous friction at the vocal tract walls
- Softness of the vocal tract walls
- Radiation of sound at the lips
- Nasal coupling
- Excitation of sound in the vocal tract

Let us consider the simplest model, a lossless tube. Its length is about $L = 17.5cm$.

In a lossless tube, sound waves with $\lambda > \frac{L}{2}$, or frequency $f < \frac{c}{L/2} = \frac{35000cm/s}{8.75cm} = 4000Hz$, can be considered 1-dimensional and propagating according to the following equations

$$\rho\frac{\partial}{\partial t}(\frac{u}{A}) + \nabla p = 0 \tag{3.6.1}$$

$$\frac{1}{\rho c^2}\frac{\partial p}{\partial t} + \frac{\partial A}{\partial t} + \nabla \cdot u = 0 \tag{3.6.2}$$

$$\tag{3.6.3}$$

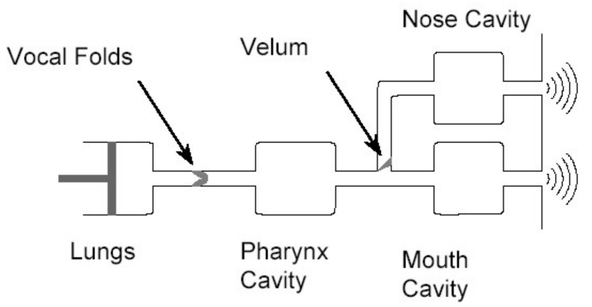


Figure 9:

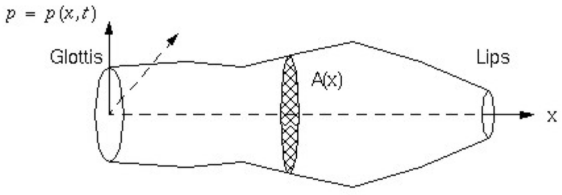


Figure 10: A linear lossless tube

where $c \simeq 20\sqrt{T(K)} = 350m/s$ is the velocity of sound in air; $p(x,t)$ is the variation of the sound pressure; $u(x,t)$ is the air flow in cc/s; $\rho = 1.2mg/cc$ is the density of air in the tube; $A(x,t)$ is the area function (about $17.5cm$ long in adults)

For a uniform tube, $A(x,t) = A$ and the solution is the superposition of two waves: u^+ in the forward direction and u^- in the reverse direction

$$u(x,t) = u^+(t - \tfrac{x}{c}) - u^-(t + \tfrac{x}{c}) \tag{3.6.4}$$

$$p(x,t) = \frac{\rho c}{A}[u^+(t - \tfrac{x}{c}) + u^-(t + \tfrac{x}{c})] \tag{3.6.5}$$

This is analogous to a lossless transmission line with longitudinal inductance $\frac{\rho}{A}$ and shunt capacitance $\frac{A}{\rho c^2}$.

Steady state solutions for a sinusoidal excitation are

$$u(x,t) = \frac{cos(\omega(L-x)/c)}{cos(\omega L/c)} U_G(\omega) e^{j\omega t} \tag{3.6.6}$$

$$p(x,t) = jZ_0 \frac{sin(\omega(L-x)/c)}{cos(\omega L/c)} U_G(\omega) e^{j\omega t} \tag{3.6.7}$$

where $Z_0 = \frac{\rho c}{A}$ is the characteristic impedance. The transfer function is given by

$$H(\omega) \equiv \frac{U(L,\omega)}{U(0,\omega)} = \frac{1}{cos(\omega L/c)} \tag{3.6.8}$$

This function has poles located at

$$\omega = \frac{(2n+1)\pi c}{2L} \tag{3.6.9}$$

Note that these correspond to the frequencies at which the tube becomes a quarter wavelength

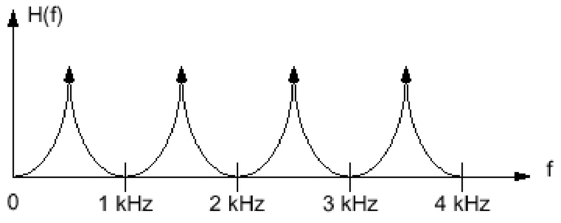


Figure 11: Transfer function for a linear lossless uniform tube

The effects of yielding walls can be predicted using perturbation analysis

$$A(x,t) = A_0(x,t) + \delta A(x,t) \tag{3.6.10}$$

$$m_w \frac{\partial \delta^2 A}{\partial t^2} + b_w \frac{\partial \delta A}{\partial t} + k_w \delta A = p(x,t) \tag{3.6.11}$$

Solving for the new transfer function we obtain something like this

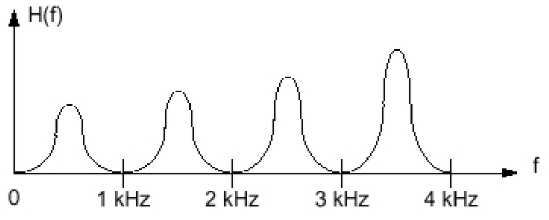


Figure 12: Transfer function for a linear uniform tube with yielding walls

Friction and thermal losses could be taken into account. Radiation of the sound waves from the lips can be modelled with a complex load applied at $x = L$.

$$Z_L(\omega) = \frac{j\omega L_r R_r}{R_r + j\omega L_r} \tag{3.6.12}$$

where $R_r = 128/9\pi^2$ and $L_r = 8a/3\pi c$ (a is the radius of the opening). The air flow at the lips becomes

$$\Phi_L(\omega) = Z_L(\omega)U(L,\omega) \tag{3.6.13}$$

Z_L acts as a high pass filter. Lip radiation introduces a zero in the spectrum at DC and a broadening of the bandwidth at higher frequencies.

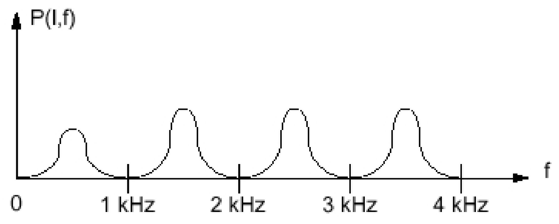


Figure 13: Transfer function for a linear uniform tube with yielding walls and lip radiation effects

We also must worry about the nasal cavity, especially for labial sounds, for which the mouth is closed during sound production. this is equivalent of placing a transmission line in parallel with the oral cavity. The net effect is to produce a zero in the spectrum at about $1kHz$. As a result, nasal sounds (such as /m/ and /n/ in US English) have very little energies at high frequencies.

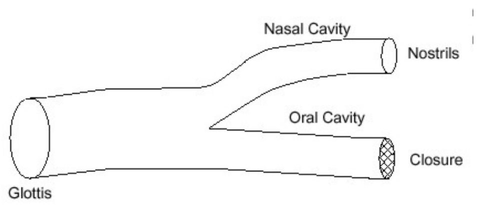


Figure 14: Nasal coupling in labial sounds

In the more realistic case of a nonuniform tube model we can introduce a piecewise linear approximation for the vocal tract (N-tube model).

It is a common practice to make all segments equal in length to the distance sound travels in half a sample period, $l = \frac{c}{2f_s}$. The number of segments needed is $N = \frac{L}{l} = 0.001 f_s$.

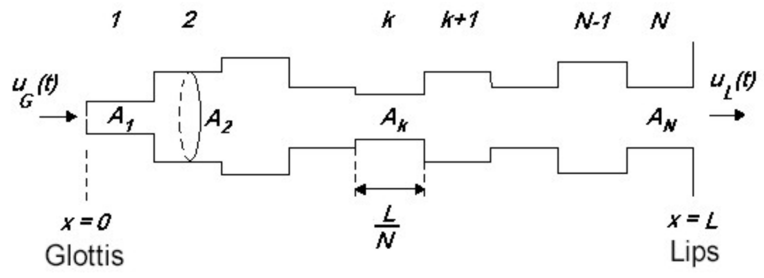


Figure 15: N-tube lossless model

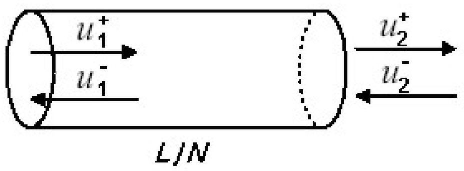


Figure 16: A single section of the N-tube lossless model

Consider one section of the N-tube lossless model.
Flow continuity for forward and backward flows along the lossless segment gives

$$u_1^+(t) = u_2^+(t+\tau) \tag{3.6.14}$$
$$u_1^-(t) = u_2^-(t-\tau) \tag{3.6.15}$$

where $\tau \equiv l/c = \frac{L}{Nc}$ is the time delay along the segment. If we take z-transforms, this corresponds to multiplying by $z^{-\frac{1}{2}}$

$$U_1^+ = U_2^+ z^{\frac{1}{2}} \tag{3.6.16}$$
$$U_1^- = U_2^+ z^{-\frac{1}{2}} \tag{3.6.17}$$

$$\begin{pmatrix} U_1^+ \\ U_1^- \end{pmatrix} = \begin{pmatrix} z^{\frac{1}{2}} & 0 \\ 0 & z^{-\frac{1}{2}} \end{pmatrix} \begin{pmatrix} U_2^+ \\ U_2^- \end{pmatrix} \tag{3.6.18}$$

At any section of the tube model the total volume flow is $u^+ - u^-$, while the total acoustic pressure is $p = \frac{\rho c}{A}(u^+ + u^-)$. Flow continuity and pressure continuity at tube junctions result in

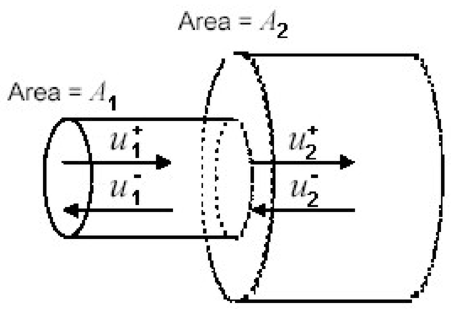


Figure 17: A junction between two segments

$$U_1^+ - U_1^- = U_2^+ - U_2^- \tag{3.6.19}$$

$$\frac{\rho c}{A_1}(U_1^+ + U_1^-) = \frac{\rho c}{A_2}(U_2^+ + U_2^-) \tag{3.6.20}$$

We can define the **reflection coefficient**

$$r = \frac{A_2 - A_1}{A_2 + A_1} \tag{3.6.21}$$

and rewrite the equations above

$$U_1^+ = \frac{1}{1+r}U_2^+ - \frac{r}{1+r}U_2^- \tag{3.6.22}$$

$$U_1^- = -\frac{r}{1+r}U_2^+ + \frac{1}{1+r}U_2^- \tag{3.6.23}$$

$$\tag{3.6.24}$$

Reflection coefficients always lie in the range $-1 \leq r \leq 1$.

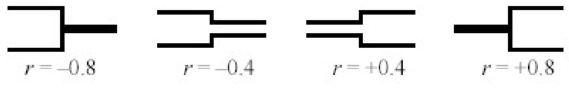


Figure 18: Reflection coefficients for different tube junctions

Taking into account both continuity equations at tube junction and delay effect along the segment we obtain

$$U_1^+ = \frac{z^{\frac{1}{2}}}{1+r}U_2^+ - \frac{rz^{-\frac{1}{2}}}{1+r}U_2^- \tag{3.6.25}$$

$$U_1^- = -\frac{rz^{\frac{1}{2}}}{1+r}U_2^+ + \frac{z^{-\frac{1}{2}}}{1+r}U_2^- \tag{3.6.26}$$

$$\tag{3.6.27}$$

$$\begin{pmatrix} U_1^+ \\ U_1^- \end{pmatrix} = \frac{z^{\frac{1}{2}}}{1+r}\begin{pmatrix} 1 & -r \\ -r & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & z^{-1} \end{pmatrix}\begin{pmatrix} U_2^+ \\ U_2^- \end{pmatrix} \tag{3.6.28}$$

The combined transfer function of the N-tube model is obtained applying this transformation iteratively, beginning from the glottis ($k = 0$) and ending to the lips ($k = N$)

$$\begin{pmatrix} U_0^+ \\ U_0^- \end{pmatrix} = \frac{z^{\frac{N}{2}}}{\prod_{k=0}^{N}(1+r_k)}\prod_{k=0}^{N-1}\begin{pmatrix} 1 & -r_k z^{-1} \\ -r_k & z^{-1} \end{pmatrix}\begin{pmatrix} 1 \\ -r_N \end{pmatrix}U_N^+ \tag{3.6.29}$$

We can assume air flow $U_0^- \equiv 0$ (i.e. $r_0 = 0$): it gets absorbed in the lungs. Also, we can assume $U_N^- \equiv 0$ (i.e. $r_N = 0$): no sound reflected back into mouth.

$$U_k^+ = \frac{z^{\frac{1}{2}}}{1+r}U_{k+1}^+ - \frac{rz^{-\frac{1}{2}}}{1+r}U_{k+1}^- \tag{3.6.30}$$

$$U_k^- = -\frac{rz^{\frac{1}{2}}}{1+r}U_{k+1}^+ + \frac{z^{-\frac{1}{2}}}{1+r}U_{k+1}^- \tag{3.6.31}$$

$$0 \leq k \leq N \tag{3.6.32}$$

$$U_0^+ \equiv U_G(z); \quad U_0^- \simeq 0 \tag{3.6.33}$$

$$U_N^+ \equiv U_L(z); \quad U_N^- \simeq 0 \tag{3.6.34}$$

Finally, the transfer function is given by

$$H(z) \equiv \frac{U_L(z)}{U_G(z)} = \frac{0.5(1+r_G)\prod_{k=1}^{N}(1+r_k)z^{-\frac{N}{2}}}{D(z)} \tag{3.6.35}$$

where

$$D(z) = 1 - \sum_{k=1}^{N} a_k z^{-k} \tag{3.6.36}$$

The denominator represents a N^{th} order all-pole filter.

$$\frac{U_L(z)}{U_G(z)} = \frac{Gz^{-\frac{N}{2}}}{1 - a_1 z^{-1} - a_2 z^{-2} \cdots - a_N z^{-N}} \tag{3.6.37}$$

where G is a gain term and $z^{-\frac{N}{2}}$ is the acoustic time delay along the vocal tract.

3.7 Speech Acquisition

Recording a digital sample is done by converting the analog signal from the microphone to an digital signal through the A/D converter in the sound card. Sound cards with the 'cleanest' ADC are recommended, but most often the clarity of the digital sample is more dependent on the microphone quality and even more dependent on the environmental noise. There are three important factors during this process:

sample rate or how often to record the voltage values.

bits per sample or how accurate the value is recorded.

number of channels (mono or stereo), but for most ASR applications mono is sufficient.

The choice of an appropriate sampling setup depends very much on the speech recognition task and the amount of computing power available.

Because speech is relatively low bandwidth (mostly between 100Hz-4kHz), $f_s = 8kHz$ is sufficient for most basic ASR. But, some people prefer $f_s = 16kHz$ because it provides more accurate high frequency information. If you have the processing power, use $11kHz$ or $16kHz$. Similar to sample rate, if you have enough processing power and memory, go with 16 bits per sample. The encoding format used should be linear signed or unsigned. U-Law/A-Law or some other compression scheme will cost you in computing power and not gain you much.

3.8 Microphone

A quality microphone is key when utilizing ASR.

Basically you need to have a 30dB SNR for optimum accuracy. In most cases, a desktop microphone just won't do the job. They tend to pick up more ambient noise that gives ASR programs a hard time. The best choice, and by far the most common is the headset style. It allows the ambient noise to be minimized, while allowing you to have the microphone at the tip of your tongue all the time. Headsets are available without earphones and with earphones (mono or stereo).

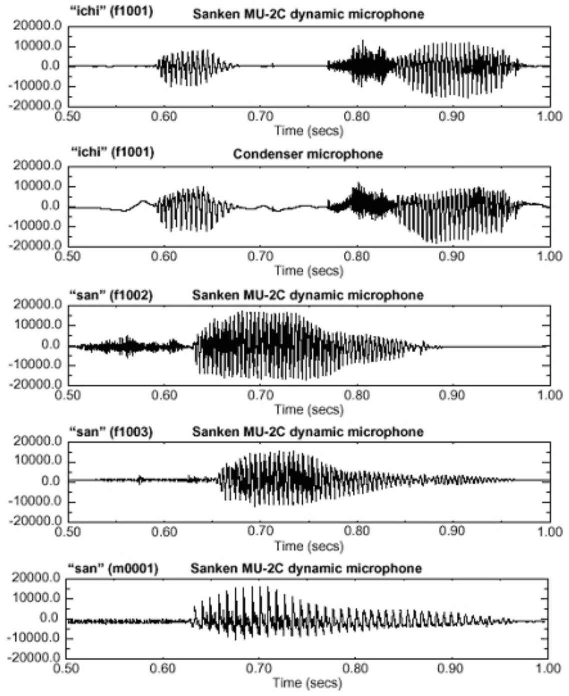


Figure 19: Different microphones elicite different responses

Bridging refers to maximizing the output voltage by increasing the load impedance and/or decreasing the microphone impedance. Low impedance microphones have an impedance of $100 - 300\Omega$ (more expensive); high impedance microphones have an impedance of $600 - 1000\Omega$ (less expensive).

A bidirectional microphone is a noise-cancelling microphone (such as the Sennheiser HD 414). Sound pressure never arrives at the front and the back of the microphone at the same time. However, noise, which arrives from the side, does, and hence is cancelled.

3.9 Finding Speech boundaries

End-point detection algorithms identify sections in an incoming audio signal that contain speech. Accurate end-pointing is a non-trivial task, however, reasonable behavior can be obtained for inputs which contain only speech surrounded by silence (no other noises). Typical algorithms look at the energy or amplitude of the incoming signal and at the rate of "zero-crossings" (where the audio signal changes from positive to negative or visa versa). When the energy and zero-crossings are at certain levels, it is reasonable to guess that there is speech. More detailed descriptions are provided in

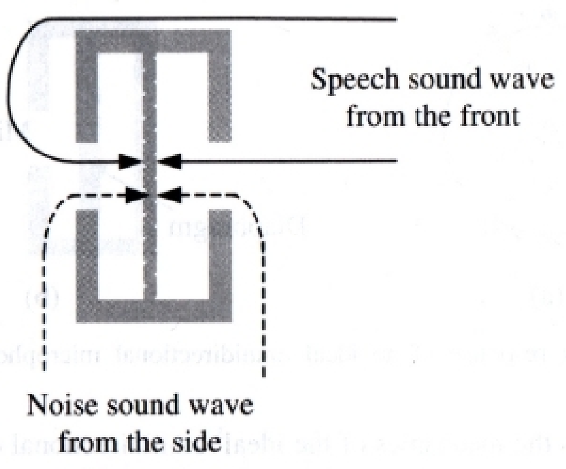

Speech sound wave from the front
Noise sound wave from the side

Figure 20: A bidirectional microphone

the papers cited below and in the documentation for the following software. End-point detection software is available from:
ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/tools/ep.1.0.tar.gz

3.10 Speech parametrization

Speech recognition is not performed over speech waveforms, instead it is performed over a more compact and meaningful representation of the speech signal. Speech parametrization (or feature extraction) removes redundancies, reduces storage and computational complexity, integrates the human speech production and hearing model in the recognition, improving recognition performances. How is Parameterization Done? The most popular approaches are

1. Mel-scaled cepstrum coefficients (FFT-based)
2. Mel-scaled cepstrum coefficients (LPC-based)
3. Delta (differential) cepstrum coefficients

The most popular FE are those that use cepstral coefficients dervied from the Fourier transform.

3.11 The Mel scale

It is a perceptual scale motivated by properties of human hearing. The human ear resolves frequencies nonlinearly. The Mel scale attempts to mimic the human ear in terms of the manner with which frequencies are sensed and resolved. A Mel is a unit of measurement of perceived frequency of a tone.

The Mel scale was obtained by Stevens and Volkman (1937). They arbitrarily selected $1kHz$ and equated it with $1000mels$. Listeners were asked to change the frequency until the pitch they perceived was twice the reference, 10 times, etc and then half, 1/10, etc. These frequencies corresponded to 2000mels, 10000mels, 500 Mels, 100mels respectively.

The Mel-function is the operator which rescales the frequency domain.

$$f_{mel}(f_{lin}) = \Lambda \cdot \log_{10}(1 + \frac{f_{lin}}{\mu}) = \lambda \cdot \ln(1 + \frac{f_{lin}}{\mu}) \quad \text{with} \quad \lambda = \frac{\Lambda}{\ln(10)} \tag{3.11.1}$$

3.12 Liftering

Low order cepstral coefficients are sensitive to spectral slope, glottal pulse shape, etc. High order cepstral coefficients are sensitive to the analysis window position and other temporal artifacts. For SI recognition, it is best to minimize such speaker-dependent variations in the features prior to recognition. We can reduce the variations in these coefficients by using a raised sine window that emphasizes coefficients at the center of the window.

3.13 Acoustic Echo Cancellation

Echo is the result of outgoing signals being reflected back to the input. These reflections can be considered as delayed, attenuated and distorted versions of the outgoing signal. Echo can destroy the accuracy, usability and value of an ASR system.

Acoustic echo results when surrounding surfaces reflect the voice, and is typically associated with the use of handsets, hands-free phones and speakerphones. It is characterized by multiple reflections that create very complex and nonlinear echo signals, often with $180ms$ or more of variable delay.

The echo returning back is not just a delayed and attenuated version of the original signal; it also includes distortion. This distortion can be modeled using an "impulse response". The basic concept of echo cancellation is simple. First, the system must "learn the echo characteristics" by comparing the outgoing signal with the incoming one; because the outgoing signal is known, the echo signal can be identified and characterized. Once the echo characteristics are known, the algorithm continuously subtracts the expected echo from the incoming signal, leaving only a small amount of residual echo mixed with the incoming signal. The tail length should be greater than or equal to the total roundtrip echo delay plus the settling time for the impulse response of the hybrid. Real-world convergence times are typically $4 - 5s$ even for G.168-compliant

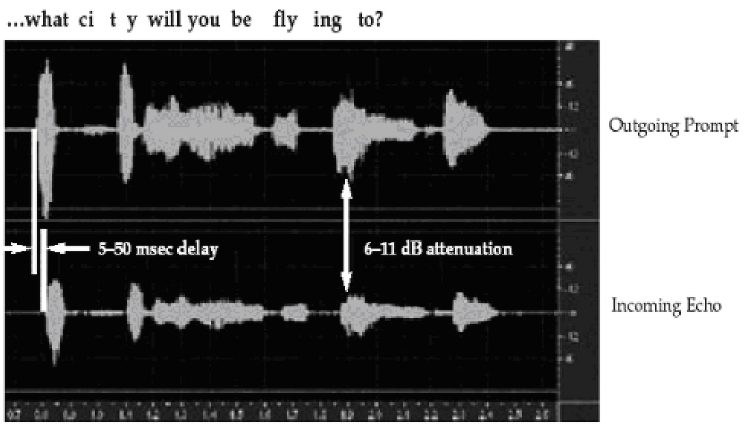


Figure 21: Acoustic Echo

systems. In general, echo cancellation systems employ a Non-Linear Processor (NLP) to eliminate this residual noise by silencing any signal that falls below an adaptive noise threshold. However, because NLP can mask the low energy consonants that ASR systems must recognize to achieve accurate results, NLP is disabled when echo cancellation is used with ASR systems. Doubling the tail length of an echo canceller quadruples the DSP resources required. Since ASR systems are particularly susceptible to echo it is important to solve the issue of how to eliminate echo in these applications.

Adaptive noise cancellation deals with the problem of removing correlated noise (that is, noise that has some redundancy associated with it).

Another technique is the use of a microphone array. The array can focus or listen in a specific direction while subtracting the noise in all other directions.

3.14 Noise Reduction

Performance of ASR systems can be degraded by a range of adverse conditions:

Environmental noise e.g., noise in a car or a factory

Acoustical distortions e.g, echoes, room acoustics

Change of microphones e.g., close-speaking, omnidirectional, or telephone

Limited frequency bandwidth e.g., in telephone transmission

Altered speaking manner e.g., shouting, whining, speaking quickly, etc.

SNR is critically important to ASR. A typical SNR for good accuracy is 24dB. As SNR drops to below 20dB, the accuracy will drop to values that would cause the average user to cease from using speech recognition. Since it can only be assumed that the user will want to speak at a “normal” conversational volume, the task, then, is to remove or reduce the noise.

There are many sources of acoustical distortion that can degrade the accuracy of an ASR system. Let’s consider an ASR system employeed in the aircraft.

1. additive noise from the aircraft
2. competing talker (ex. co-pilot)
3. reverberation from surface reflections in the cockpit
4. spectral shaping by microphone and the vocal tracts of the pilot

These sources of distortion cluster into two complementary classes: *additive noise* (as in 1,2), *distortions resulting the convolution of the speech signal with an unknown linear system* (as in 3,4).

Classically, two complementary techniques have been proposed in the cepstral domain: **Spectral Subtraction** (SS) and **Spectral Normalization** (SN). In SS one estimates the amount of background noise present during non-speech intervals, and subtracts the estimated spectral density of the noise from the incoming signal. In SN (or ”blind deconvolution”), one estimates the average spectrum when speech is present and applies a multiplicative normalization factor with respect to a reference spectrum. Since the cepstrum is the log of the magnitude of the spectrum, this corresponds to a simple additive correction of the cepstrum vector.

3.14.1 SS

The estimated noise spectrum is either over-subtracted or under-subtracted from the input spectrum, depending on the estimated instantaneous SNR (of the current analysis frame). First, we determine empirically 1) ”noise threshold” (putative max power level for noise frames) 2) ”speech threshold” (putative minimum power level for speech frames) they could easily be estimated from histograms of the average power for the signals in the analysis fxames. The estimated noise vector is obtained by averaging the cepstra of all frames with a power that falls below the noise threshold. Once the noise vector is estimated, a magnitude equal to that of the reference spectrum plus 5 dB is subtracted from the magnitude of the spectrum of the incoming signal, for all frames in which the power of the incoming signal falls below the noise threshold. If the power of the incoming signal is above the speech threshold, the magnitude of the reference spectrum minus 2.5 dB is subtracted from the magnitude of the spectrum of the incoming signal. The amount of over- or under-subtraction (in dB) is a linearly interpolated function of the instantaneous SNR (in dB) for incoming signals whose power is between the two thresholds. We note that we subtract the magnitudes of spectra rather than the more intuitively appealing spectral power because we found that magnitude subtraction provides greater recognition accuracy.

4 Phonetics and Phonology

4.1 Phonemes

Phonemes are the basic acoustic units of spoken language. For example, the "ee" sound in "Speech" is a phoneme. Phonemes are embraced by "/". A phoneme is a word-distinguishing sound unit in the language being spoken (like a letter, but for sounds). The actual sound that corresponds to a phoneme depends on:

- the adjacent phonemes in the word or sentence
- the accent of the speaker
- the talking speed

ASR systems match speech sounds to phonemes using acoustic models that represent the variability of the acoustic profile of a phoneme under different circumstances.

Phonemes can be classified into the following broad categories:

1. VOWELS
   (a) Monophthongs. American English has some 11 vowels having a single vowel quality, including the nine stressed vowels in the words "beet", "bit", "bet", "bat", "Bert", "boot", "book", "but", "bought" and the two reduced vowels as in the final syllables of "abbot" and "Hubert". Some linguists distinguish more than two reduced vowels, and some dialects of American English have an additional vowel exemplified by the first member of the contrastive pair "caught" / "cot".

   (b) Diphthongs - American English has 6 diphthongs - vowels which manifest a clear change in quality from start to end as in the words "bite", "Boyd", "bate", "beaut", "bout", "boat".

2. CONSONANTS
   (a) Approximants - English has 4 approximants or semivowels - speech sounds midway between a vowel and a consonant - the 'w' in "won", the 'l' in "like", the 'r' in "red", and the 'y' in "yes." In these phonemes, there is more constriction in the vocal tract than for the vowels, but less than the other consonant categories below.

   (b) Nasals - English has 3 nasals in which the airflow is blocked completely at some point in the oral tract, but in which the simultaneous lowering of the velum allows a weak flow of energy to pass through the nose - 'm' as in "me", 'n' as in "new", and 'ng' as in "sing".

(c) Fricatives - English has 9 fricatives - weak or strong friction noises produced when the articulators are close enough together to cause turbulence in the airflow - 'h', 'f', 'v', 'th' as in "thing", 'th' as in "the", 's', 'z', 'sh' as in "ship", and 'z' as in "azure".

(d) Plosives - English has 6 bursts or explosive sounds produced by complete closure of the vocal tract followed by a rapid release of the closure - 'p', 't', 'k', 'b', 'd', 'g'.

(e) Affricates - English has 2 affricates - plosives released with frication - the 'ch' sounds of "church" and the 'j' and 'dge' of "judge".

Phoneticians have developed a set of symbols which represent speech sounds not only for English, but for all existing spoken languages. The *International Phonetic Alphabet* (IPA) is recognized as the international standard for the transcription of phonemes in all of the world's languages. Since the IPA is not easily processed by computer, at the CSLU they use an IPA-like ASCII symbol set called Worldbet.

These terms are different in meaning but refer to things that are often confused. 1) phonetic unit (phonetic symbol) 2) phonemic unit (phonemic or phonological symbol) 3) monophone 4) biphone 5) triphone. "Phone" is used to be ambiguous between (1) and (2) when you might have a linguist arguing with you. Good phone sets for speech recognition purposes are always sets of phonemes, not sets of phonetic units. Mono-, bi-, and tri-phones are HMM models, statistical models, not units of human language that reside in human minds or brains or behavior. Depending on the language used there are about 35-50 phonemes in western European languages. In American English this is approximately 43 units.

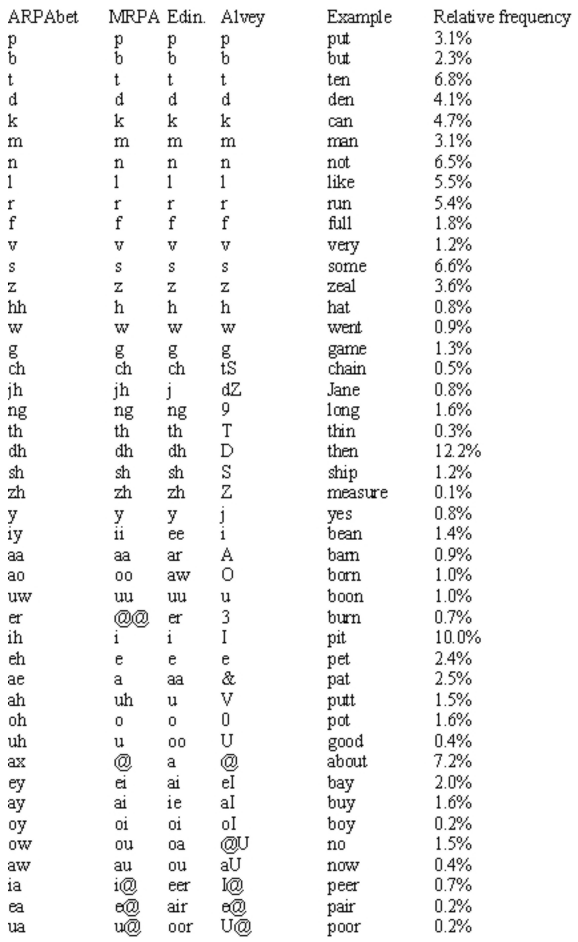
A **pronunciation** for a word is a sequence of phonemes **Example**: "ABANDONDED": *ax b ae n d ax n d ih d* (from BEEP) *@ bc b @ n dc d & n dc d I dc d* (from CSLU). A **Pronouncing Dictionary** is simply a list of words along with their pronunciations.

4.2 Homophones

Two words are **homophone** if they sound the same, i.e. they have the same phoneme sequence. **Example**: "BEET" and "BEAT": *b iy t* (from BEEP)

4.3 Formants

Each phoneme is distinguished by its own unique pattern in the spectrogram. For voiced phonemes, the signature involves large concentrations of energy called formants; within each formant, and typically across all active formants, there is a characteristic waxing and waning of energy in all frequencies which is the most salient characteristic



ARPAbet	MRPA	Edin.	Alvey	Example	Relative frequency
p	p	p	p	put	3.1%
b	b	b	b	but	2.3%
t	t	t	t	ten	6.8%
d	d	d	d	den	4.1%
k	k	k	k	can	4.7%
m	m	m	m	man	3.1%
n	n	n	n	not	6.5%
l	l	l	l	like	5.5%
r	r	r	r	run	5.4%
f	f	f	f	full	1.8%
v	v	v	v	very	1.2%
s	s	s	s	some	6.6%
z	z	z	z	zeal	3.6%
hh	h	h	h	hat	0.8%
w	w	w	w	went	0.9%
g	g	g	g	game	1.3%
ch	ch	ch	tS	chain	0.5%
jh	jh	j	dZ	Jane	0.8%
ng	ng	ng	9	long	1.6%
th	th	th	T	thin	0.3%
dh	dh	dh	D	then	12.2%
sh	sh	sh	S	ship	1.2%
zh	zh	zh	Z	measure	0.1%
y	y	y	j	yes	0.8%
iy	ii	ee	i	bean	1.4%
aa	aa	ar	A	barn	0.9%
ao	oo	aw	O	born	1.0%
uw	uu	uu	u	boon	1.0%
er	@@	er	3	burn	0.7%
ih	i	i	I	pit	10.0%
eh	e	e	e	pet	2.4%
ae	a	aa	&	pat	2.5%
ah	uh	u	V	putt	1.5%
oh	o	o	0	pot	1.6%
uh	u	oo	U	good	0.4%
ax	@	a	@	about	7.2%
ey	ei	ai	eI	bay	2.0%
ay	ai	ie	aI	buy	1.6%
oy	oi	oi	oI	boy	0.2%
ow	ou	oa	@U	no	1.5%
aw	au	ou	aU	now	0.4%
ia	i@	eer	I@	peer	0.7%
ea	e@	air	e@	pair	0.2%
ua	u@	oor	U@	poor	0.2%

Figure 22: Phonemes in English language

of what we call the human voice; this cyclic pattern is caused by the repetitive opening and closing of the vocal cords which occurs at an average of 125 times per second in the average adult male, and approximately twice as fast (250 Hz) in the adult female, giving rise to the sensation of pitch. Voicing is a relatively long-lasting phenomenon in speech; during voicing, the spectral or frequency characteristics of a formant evolves as phonemes unfold and succeed one another. Formants which are relatively unchanging over time are found in the monophthong vowels and the nasals; formants which are more variable over time are found in the diphthong vowels and the approximants, but in all cases the rate of change is relatively slow. In addition, there is good spectral continuity, the exceptions being the singularities involved in the beginning and end of the nasals (/m/, /n/, and /N/) and the lateral /l/.

Formant values can vary widely from person to person, but the spectrogram reader learns to recognize patterns which are independent of particular frequencies and which identify the various phonemes with a high degree of reliability.

The monophthong vowels have strong stable formants; in addition, these vowels can usually be easily distinguished by the frequency values of the first two or three formants, which are called F1, F2, and F3. For these reasons the monophthong vowels are often used to illustrate the concept of formants, but it is important to remember that all voiced phonemes have formants, even if they are not as easy to recognize and classify as the monophthong vowel formants. Voiceless sounds are not usually said to have formants; instead, the plosives should be visualized as a great burst of energy across all frequencies occurring after relative silence, while the aspirates and fricatives are better considered as clouds or oceans of relatively smooth energy along both the time and frequency axes.

In the vowels, F1 can vary from 300 Hz to 1000 Hz. The lower it is, the closer the tongue is to the roof of the mouth. The vowel /i:/ as in the word 'beet' has one of the lowest F1 values - about 300 Hz; in contrast, the vowel /A/ as in the word "bought" (or "Bob" in speakers who distinguish the vowels in the two words) has the highest F1 value - about 950 Hz. Pronounce these two vowels and try to determine how your tongue is configured for each.

F2 can vary from 850 Hz to 2500 Hz; the F2 value is proportional to the frontness or backness of the highest part of the tongue during the production of the vowel. In addition, lip rounding causes a lower F2 than with unrounded lips. For example, /i:/ as in the word "beet" has an F2 of 2200 Hz, the highest F2 of any vowel. In the production of this vowel the tongue tip is quite far forward and the lips are unrounded. At the opposite extreme, /u/ as in the word "boot" has an F2 of 850 Hz; in this vowel the tongue tip is very far back, and the lips are rounded.

F3 is also important is determining the phonemic quality of a given speech sound, and the higher formants such as F4 and F5 are thought to be significant in determining voice quality.

5 General techniques for acoustic variability

Statistical-based approaches (HMM) Variations in speech are modeled statistically using automatic learning procedures. This approach represents the current state of the art. The main disadvantage of statistical models is that they must make a priori modeling assumptions, which are liable to be inaccurate

Neural networks-based approaches (ANN) to do

Hybrid approaches (HMM/ANN) to do

The dominant technology used in ASR is called the **Hidden Markov Model** (HMM). This technology recognizes speech by estimating the likelihood of each phoneme at contiguous, small regions (*frames*) of the speech signal. Each word in a vocabulary list is specified in terms of its component phonemes. A search procedure is used to determine the sequence of phonemes with the highest likelihood. This search is constrained to only look for phoneme sequences that correspond to words in the vocabulary list, and the phoneme sequence with the highest total likelihood is identified with the word that was spoken. In standard HMMs, the likelihoods are computed using a *Gaussian Mixture Model* (GMM); in the HMM/ANN framework, these values are computed using an artificial neural network (ANN).

6 HMM

Il parlato è un messaggio codificato in una sequenza di simboli, le unità linguistiche della frase, le parole. Ma la pronuncia di una stessa parola non è mai uguale e pronuncie di parole diverse possono contenere gli stessi suoni. Le diverse forme d'onda prodotte nel pronunciare una parola possono essere pensate come realizzazioni di un processo stocastico. Su ogni unità linguistica riconoscibile dal sistema viene costruito un processo stocastico generatore, o modello acustico, che in fase di riconoscimento deve essere identificato con elevata accuratezza, basandosi su una singola pronuncia.

Due complicazioni si aggiungono, la presenza di rumore di fondo, che aumenta la variabilità delle forme d'onda, e la continuità del parlato, che non permette di isolare immediatamente le singole unità linguistiche.

Consideriamo dapprima il caso semplice in cui le unità linguistiche riconoscibili dal sistema siano parole, ben isolate da pause di silenzio (*isolated word recognition*).

La forma d'onda della parola pronunciata (tra quelle riconoscibili $\{w_i\}$), w, è dapprima segmentata uniformemente nel tempo in porzioni assunte statisticamente stazionarie (tipicamente 10 ms). Sulla porzione $t - esima$ (detta *frame*) viene calcolato un vettore di parametri, $\mathbf{o}_t$ (con n componenti). Su tutta la durata della pronuncia di w si ottiene la sequenza di vettori

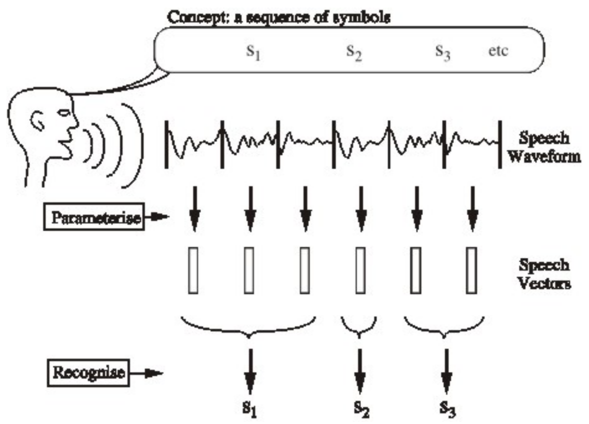


Figure 23: Il sistema fonatorio codifica gli elementi simbolici della frase in forme d'onda acustiche. Il sistema ASR decodifica le forme d'onda e restituisce i simboli

$$\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T \tag{6.0.1}$$

Supponiamo inizialmente che gli elementi di $\mathbf{o}_t$ siano variabili continue. La sequenza dei vettori è una rappresentazione esatta della forma d'onda. La parola riconosciuta sarà quella, presente nel vocabolario $\{w_i\}$, che massimizza la probabilità

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})} \tag{6.0.2}$$

cioè

$$w = argmax[P(w_i|\mathbf{O})] \tag{6.0.3}$$

La stima diretta della densità di probabilità congiunta condizionale $P(\mathbf{O}|w_i)$ (*likelihood function*) è proibitiva a causa della dimensionalità di $\mathbf{O}$. Si preferisce stimare $P(\mathbf{O}|M_i)$, dove M_i è un modello parametrico di Markov associato alla parola w_i.

Il modello HMM M è una macchina a N stati che a ogni istante discreto t passa da uno stato (i) all'altro (j) con probabilità a_{ij}, producendo il vettore di uscita $\mathbf{o}_t$ con una densità di probabilità di emissione $b_j(\mathbf{o}_t)$. Per semplicità è stabilito che l'ingresso nello stato iniziale (1) e l'uscita dallo stato finale (N) non emettono vettori. Questi stati rappresentano i punti di aggancio per costruire sequenze di HMMs, una esigenza che nasce passando ai sistemi continui. Mentre la sequenza $\mathbf{O}$ prodotta da M è osservabile, non è osservabile la sequenza $X = x(0), \cdots, x(T+1)$ degli stati attraversati da M, ecco perchè M è detto *Hidden Markov Model.* Nella figura precedente la sequenza degli stati X è 1,2,2,3,4,4,5,6.

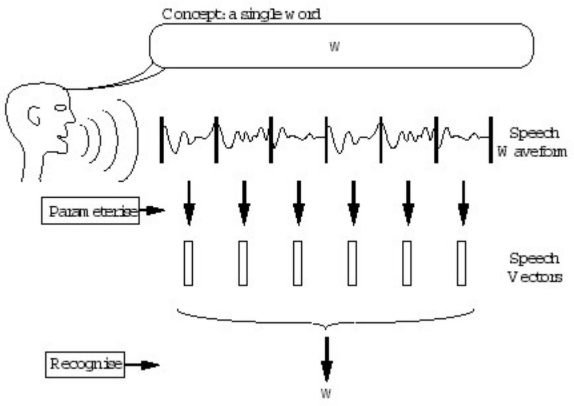


Figure 24: Isolated word recognition

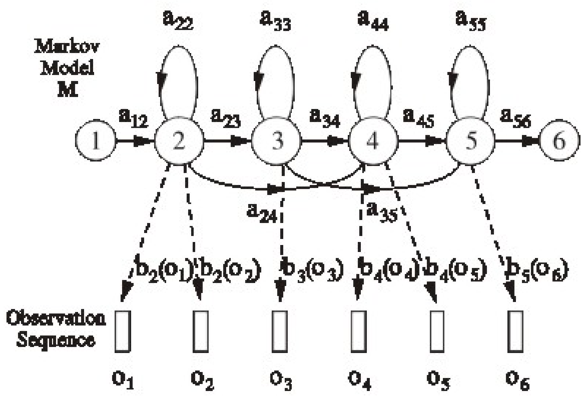


Figure 25: HMM a 6 stati

$$P(\mathbf{O}|M) = \sum_X P(\mathbf{O}, X|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} a_{x(t)x(t+1)} b_{x(t)}(\mathbf{o}_t) \tag{6.0.4}$$

Esistono procedure ricorsive per calcolare la precedente espressione e quindi per risolvere il problema dell'ASR, cioè dato $\mathbf{O}$ trovare il modello M_i più probabile che genera $\mathbf{O}$. Ciò implica che siano note le quantità a_{ij} e $b_j(\mathbf{o}_t)$, cioè i parametri del modello HMM associato a ciscuna parola del dizionario. Queste quantità possono essere stimate con un algoritmo a partire da un certo numero di esempi rappresentativi di pronuncia di ciascuna parola.

Inizialmente si è assunto che $\mathbf{o}_t$ fosse un processo multivariato continuo. Le densità congiunte di uscita, $b_j(\mathbf{o}_t)$ possono essere rappresentate come una mescolanza di gaussiane multivariate (*Gaussian Mixture Densities*, GMD). Assumiamo anche che $\mathbf{o}_t$ possa essere scomposto in S sotto-vettori o *flussi* indipendenti, $\mathbf{o}_{st}$, ciascuno con un suo peso γ_s, che in qualche modo ne determina l'importanza. La probabilità di eventi congiunti indipendenti è il prodotto delle probabilità marginali, e l'espressione per $b_j(\mathbf{o}_t)$ diventa

$$b_j(\mathbf{o}_t) = \prod_{s=1}^{S} \Big[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(\mathbf{o}_{st}, \mu_{jsm}, \Sigma_{jsm}) \Big]^{\gamma_s} \tag{6.0.5}$$

dove M_s è il numero di gaussiane della mistura del flusso s-esimo, c_{jsm} il relativo peso, mentre $\mathcal{N}(\cdot, \mu, \Sigma)$ è la gaussiana multivariata con vettore di media μ e matrice di covarianza Σ

$$\mathcal{N}(\mathbf{o}, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(\mathbf{o}-\mu)'\Sigma^{-1}(\mathbf{o}-\mu)} \tag{6.0.6}$$

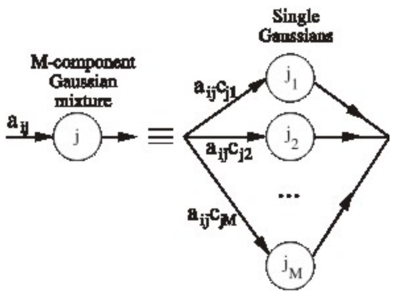


Figure 26: Rappresentazione di una mistura. Le M componenti della mistura possono essere pensate come altrettanti sotto-stati di un singolo stato

Il fatto di distinguere vari *streams* serve a tenere conto della presenza di sorgenti di informazioni multiple e indipendenti nell'apparato fonatorio. Tipicamente si utilizzano $S = 4$ sorgenti principali

1. parametri base (coefficienti LP)
2. coefficienti differenza del primo ordine (delta)
3. coefficienti differenza del secondo ordine (accelerazioni)
4. logaritmo dell'energia

6.1 Addestramento di HMM mediante la formula di Baum-Welch

La formula di Baum-Welch (*Baum-Welch re-estimation formulae*) permette di calcolare i parametri ottimi, nel senso della massima verosimiglianza (*maximum likelihood*), del modello di Markov M, a partire da una stima iniziale. Per illustrare il metodo, trascuriamo per il momento la presenza di più flussi ($S = 1$). Il problema è stimare i parametri ottimali della densità

$$b_j(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(\mathbf{o}-\mu_j)'\Sigma_j^{-1}(\mathbf{o}-\mu_j)} \tag{6.1.1}$$

a partire dalla sequenza di uscita $\mathbf{O}$ (lasciando un attimo da parte le probabilità di transizione). Se ci fosse un solo stato j nella HMM tutti i vettori $\mathbf{o}_t$ sarebbero generati da quell'unico stato, e avremmo la soluzione semplice

$$\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{o}_t \tag{6.1.2}$$

$$\hat{\Sigma} = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{o}_t - \hat{\mu})(\mathbf{o}_t - \hat{\mu})' \tag{6.1.3}$$

Ma gli stati sono molti, e inoltre non sappiamo da quale stato un dato vettore della sequenza osservata $\mathbf{O}$ è stato emesso, proprio perchè non è nota la sequenza X. Il modulo **HInit** di HTK ha la funzione di tentare una associazione iniziale tra vettori osservati e stati, quindi per ciascuno stato calcola le stime $\hat{\mu}$ e $\hat{\Sigma}$ usando le precedenti formule. HInit calcola poi la sequenza più probabile usando l'algoritmo di Viterbi, riassegna i vettori agli stati e riaggiorna le stime dei parametri. Questo processo è reiterato finchè le stime convergono. In realtà HTK segue un procedimento leggermente differente. Dato che il calcolo di $P(\mathbf{O}|M)$ utilizza tutte le possibili sequenze X, ciascun $\mathbf{o}_t$ contribuisce al computo dei parametri relativi a tutti gli stati, quindi invece di assegnare rigidamente ciascun $\mathbf{o}_t$ a un singolo stato lo si può assegnare a tutti gli stati in proporzione alla probabilità che M passi occupi quello stato all'istante considerato t. Detta $L_j(t)$ questa *probabilità di occupazione*, avremo le seguenti *formule di Baum-Welch*

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}_t}{\sum_{t=1}^{T} L_j(t)} \tag{6.1.4}$$

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}_t - \hat{\mu})(\mathbf{o}_t - \hat{\mu})'}{\sum_{t=1}^{T} L_j(t)} \tag{6.1.5}$$

Una formula analoga esiste per le probabilità di transizione.

$L_j(t)$ si calcola con l'algoritmo *Forward-Backward*. La probabilità diretta (*Forward Probability*) è definita come

$$\alpha_j(t) \equiv P(\mathbf{o}_1, \ldots, \mathbf{o}_t, x(t) = j|M) \tag{6.1.6}$$

ed è ricorsivamente calcolata con

$$\alpha_j(t) = \Big[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\Big] b_j(\mathbf{o}_t) \tag{6.1.7}$$

$$\texttt{condizioni al contorno} \quad \alpha_1(1) = 1 \quad \alpha_j(1) = a_{1j}b_j(\mathbf{o}_1) \quad j = 1\ldots, N \tag{6.1.8}$$

La probabilità inversa (*Backward Probability*) è definita come

$$\beta_j(t) \equiv P(\mathbf{o}_{t+1}, \ldots, \mathbf{o}_T|x(t) = j, M) \tag{6.1.9}$$

ed è ricorsivamente calcolata con

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1) \tag{6.1.10}$$

$$\texttt{condizioni al contorno} \quad \beta_i(T) = a_{iN} \quad i = 1\ldots, N \quad \beta_1(1) = \sum_{j=2}^{N-1} a_{1j}b_j(\mathbf{o}_1)\beta_j(1) \tag{6.1.11}$$

Valgono le seguenti identità

$$P(\mathbf{O}|M) = \alpha_N(T) \tag{6.1.12}$$

$$P(x(t) = j|\mathbf{O}, M) = \alpha_j(t)\beta_j(t) \tag{6.1.13}$$

Dato che $L_j(t) \equiv \frac{P(\mathbf{O},x(t)=j|M)}{P(\mathbf{O}|M)}$ si ricava

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\alpha_N(T)} \tag{6.1.14}$$

I calcoli descritti comportano continue moltiplicazioni di numeri che rappresentano probabilità, con un costante rischio di *underflow*. Per evitare ciò si utilizza l'aritmetica logaritmica. In HTK moduli che realizzano le operazioni descritte sono **HInit** e **HRest**.

6.2 Riconoscimento mediante l'algoritmo di Viterbi

L'*algoritmo di Viterbi* è alla base della fase di riconoscimento. Sfrutta sostanzialmente la stessa equazione ricorsiva del *Forward Algorithm* descritto sopra.

$$\Phi_j(t) \equiv P(\mathbf{o}_1, \ldots, \mathbf{o}_t, x(t) = j|M) \tag{6.2.1}$$

ed è ricorsivamente calcolato con

$$\Phi_j(t) = max_i\{\Phi_i(t-1)a_{ij}\}b_j(\mathbf{o}_t) \tag{6.2.2}$$

$$\texttt{condizioni al contorno} \quad \Phi_1(1) = 1 \quad \Phi_j(1) = a_{1j}b_j(\mathbf{o}_1) \quad j = 1\ldots, N \tag{6.2.3}$$

oppure, usando aritmetica logaritmica,

$$\Psi_j(t) = max_i\{\Psi_i(t-1) + log(a_{ij})\} + log(b_j(\mathbf{o}_t)) \tag{6.2.4}$$

avendo definito $\Psi \equiv log(\Phi)$.

Vale dunque

$$\hat{P}(\mathbf{O}|M) = \Phi_N(T) = max_i\{\Phi_i(T)a_{iN}\} \tag{6.2.5}$$

L'algoritmo di Viterbi può essere rappresentato come nella seguente matrice di probabilità logaritmiche, con gli stati di M che si estendono in verticale e i *frames* del parlato (il tempo t) che si estendono in orizzontale.

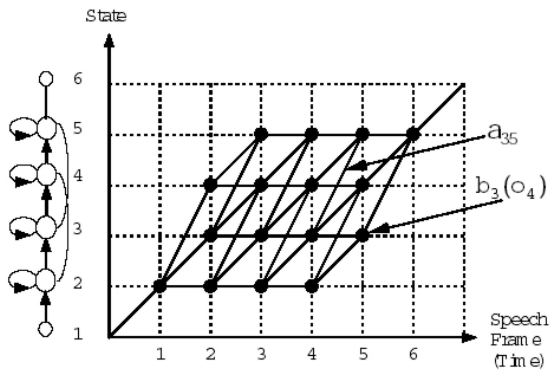


Figure 27: Algoritmo di Viterbi per un *isolated word recognition.*

L'algoritmo consiste nella ricerca del cammino ottimo attraverso questa matrice. Ciascun arco contribuisce con la somma di una probabilità logaritmica a quella dell'intero cammino. Questo concetto può essere generalizzato al caso di *continous speech recognition.*

6.3 Caso Continuo

Il tool **HVite** di HTK (con le librerie **HNet** e **HRec**) gestisce il *continuous speech recognition* (e l'*isolated-word recognition* come caso speciale). L'estensione di quanto detto finora al caso continuo comporta semplicemente la concatenazione di più HMMs, ciascuna delle quali rappresenta un simbolo, sia esso un'intera parola (*connected speech recognition*) o un singolo fonema (*continous speech recognition*). Nella fase di addestramento nasce il problema di suddividere la frase pronunciata nei segmenti di competenza delle singole HMMs. Se le frasi sono poco numerose di solito tali segmenti vengono marcati a mano. HTK utilizza **HInit** e **HRest** per inizializzare i modelli sub-word (*bootstrap training*). La fase di addestramento vera e propria utilizza il tool **HERest** (*embedded training*). L'*embedded training* è ancora basato sull'algoritmo di Baum-Welch come nel caso *isolated-word*, ma invece di addestrare un modello alla volta, tutti i modelli vengono addestrati in parallelo. La procedura è la seguente:

1. alloca e azzera memoria per gli accumulatori (sommatorie) necessari al calcolo dei parametri di tutte le HMMs
2. prende la successiva frase di addestramento w (con la sua trascrizione simbolica) e la sequenza associata $\mathbf{O}$
3. costruisce una HMM composita (M) collegando in cascata i modelli sub-word M_k correspondenti alla trascrizione simbolica
4. per ogni tempo t e per ogni stato j di M calcola le probabilità $\alpha_j(t)$ e $\beta_j(t)$
5. calcola le probabilità di occupazione di stato $L_j(t)$ e aggiorna gli accumulatori (cioè le somme che compaiono nelle formule di Baum-Welch)
6. ripete da 2 fino a esaurire tutto l'insieme delle frasi di addestramento
7. utilizza gli accumulatori per calcolare nuove stime di parametri per tutti i modelli

Tutti questi passi vengono eseguiti più volte, sullo stesso insieme di addestramento, fino a raggiungere la convergenza desiderata. La procedura descritta non necessita espressamente della segmentazione delle frasi, mentre è richiesta la trascrizione simbolica. L'estensione descritta dell'algoritmo di Baum-Welch ai modelli sub-word non richiede grosse modifiche rispetto al caso *isolated-word*. Diverso è per l'algoritmo di Viterbi. In questo caso HTK utilizza la tecnica del *Token Passing*. Ciascuno stato j di una HMM, al tempo t ha un singolo token (una variabile strutturata in memoria), contenente tra le altre informazioni la probabilità logaritmica $\Psi_j(t)$. L'algoritmo evolve in due passi:

1. passa una copia del token dello stato i a tutti gli stati j concatenati, aumentando la probabilità logaritmica di $log(a_{ij}) + log(b_j(\mathbf{o}_t))$

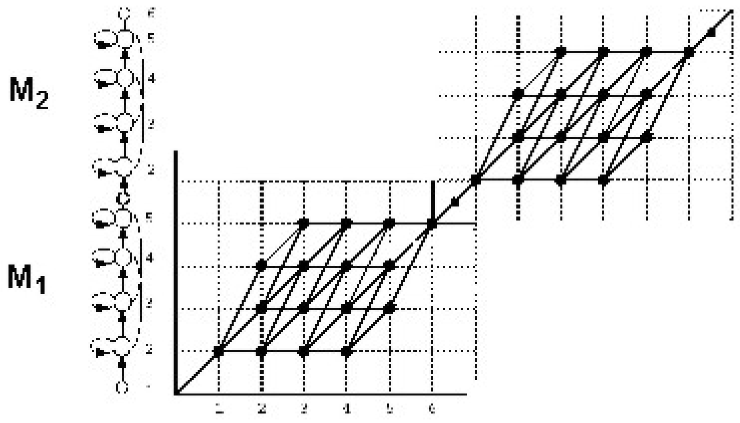


Figure 28: Algoritmo di Viterbi esteso a una concatenazione di HMMs con la tecnica del token-passing. Notare la fusione degli stati iniziali e finali dei modelli sub-word consecutivi

2. controlla tutti i token e conserva solo quello avente la probabilità maggiore

La figura seguente mostra una rete semplice in cui ciascuna parola è definita da una sequenza di HMMs su base fonema. Tutte le parole sono chiuse in un loop. La grammatica corrispondente permette quindi il riconoscimento di una concatenazione lunga a piacere delle parole del vocabolario. Le scatole ovali sono le HMMs mentre le scatole quadrate sono i nodi di fine parola (*word-ends*).

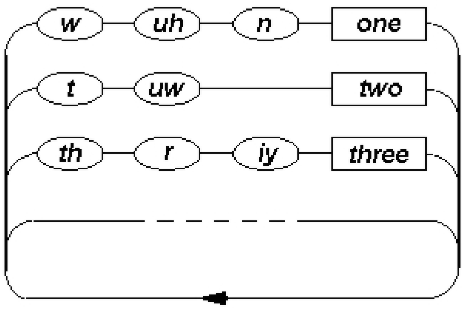


Figure 29: schema di rete per un sistema continous-speech

Questa rete è sostanzialmente una singola HMM composita alla quale si può applicare la tecnica del *token-passing.* L'unica differenza è che occorre inserire nella struttura del token informazioni aggiuntive. Quando il token ottimo raggiunge la fine della frase deve essere possibile ritracciare tutto il percorso a ritroso, in modo da ricostruire la giusta sequenza delle parole.

Un modo per fare questo è il seguente. Ogni token ha un puntatore di fine parola (*word-end link*). Quando il token passa attraverso un nodo di fine parola viene generato un *Word Link Record* (WLR) in cui viene memorizzata l'identità della parola appena attraversata. Il link al token attuale viene sostituito dal puntatore al WLR appena creato. Quando la frase di test è stata processata, il WLR del token vincente (cioè quello con probabilità logaritmica maggiore) può essere ripercorso a ritroso ottenendo la sequenza di parole riconosciuta.

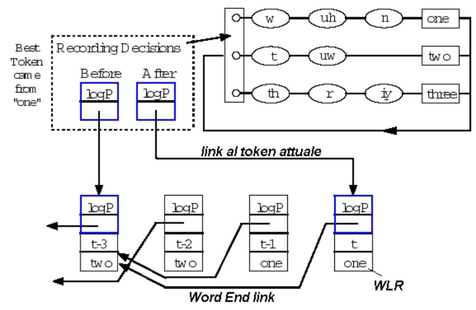


Figure 30: Implementazione del token-passing sulla grammatica della figua precedente.

Lo stesso principio può essere usato per tenere memoria non solo delle parole attraversate durante il riconoscimento, ma anche di altre informazioni al livello di modelli e singoli stati. Per esempio, altre al token vincente alla fine di una parola si possono memoriazzare altri token, ottenendo una varietà di ipotesi di riconoscimento (algoritmo *lattice N-best*). Questi sistemi sono sub-ottimi in quanto le loro potenzialità sono limitate dall'uso di un solo token per stato. Se invece ogni stato può avere un token per ciascuna diversa parola attraversata si ha un sistema ottimo (algoritmo *word N-best*). I moduli HNet e HRec di HTK implementano tutte le tecniche discusse (single e multiple-token passing recognition, single-best output, lattice output, N-best lists, supporto per cross-word context-dependency, lattice rescoring e forced alignment) e sono invocate dal tool HVite.

6.4 Speaker Adaptation

HTK permette anche di adattare le HMMs alle caratteristiche di un particolare speaker (*Speaker Adaptation*). I tools sono HERest e HVite. HERest esegue an adattamento off-line supervisionato, mentre HVite riconosce i dati di adattamento e usa le trascrizioni generate per eseguire l'adattamento.

7 ETSI Front-end

The signal processing Front-end summarizes the spectral characteristics of the speech waveform into a sequence of acoustic vectors that are suitable for processing by the acoustic model.

The first ETSI standard for DSR (*Distributed Speech Recognition*) front-end, ES201-108, was published in February 2000 and is based on the Mel-Cepstrum representation that has been used extensively in speech recognition systems. A second standard was published by ETSI in November 2003 that provides substantially improved recognition performance in background noise (ES202-050 V1.1.3). Performance evaluation of this standard showed an average of 53% reduction in speech recognition error rates in noise compared to ES201-108.

In the ES202-050, noise reduction is performed first. Then, waveform processing is applied to the de-noised signal and cepstral features are calculated. At the end, blind equalization is applied to the cepstral features. Voice activity detection (VAD) for the non-speech frame dropping is also implemented.

ETSI standards for DSR contain also a Feature Compression and Encoding in order to transmit speech feature vectors to a remote Back-End for recognition processing. But we are interested in the Fueature Extraction part only.

HTK is a complete speech recognition packet modularly consisting of Front-End Processing and HMM recognition engine. Therefore, the Front-End Processing can easily be exchanged. Basically, ETSI's Front-End represents a subset of HTK's Front-End. ETSI omits cepstral liftering and calculation of cepstral derivatives. Further, HTK applies a slightly different Mel-frequency filter bank and DCT compared to ETSI, though affecting the speech features in such a way that interchanging the two Front-End Processings between training and testing phase results in a dramatic drop of the recognition performance.

7.1 Analog-to-digital conversion

The specifics of the analog-to-digital conversion are not part of the present document. The output sampling rate of the ADC block is $f_s = 16kHz$.

7.2 Buffering

7.3 Offset compensation

A notch filtering operation is applied to the digital samples of the input speech signal s_{in} to remove their DC offset, producing the offset-free input signal s_{of}.

$$s_{of}(n) = s_{in}(n) - s_{in}(n-1) + 0.999 \cdot s_{of}(n-1) \tag{7.3.1}$$

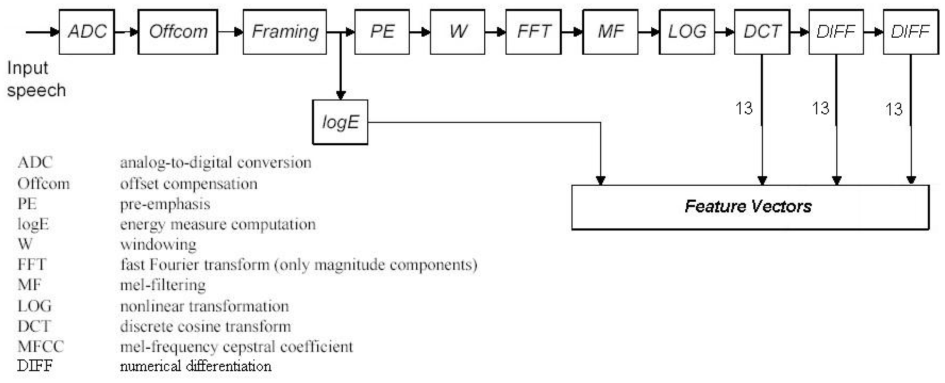

ADC
Offcom
Framing
PE
W
FFT
MF
LOG
DCT
DIFF
DIFF
Input speech
logE
13
13
13
Feature Vectors
ADC analog-to-digital conversion
Offcom offset compensation
PE pre-emphasis
logE energy measure computation
W windowing
FFT fast Fourier transform (only magnitude components)
MF mel-filtering
LOG nonlinear transformation
DCT discrete cosine transform
MFCC mel-frequency cepstral coefficient
DIFF numerical differentiation

Figure 31: Block diagram of the front-end algorithm

7.4 Framing

This is the process of splitting the continuous stream of signal samples into segments of constant length to facilitate blockwise processing of the signal. The offset-free input signal is divided into overlapping frames of N_F samples. The frame shift interval (difference between the starting points of consecutive frames) defines the number of frames per unit time, or *Frame Rate*. It is common to have 100 frames per second, so a new frame is started every $10ms$.

A new frame contains the last $7.5ms$ of the previous frame's data and the first $7.5ms$ of the next frame's data (symmetric window). Thus, even though a new frame is made every $10ms$, each frame is $25ms$ in duration. This overlap decreases problems that might otherwise occur due to signal data discontinuity.

For $f_s = 16kHz$ the **Frame Length** is $25ms$ (or $N_F = 400$ samples) and the **Frame Rate** is $10ms$ (or $M = 160$ samples).

7.5 Energy

The logarithmic frame energy measure ($logE$) is computed as

$$LogE = ln\big(\sum_{i=1}^{N_F} s_{of}(i)^2\big) \tag{7.5.1}$$

A floor is used in the energy calculation which makes sure that the result is not less than -50. The floor value (lower limit for the argument of ln) is approximately 2e-22.

7.6 Pre-emphasis

This stage spectrally flattens the offset-free signal using a first order FIR filter:

$$s_{pe}(n) = s_{of}(n) - 0.97 \cdot s_{of}(n-1) \qquad 1 \leq n \leq N_F \tag{7.6.1}$$

Other parameter $\alpha \in [0.9, 1]$ could be used. CMU Sphinx 3.2 system uses and ETSI ES201-108 suggests $\alpha = 0.97$.

7.7 Windowing

This is the process of multiplying a waveform signal segment by a time window of given shape, to minimize the effect of discontinuities at the edges of the frame during FFT. The transformation is:

$$s_w(n) = s_{pe}(n)w(n) \qquad 1 \leq n \leq N_F \tag{7.7.1}$$

A Hamming window of length N_F is applied:

$$s_w(n) = s_{pe}(n)\{0.54 - 0.46 \cdot cos(\tfrac{2\pi(n-1)}{N_F-1})\} \qquad 1 \leq n \leq N_F \tag{7.7.2}$$

The constants used are the same of CMU Sphinx 3.2 system and ETSI ES201-108 specifications.

7.8 FFT

Each frame of N_F samples is zero padded to make the frame size a power of two (N'_F). In our case $N_F = 400$, so $N'_F = 512$. After that a $N_{FFT} = N'_F$ -point Fourier transform is applied to convert the frame from the time domain to the frequency domain.

$$X(k) = \sum_{n=0}^{N_{FFT}-1} s_w(n)e^{-\frac{2\pi jnk}{N_{FFT}}} \qquad 0 \leq k \leq N_{FFT} - 1 \tag{7.8.1}$$

We need to compute the power spectrum of the signal

$$P(k) = |X(k)|^2 \qquad 0 \leq k \leq \frac{N_{FFT}}{2} \tag{7.8.2}$$

Note that, due to symmetry, only $P(0)..P(\frac{N_{FFT}}{2})$ are used for further processing.

7.9 Mel filtering

A popular alternative to LP analysis is Filter-bank analysis. This technique is based on the assumption that the human hear resolves frequencies non-linearly across the audio spectrum. The analysis in the cochlea takes place on a nonlinear frequency scale (known as the Bark scale or the mel scale). This scale is approximately linear up to about $1000Hz$ and is approximately logarithmic thereafter. Empirical evidence suggests that designing a front-end to operate in a similar nonlinear manner improves recognition performance. So, in the feature extraction, it is very common to perform a frequency warping of the frequency axis after the spectral computation.

It is assumed that the useful frequency band lies between a low frequency boundary f_{start} and half of the actual sampling frequency $\frac{f_s}{2}$. This band is divided into K_{FB} channels equidistant in Mel frequency domain.

In the Aurora Front-end each channel has triangular-shaped frequency window. Consecutive channels are half-overlapping. In their proposal, parameters are chosen as follows:

$$f_s = 8000Hz \tag{7.9.1}$$

$$K_{FB} = 23 \tag{7.9.2}$$

$$f_{start} = 64Hz \tag{7.9.3}$$

The choice of the starting frequency of the filter bank, $64Hz$, roughly corresponds to the case where the full frequency band is divided into 24 channels and the first channel is discarded.

Firstly let's remember the Mel transformation $f_{mel}(\bullet)$ and its inverse

$$f_{mel}(f_{lin}) = 2595 \cdot \log_{10}(1 + \frac{f_{lin}}{700}) \qquad [Mel] \tag{7.9.4}$$

$$f_{lin}(f_{mel}) = 700 \cdot (10^{\frac{f_{mel}}{2595}} - 1) \qquad [Hz] \tag{7.9.5}$$

The central frequencies of the filters are calculated from the Mel-function, in order to have an equidistant distribution of the bands in the Mel domain.

$$f_{c_i} = f_{lin}\{f_{mel}(f_{start}) + i\frac{f_{mel}(f_s/2) - f_{mel}(f_{start})}{K_{FB} + 1}\} \qquad 1 \le i \le K_{FB} \tag{7.9.6}$$

In terms of FFT index, the central frequencies of the filters correspond to

$$cbin_i = round(\frac{f_{c_i}}{f_s}N_{FFT}) \qquad 1 \le i \le K_{FB} \tag{7.9.7}$$

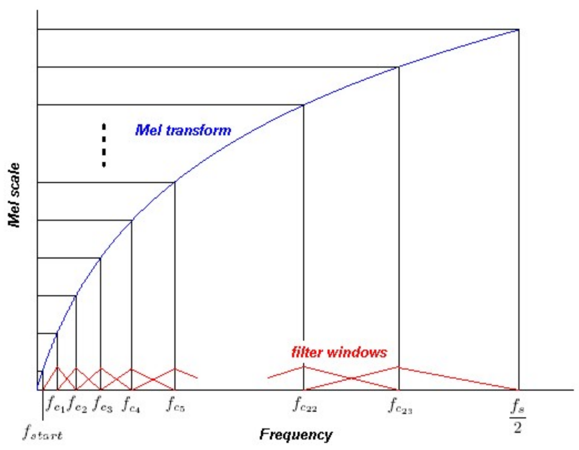

Mel scale
Mel transform
filter windows
f_{c_1} f_{c_2} f_{c_3} f_{c_4} f_{c_5} $f_{c_{22}}$ $f_{c_{23}}$ $\frac{f_s}{2}$
f_{start}
Frequency

Figure 32: Linear to Mel frequency mapping. Filter shape and positioning

where $round(\bullet)$ stands for rounding towards the nearest integer. The output of the Mel filter bank is the weighted sum of the magnitude spectrum values in each band.

$$E_{FB}(i) = \sum_{k=0}^{N_{FFT}/2} P(k)W_i(k) \qquad 1 \le i \le K_{FB} \tag{7.9.8}$$

For $W_i(k)$ Aurora Front-end uses triangular, half-overlapped windows:

$$E_{FB}(i) = \sum_{k=cbin_{i-1}}^{cbin_i} \frac{k - cbin_{i-1} + 1}{cbin_i - cbin_{i-1} + 1} P(k) + \sum_{k=cbin_i+1}^{cbin_{i+1}} \left(1 - \frac{k - cbin_i}{cbin_{i+1} - cbin_i + 1}\right) P(k) \qquad 1 \le i \le K_{FB} \tag{7.9.9}$$

where $cbin_0$ and $cbin_{K_{FB}+1}$ denote the FFT bin indices corresponding to the starting frequency and half of the sampling frequency, respectively,

$$cbin_0 = round(\frac{f_{start}}{f_s} N_{FFT}) \tag{7.9.10}$$

$$cbin_{24} = round(\frac{f_s/2}{f_s} N_{FFT}) = \frac{N_{FFT}}{2} \tag{7.9.11}$$

7.10 Non-linear transformation

The output of Mel filtering is subjected to a logarithm function (natural logarithm)

$$S_{FB}(i) = ln(E_{FB}(i)) \qquad 1 \le i \le K_{FB} \tag{7.10.1}$$

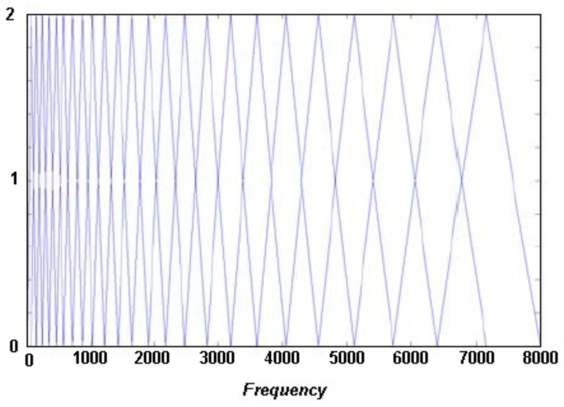


Figure 33: Filter Bank with 23 sub-bands, 16 kHz sampling frequency

A flooring is applied in such a way that the log filter bank outputs cannot be smaller than -10 (in Aurora specs was -50).

7.11 Cepstral coefficients

The term *cepstrum* was introduced by Bogert et al. and has come to be accepted terminology for the inverse Fourier transform of the logarithm of the power spectrum of a signal. (L.R.Rabiner and R.W.Schafer, Digital Processing of Speech Signals, Prentice Hall, Englewood Cliffs, NJ, 1978) In 1963, Bogert, Healy and Tukey published a paper with the unusual title "The Quefrency Analysis of Time Series for Echoes: Cepstrum, Pseudoautocovariance, Cross-Cepstrum, and Saphe Cracking." They observed that the logarithm of the power spectrum of a signal containing an echo has an additive periodic component due to the echo, and thus the Fourier transform of the logarithm of the power spectrum should exhibit a peak at the echo delay. They called this function the cepstrum, interchanging letters in the word spectrum because "in general, we find ourselves operating on the frequency side in ways customary on the time side and vice versa."

In ES202-050 13 cepstral coefficients are calculated from the output of the Non-linear transformation block by applying a DCT.

$$c(j) = \sum_{i=1}^{K_{FB}} S_{FB}(i) cos(\frac{j\pi(i-0.5)}{K_{FB}}) \qquad 0 \leq j \leq 12 \tag{7.11.1}$$

The sequence $S_{FB}(i)$ is real with mirror symmetry. Standard IFFT requires complex arithmetic, DCT does not. The DCT implements the same function as the FFT more efficiently by taking advantage of the redundancy in a real signal. Besides, the DCT is more efficient computationally.

Notice that ES202-050, in the case that $f_s = 16kHz$, increases by 3 the number of FB bands K_{FB}.

ES202-050 considers the final feature vector consisting in 14 coefficients: the log-energy coefficient $LogE$ and the 13 cepstral coefficients $c(0)\ldots c(12)$. Coefficient $c(0)$ is often redundant when $LogE$ is used. Depending on the application, either the coefficient $c(0)$ or the log-energy coefficient, or a combination both may be used.

7.12 Delta-Cepstral and acceleration coefficients

The performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters, the first order regression coefficients (called *delta coefficients*) and the second order regression coefficients (called *acceleration coefficients*, or delta-delta coefficients). The delta coefficients are computed using the following regression formula

$$\Delta c(j,t) = \frac{\sum_{l=1}^{L_D} l[c(j,t+l) - c(j,t-l)]}{2\sum_{l=1}^{L_D} l^2} = \frac{\sum_{l=-L_D}^{L_D} lc(j,t+l)}{2\sum_{l=1}^{L_D} l^2} \qquad 0 \leq j \leq 12 \tag{7.12.1}$$

The same formula is applied to $\Delta c(j,t)$ to obtain acceleration coefficients.

$$\Delta\Delta c(j,t) = \frac{\sum_{l=1}^{L_{DD}} l[\Delta c(j,t+l) - \Delta c(j,t-l)]}{2\sum_{l=1}^{L_{DD}} l^2} = \frac{\sum_{l=-L_{DD}}^{L_{DD}} l\Delta c(j,t+l)}{2\sum_{l=1}^{L_{DD}} l^2} \qquad 0 \leq j \leq 12 \tag{7.12.2}$$

The delta and accumulation windows L_D and L_{DD} not need to be the same. When delta and acceleration coefficients are requested, they are computed for all static parameters including energy if present. In some applications, the absolute energy is not useful but time derivatives of the energy may be.

7.13 Feature Vectors

The final feature vector is built by adding 13 first order regression-delta coefficients, computed over $L_D = 5$ frames, resulting in a feature vector dimension of 26.

8 ETSI Advanced Front-end

to do

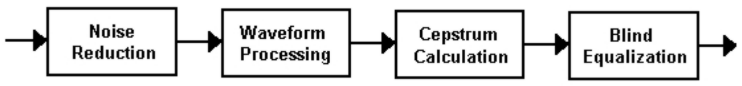


Figure 34: Block diagram of the Advanced Front-end

8.1 Noise Reduction

The noise reduction block processes a single audio stream to remove stationary or slowly varying background noise. The amount of noise suppression is about $20dB(?)$ during pauses and $0 - 6dB(?)$ during speech.

In the ES202-050 the noise reduction is based on Wiener filter theory and it is performed in two stages on a frame-by-frame basis. The input signal is first de-noised in the first stage and the output of the first stage then enters the second stage. In the second stage, an additional, dynamic noise reduction is performed, which is dependent on the SNR of the processed signal.

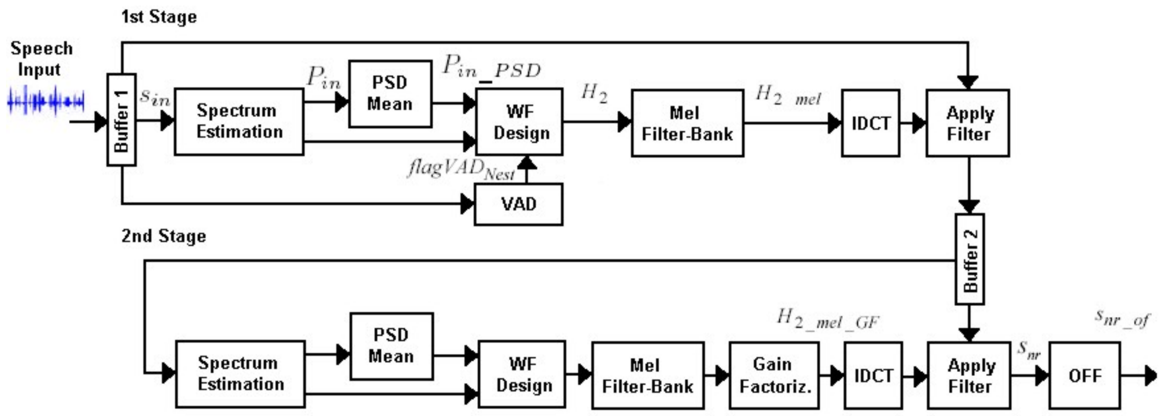


Figure 35: Noise Reduction Block

After framing the input signal, the linear spectrum of each frame is estimated in the Spectrum Estimation block. In PSD Mean block (Power Spectral Density), the signal spectrum is smoothed along the time (frame) index. Then, in the WF

Design block, frequency domain Wiener filter coefficients are calculated by using both the current frame spectrum estimation and the noise spectrum estimation. The noise spectrum is estimated from noise frames, which are detected by a voice activity detector (VADNest). Linear Wiener filter coefficients are further smoothed along the frequency axis by using a Mel Filter-Bank, resulting in a Mel-warped frequency domain Wiener filter. The impulse response of this Mel-warped Wiener filter is obtained by applying a Mel IDCT (Mel-warped Inverse Discrete Cosine Transform). Finally, the input signal of each stage is filtered in the Apply Filter block. At the end of Noise Reduction, the DC offset of the noise-reduced signal is removed in the OFF block.

The Wiener filter is the MSE-optimal stationary linear filter for a signal degraded by additive noise. Calculation of the Wiener filter requires the assumption that the signal and noise processes are second-order stationary. Given a degraded signal x(n), one takes the Discrete Fourier Transform (DFT) to obtain X(u). The original noise-free spectrum is estimated by taking the product of X(u) with the Wiener filter G(u). The inverse DFT is then used to obtain the signal estimate from its spectrum.

8.1.1 Buffering

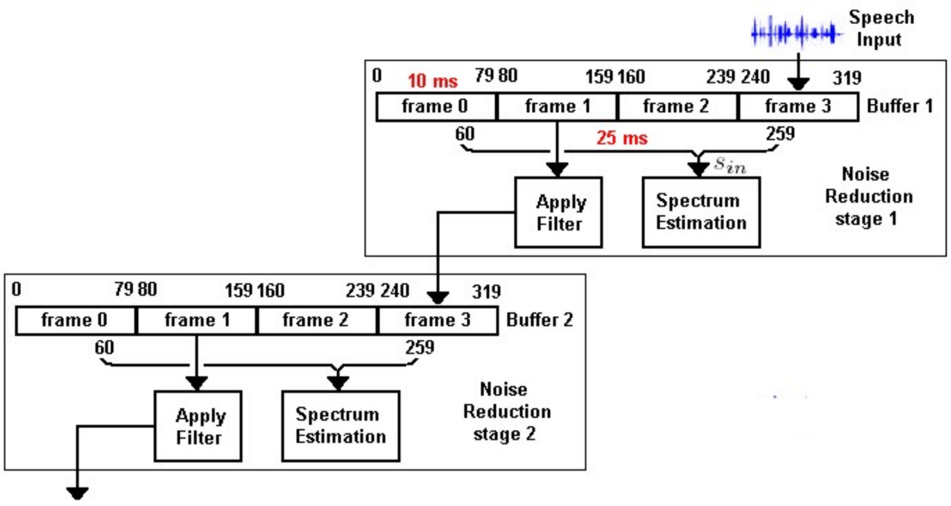


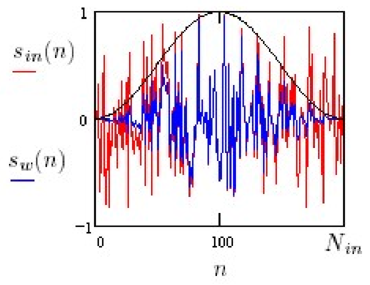
Figure 36: Buffering logic in the Noise Reduction block

The input of the noise reduction block is a 80-sample frame. A 4-frame (frame 0 to frame 3) buffer is used for each stage of the noise reduction. At each new input frame, the 2 buffers are shifted by one frame. The new input frame becomes frame 3 of the first buffer. Then the frame 1 (from position 80 to position 159 in the buffer) of the first buffer is denoised and this denoised frame becomes frame 3 of the second buffer. The frame 1 of the second buffer is denoised and this denoised frame is the output of the noise reduction block. Hence at each stage of the noise reduction block, there is a latency of 2 frames (20 ms). For each stage of the noise reduction block, the spectrum estimation is performed on the window which starts at position 60 and ends at position 259.

8.1.2 Spectrum estimation

Input signal is divided into overlapping frames $s_{in}(n)$ of N_{in} samples. With $f_s = 8kHz$ the **Frame Length** is $25ms$ (or $N_{in} = 200$ samples) and the **Frame Rate** is $10ms$ (or 80 samples). A Hanning window of length N_{in} is applied:

$$s_w(n) = s_{in}(n)\{0.5 - 0.5 \cdot cos(\tfrac{2\pi(n+0.5)}{N_{in}})\} \qquad 0 \le n \le N_{in} - 1 \tag{8.1.1}$$



Each frame of N_{in} samples is zero padded to make the frame size a power of two. In our case $N'_{in} = 256$. After that a $N_{FFT} = N'_{in}$ -point Fourier transform is applied.

$$X(k) = \sum_{n=0}^{N_{FFT}-1} s_w(n) e^{-\frac{2\pi jnk}{N_{FFT}}} \qquad 0 \le k \le N_{FFT} - 1 \tag{8.1.2}$$

and the power spectrum is calculated

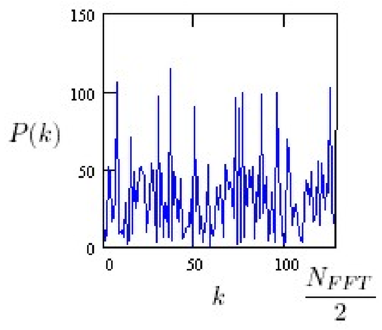
$$P(k) = |X(k)|^2 \qquad 0 \le k \le \frac{N_{FFT}}{2} \tag{8.1.3}$$

The power spectrum is smoothed like

$$P_{in}(k) = \frac{P(2k) + P(2k+1)}{2} \qquad 0 \le k < \frac{N_{FFT}}{4} \tag{8.1.4}$$

$$P_{in}(\frac{N_{FFT}}{4}) = P(\frac{N_{FFT}}{2}) \tag{8.1.5}$$

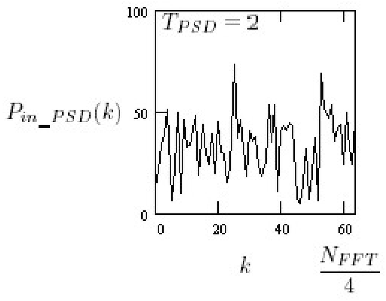
The length of the power spectrum is now reduced to $N_{SPEC} = \frac{N_{FFT}}{4} + 1$



8.1.3 Power spectral density mean

This module computes for each power spectrum $P_{in}(k)$ the mean over the last T_{PSD} frames.

$$P_{in\_PSD}(k) = \frac{1}{T_{PSD}} \sum_{i=0}^{T_{PSD}-1} P_{in}(k, t-i) \qquad 0 \leq k \leq N_{SPEC} - 1 \tag{8.1.6}$$



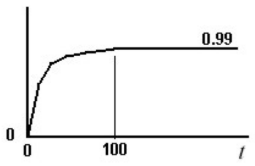
The chosen value for T_{PSD} is 2 and t is frame (time) index. PSD is 2 and t is frame (time) index. Note that throughout this document, we use frame index t only if it is necessary for explanation, otherwise current frame is referred.

8.1.4 Wiener filter design

A forgetting factor λ_{NSE} (used in the update of the noise spectrum estimate in first stage of noise reduction) is computed for each frame depending on the frame time index t:

where $NB\_FRAME\_THRESHOLD\_NSE = 100$ and $LAMBDA\_NSE = 0.99$.

```
if ($t < NB\_FRAME\_THRESHOLD\_NSE$)
    $lambdaNSE = 1 - 1/t$
else
    $lambdaNSE = LAMBDA\_NSE$
```



In first stage the noise spectrum estimate is updated according to the following equation, dependent on the $flagVADNest$ from the VADNest block:

$$P^{0.5}_{noise}(k, t_n) = max\{\lambda_{NSE} \cdot P^{0.5}_{noise}(k, t_n - 1) + (1 - \lambda_{NSE}) \cdot P^{0.5}_{in\_PSD}(k, t_n), EPS\} \tag{8.1.7}$$

$$P^{0.5}_{noise}(k, t) = P^{0.5}_{noise}(k, t_n) \tag{8.1.8}$$

$$P^{0.5}_{noise}(k, -1) = EPS \tag{8.1.9}$$

where $EPS = 10^{-10}$, t represents the current frame index and t_n represents the index of the last non-speech frame.

In the second stage the noise spectrum estimate is updated permanently according to the following equation:

```
if ($t < 11$)
        $lambdaNSE = 1 - 1/t$
        $P_{noise}(bin,t) = lambdaNSE \times P_{noise}(bin,t-1) + (1 - lambdaNSE) \times P_{in\_PSD}(bin,t)$
else
        $upDate = 0,9 + 0,1 \dfrac{P_{in\_PSD}(bin,t)}{P_{in\_PSD}(bin,t) + P_{noise}(bin,t-1)} \left(1 + \dfrac{1}{1 + 0,1 \dfrac{P_{in\_PSD}(bin,t)}{P_{noise}(bin,t-1)}}\right)$

        $P_{noise}(bin,t) = P_{noise}(bin,t-1) \times upDate$
if ($P^{1/2}_{noise}(bin,t) < EPS$)
        $P^{1/2}_{noise}(bin,t) = EPS$
```

Then the noiseless signal spectrum is estimated using a "decision-directed" approach:

$$P^{0.5}_{den}(k, t) = \beta P^{0.5}_{den3}(k, t - 1) + (1 - \beta)T[P^{0.5}_{in\_PSD}(k, t) - P^{0.5}_{noise}(k, t)] \tag{8.1.10}$$

$$P^{0.5}_{den}(k, -1) = 0 \tag{8.1.11}$$

where $T[u] \equiv u \cdot 1(u)$ and $\beta = 0.98$ Then the a priori SNR η is computed as:

$$\eta(k,t) = \frac{P_{den}(k,t)}{P_{noise}(k,t)} \tag{8.1.12}$$

The first estimation of the filter transfer function is obtained according to the following equation:

$$H(k,t) = \frac{\sqrt{\eta(k,t)}}{1+\sqrt{\eta(k,t)}} \tag{8.1.13}$$

This is used to improve the estimation of the noiseless signal spectrum:

$$P^{0.5}_{den2}(k,t) = H(k,t)P^{0.5}_{in\_PSD}(k,t) \tag{8.1.14}$$

Then an improved a priori SNR $\eta_2(k,t)$ is obtained:

$$\eta_2(k,t) = max\{\frac{P_{den2}(k,t)}{P_{noise}(k,t)}, \eta_{TH}\} \tag{8.1.15}$$

where $\eta_{TH} = 0.079432823$ (value corresponding to a SNR of $-22dB$)
The improved transfer function is then obtained according to the following equation:

$$H_2(k,t) = \frac{\sqrt{\eta_2(k,t)}}{1+\sqrt{\eta_2(k,t)}} \qquad 0 \leq k \leq N_{SPEC} - 1 \tag{8.1.16}$$

This is used to calculate the noiseless signal spectrum that will be used for the next frame in Equation 8.1.10:

$$P^{0.5}_{den3}(k,t) = H_2(k,t)P^{0.5}_{in\_PSD}(k,t) \tag{8.1.17}$$

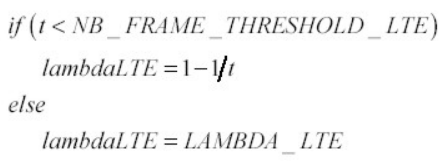
9 VAD for noise estimation (VADNest)

It is well known that noise reduction schemes are beneficial in ASR to reduce training-test mismatch due to noise. However, a significant mismatch may still remain after noise reduction, especially in the non-speech portions of the signals. When training-test mismatch occurs in non-speech signal portions, it is reasonable to expect that additional insertion errors are introduced.

To reduce the impact of this mismatch, two methods for discarding nonspeech acoustic vectors at recognition time are investigated: variable frame rate (VFR) processing and voice activity detection (VAD).

With VFR processing an observation vector is discarded if it does not differ much from the previous observation vector.

The VAD block determines the endpoints (beginning/end) of the utterances contained in the input waveform. To decide if the current frame is speech ($flagVAD_{Nest} = 1$) or not ($flagVAD_{Nest} = 0$) frame energy and mean energy are used. A forgetting factor λ_{LTE} (used in the update of the long-term energy) is computed for each frame depending on the frame time index t:



```
if (t < NB_FRAME_THRESHOLD_LTE)
    lambdaLTE = 1 − 1/t
else
    lambdaLTE = LAMBDA_LTE
```

where $NB\_FRAME\_THRESHOLD\_LTE = 10$ and $LAMBDA\_NSE = 0.97$. Then the logarithmic energy $frameEn$ of the M ($M = 80$) last samples of the input signal $s_{in}(n)$ is computed:

$$frameEn = 0.5 + \frac{16}{ln(2)} \ln(\frac{64 + \sum_{n=0}^{M-1} s_{in}(n)}{64}) \tag{9.0.18}$$

Then $frameEn$ is used in the update of $meanEn$

where $SNR\_THRESHOLD\_UPD\_LTE = 20$, $ENERGY\_FLOOR = 80$, $MIN\_FRAME = 10$ and $lambdaLTEhigherE = 0.99$. Then $frameEn$ and $meanEn$ are used to decide if the current frame is speech

where $SNR\_THRESHOLD\_VAD = 15$, $MIN\_SPEECH\_FRAME\_HANGOVER = 4$ and $HANGOVER = 15$. $nbSpeechFrame$, $meanEn$, $Nest\_flagVAD$ and $hangOver$ are initialized to 0. The frame time index t is initialised to 0 and is incremented each frame by 1 so that it equals 1 for the first frame processed.

```
if ( ((frameEn - meanEn) < SNR_THRESHOLD_UPD_LTE)
     OR
     (t < MIN_FRAME) )
    if ((frameEn < meanEN) OR (t < MIN_FRAME))
    then
        meanEn = meanEn + (1 - lambdaLTE) × (frameEn - meanEn)
    else
        meanEn = meanEn + (1 - lambdaLTEhigherE) × (frameEn - meanEn)
    if (meanEn < ENERGY_FLOOR) then meanEn = ENERGY_FLOOR


if (t > 4)
    if ((frameEn - meanEn) > SNR_THRESHOLD_VAD)
        flagVAD_Nest = 1
        nbSpeechFrame = nbSpeechFrame + 1
    else
        if (nbSpeechFrame > MIN_SPEECH_FRAME_HANGOVER)
                hangOver = HANGOVER
        nbSpeechFrame = 0
        if (hangOver != 0)
                hangOver = hangOver - 1
                flagVAD_Nest = 1
        else
                flagVAD_Nest = 0
```

9.1 Mel Filter-Bank

The linear-frequency Wiener filter coefficients $H_2(k,t)$ $(0 \leq k \leq N_{SPEC} - 1)$ have to be smoothed and transformed to the Mel-frequency scale. Mel-warped Wiener filter coefficients $H_{2\_mel}(k,t)$ are estimated by using triangular-shaped, half-overlapped frequency windows applied on $H_2(k,t)$. To obtain the central frequencies of FB bands in terms of FFT bin indices, ( ) k bincentr

The central frequencies of the filters are calculated from the Mel-function, in order to have an equidistant distribution of the bands in the Mel domain.

$$f_{c_i} = f_{lin}\{i\frac{f_{mel}(f_s/2)}{K_{FB}+1}\} \qquad 1 \leq i \leq K_{FB} \tag{9.1.1}$$

where $K_{FB} = 23$, and $f_s = 8000Hz$ is the sampling frequency. Additionally, two marginal FB bands with central frequencies $f_{c_0} = 0$ and $f_{c_{K_{FB}+1}} = \frac{f_s}{2}$ are added for purposes of following DCT transformation to the time domain; thus, in total we calculate $K_{FB} + 2 = 25$ Mel-warped Wiener filter coefficients. In terms of FFT index, the central frequencies of the filters correspond to

$$cbin_i = round(\frac{f_{c_i}}{f_s}(2N_{SPEC} - 1)) \tag{9.1.2}$$

The output of the Mel filter bank is the weighted sum of the magnitude spectrum values in each band.

$$H_{2\_mel}(i,t) = \frac{1}{\sum_{k=0}^{N_{SPEC}-1} W_i(k)} \sum_{k=0}^{N_{SPEC}-1} H_2(i,t)W_i(k) \qquad 0 \le i \le K_{FB} + 1 \tag{9.1.3}$$

with

$$W_i(k) = \frac{k - cbin_{i-1}}{cbin_i - cbin_{i-1}} \quad \text{for} \quad cbin_{i-1} + 1 \le k \le cbin_i \tag{9.1.4}$$

$$W_i(k) = 1 - \frac{k - cbin_i}{cbin_{i+1} - cbin_i} \quad \text{for} \quad cbin_i + 1 \le k \le cbin_{i+1} \tag{9.1.5}$$

$$W_i(k) = 0 \quad \text{otherwise} \tag{9.1.6}$$

9.2 Gain factorization

The intention of gain factorization is to apply more aggressive noise reduction of the second stage Wiener filter to purely noisy frames and less aggressive noise reduction to frames also containing speech.

to do

9.3 Mel IDCT

The time-domain impulse response of Wiener filter h_{WF} is computed from the Mel Wiener filter coefficients $H_{2\_mel}(i)$ (in the second stage, $H_{2\_mel\_GF}(i)$) by using Mel-warped inverse DCT:

$$h_{WF}(n) = \sum_{i=0}^{K_{FB}+1} H_{2\_mel}(i) IDCT_{mel}(i,n) \qquad 0 \le n \le K_{FB} + 1 \tag{9.3.1}$$

where $IDCT_{mel}(i,n)$, are Mel-warped inverse DCT basis computed as follows. First, central frequencies of each band are computed for $1 \le i \le K_{FB}$ like:

$$f_{c_i} = \frac{1}{\sum_{k=0}^{N_{SPEC}-1} W_i(k)} \sum_{k=0}^{N_{SPEC}-1} \frac{k f_s}{2(N_{SPEC}-1)} W_i(k) \qquad 0 \le i \le K_{FB} \tag{9.3.2}$$

$$f_{c_0} = 0 \tag{9.3.3}$$

$$f_{c_{K_{FB}+1}} = \frac{f_s}{2} \tag{9.3.4}$$

Then, Mel-warped inverse DCT basis are obtained as

$$IDCT_{mel}(i,n) = \cos(\frac{2\pi n f_{c_i}}{f_s}) df(i) \qquad 0 \le i, n \le K_{FB}+1 \tag{9.3.5}$$

where *df* is computed like

$$df(i) = \frac{f_{c_{i+1}} - f_{c_{i-1}}}{f_s} \qquad 1 \le i \le K_{FB} \tag{9.3.6}$$

$$df(0) = \frac{f_{c_1} - f_{c_0}}{f_s} \tag{9.3.7}$$

$$df(K_{FB}+1) = \frac{f_{c_{K_{FB}+1}} - f_{c_{K_{FB}}}}{f_s} \tag{9.3.8}$$

The impulse response of Wiener filter is mirrored as

$$h_{WF\_mirr}(n) = h_{WF}(n) \quad \text{if} \quad 0 \le n \le K_{FB}+1 \tag{9.3.9}$$

$$h_{WF\_mirr}(n) = h_{WF}(2(K_{FB}+1)+1-n) \quad \text{if} \quad K_{FB}+2 \le n \le 2(K_{FB}+1) \tag{9.3.10}$$

9.4 Apply Filter

The causal impulse response $h_{WF\_caus}(n)$ is obtained from $h_{WF\_mirr}(n)$ according to the following relations:

$$h_{WF\_caus}(n) = h_{WF}(n+K_{FB}+1) \quad \text{if} \quad 0 \le n \le K_{FB} \tag{9.4.1}$$

$$h_{WF\_caus}(n) = h_{WF}(n-K_{FB}-1) \quad \text{if} \quad K_{FB}+1 \le n \le 2(K_{FB}+1) \tag{9.4.2}$$

The causal impulse response is then truncated giving

$$h_{WF\_trunc}(n) = h_{WF\_caus}(n+K_{FB}+1-\tfrac{FL-1}{2}) \qquad 0 \le n \le FL-1 \tag{9.4.3}$$

where the filter length $FL = 17$. The truncated impulse response is weighted by a Hanning window:

$$h_{WF\_w}(n) = h_{WF\_trunc}(n)\{0.5 - 0.5 \cdot cos(\tfrac{2\pi(n+0.5)}{FL})\} \qquad 0 \le n \le FL - 1 \qquad (9.4.4)$$

Finally the input signal s_{in} is filtered with the filter impulse response to produce the noise-reduced signal:

$$s_{nr}(n) = \sum_{m=-\frac{FL-1}{2}}^{\frac{FL-1}{2}} h_{WF\_w}(m + \tfrac{FL-1}{2}) s_{in}(n-m) \qquad 0 \le n \le M - 1 \qquad (9.4.5)$$

where the frame shift interval M equals 80.

9.5 Offset compensation

To remove the DC offset, a notch filtering operation is applied to the noise-reduced signal like:

$$s_{nr\_of}(n) = s_{nr}(n) - s_{nr}(n-1) + (1 - \tfrac{1}{1024}) s_{nr\_of}(n-1) \qquad 0 \le n \le M - 1 \quad (9.5.1)$$

where $s_{nr}(-1)$ and $s_{nr\_of}(-1)$ correspond to the last samples of the previous frame and equal 0 for the first frame.

9.6 Waveform Processing

to do

9.7 Cepstrum Calculation

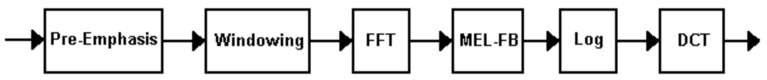


Figure 37: Block diagram of the Cepstrum calculation algorithm

9.8 Blind Equalization

Equalization is important for the receiver of communication systems to correctly recover the symbols sent by the transmitter, because the received signals may contain inter-symbol interference or ISI (noises, etc). Blind equalization algorithms attempt to recover the transmitted symbol sequence in the presence of ISI without resorting to training sequences as in the case of conventional equalizers. They do so by designing the inverse of the channel, usually in the form of the FIR filter

Blind Equalization is the process of compensating the filtering effect that occurs in signal recording.

Blind equalization (BE) refers to the problem of determining the system impulse response or the input signal when the system is unknown and the input is inaccessible.

9.9 Extension to 11 kHz and 16 kHz sampling frequencies

to do

10 Language Modeling

to do

11 Search

to do

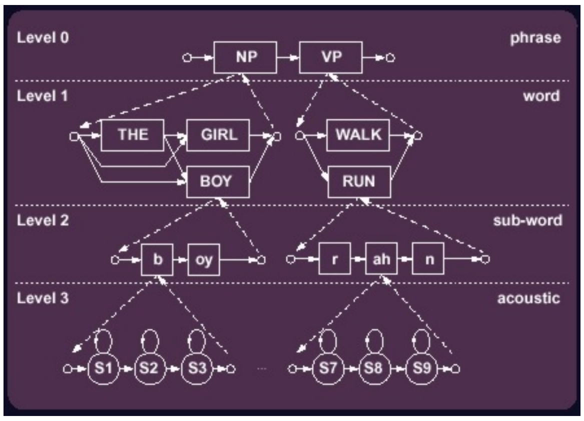


Figure 38: A hierarchical search space

12 Radio Voice Control for avionics

12.1 Functional and technical specifications

The characteristics of communication are the following:

1. Command&Control
2. 3 level vocabulary (max 20 words per level)
3. Important variability of speakers (potentially bad accents) with no availability for training
4. Push To Talk (PTT)
5. Language selectable among US English and Italian
6. Syntetic speech feedback (optional)
7. Noisy environment ($+110dB$ broadband-stationary-white noise. If colured give PSD)
8. Canceling microphone for aviation (give specifications)

12.2 Activity plan A - Outsourced Object Code

1. Define the language and the phone set
2. Define the dictionary and write the task grammar
3. Record background noise in the real operating environment
4. Model and simulate Background noise
5. Record valid and unvalid utterances in the real operating environment for training and testing. Alternatively, simulated noise can be added to clean utterances over a range of SNR
6. Choose a commercial sw library for an embedded processor e.g. Fonix VoiceIn SDK (1000$ + royalty fees) e.g. Sensory fluentSpeech SDK (1995$ + royalty fees) e.g. TI Embedded Speech Recognizer (TIESR)
7. hw development and prototyping of the target board
8. Buy processor-specific IDE e.g. TI TMDXKSP5912-M OMAP Starter Kit (OSK) Bundle with a Mistral Kickstart Program (4995$)
9. Vocabulary customization and sw integration of SDK with the top level application on the target hw
10. Testing
11. Implement Noise Suppression. Commercial solutions comprise Clarity Technologies Inc. CVC®-OMS (20000$ + royalty fees); Sensory Noise Suppression Program (<32000$)
12. sw integration of Noise Suppression
13. Testing
14. Final validation

Note that noise suppression is optional only in principle. SNR is critically important to ASR: a typical SNR for good accuracy is 24dB. Aircraft noise doesn't meet these constraints.

- Benefits: less developement time
- Drawbacks: no source available; not useful over a different platform or OS; less flexibility in tuning parameters (less performance); royalty fees
- Developement Time: 8 months

12.3 Activity plan B - Proprietary solution

1. Define the language and the phone set
2. Define the dictionary and write the task grammar
3. sketch ASR design (fix tecniques and constraints) e.g. LPC/FFT front-end, word/phone acoustic model, HMM/ANN arch.
4. Look for PC sw Utilities for displaying, editing, playing, recording, and labeling waveforms (e.g. Snack)
5. Look for PC sw Toolkit for acoustic-model training, grammar editing e.g. CSLU (contact dr. Piero Cosi at cosi@pd.istc.cnr.it for consultation); HTK
6. Look for free and commercial speech databases(e.g. TIMIT for english, APASCI for italian)
7. Look for free and commercial pronunciation dictionaries (italian and english) for training/validation/testing e.g. *http://www.elda.org/rubrique6.html*, BEEP
8. Record background noise in the real operating environment
9. Model and simulate Background noise
10. Record valid and unvalid utterances in the real operating environment for training and testing. Alternatively, simulated noise can be added to clean utterances over a range of SNR
11. Refine ASR design (fix algorithms and hw&sw components to be used) e.g. Select dsp (TMS320C6414TGLZ7 720MHz 5760MIPS 532BGA 107.32$)
12. Acoustic model training and testing with HTK or equivalent Toolkit
13. Look for free and commercial sw libraries e.g. Intel(R) Integrated Performance Primitives for Speech recognition
14. sw developement of proprietary FE on PC (based on ETSI Advanced FE specification); Acoustic model training and testing with proprietary FE and HTK RE; Porting of FE on a commercial dsp board.

Alternatively, Porting of HTK FE on dsp board; interfacing PC with dsp board; Acoustic model training and testing with FE running on dsp board and HTK RE running on PC.

15. Porting of HTK RE on dsp board
16. Testing and performance evaluation on dsp board
17. hw development of the target board
18. sw integration with the top level application on the target board
19. Testing and performance evaluation on target board
20. Final validation

- Benefits: With full programmability of the system the customer is able to upgrade its speech algorithms as these evolve, to modify vocabulary and language support as needed, to port code to other platforms and OSs, to finely adjust parameters to achieve maximum performance, to freely choose dsp processor and use full processing power for maximum performance.
- Drawbacks: longer developement time
- Developement Time: 18 months

13 Fixed Point Versus Floating Point

A floating-point processor makes coefficient representation a nonissue. The elimination of numerical concerns can significantly reduce development time, but a fixed-point processor performs the HMM algorithm equally well and can significantly reduce system cost. However, executing a fixed-point system requires a thorough understanding of the numerical issues.

13.1 Fixed Point Qn.m format

A fixed-point DSP represents a fractional number by using a fixed decimal point. In classifying the different ranges of the decimal number, a **Qn.m**-format is used. Different Q-formats represent different locations of the decimal point, and thus the integer range. Figure below shows a format of a Q12.4 number (or Q15, Q15/16).

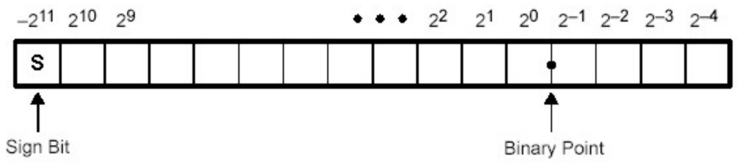


Figure 39: Example of a Q12.4 notation

Note that the most-significant bit is used also as the sign bit. The maximum positive number is obtained when the sign bit is 0 and the rest are 1s (7FFFH). For a Q12.4 this number is $2^{10}+2^9+2^8+2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0+2^{-1}+2^{-2}+2^{-3}+2^{-4} = 2^{11} - 2^{-4} = 2047.9375$. In formula

$$w_{max} = 2^{n-1} - 2^{-m} \tag{13.1.1}$$

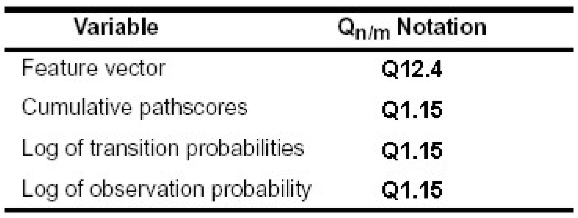
The negative limit is obtained when the sign bit is 1 and the rest of the bits are 0s (8000H). In formula

$$w_{min} = -2^{n-1} \tag{13.1.2}$$

For a Q12.4 this number is $-2^{11} = -2048$. In the case of a Q1.15 format these numbers become 0.9999694 and -1. Therefore, we can increase the decimal range by shifting the decimal point to the right. In Q2.14, the decimal range has been increased to 1.9999694 to -2. However, the increase in range means a decrease in precision. Fractions require scaling up and rounding to integers such that the targeted accuracy can be achieved. The following table shows several examples of Qn/m notation, as used in the implementation of a typical fixed-point HMM recognizer.

As noted, a fixed-point DSP can reduce system cost. However, in SI systems, size of vocabulary is all-important. SI recognition need a floating-point system, not only for its ability to represent values in floating-point format, but also for its memory reach. The 'C5x can access only 64K words of linear data memory, while the 'C3x can access 16M words, and the 'C4x 4G words.

Typically a fixed-point implementation uses many transformations that allow table look-ups instead of on the fly calculations. Of course, using these methods requires a higher amount of memory resources. Table 4.1 gives an overview of look-up tables applied in the code.



Variable	$Q_{n/m}$ Notation
Feature vector	**Q12.4**
Cumulative pathscores	**Q1.15**
Log of transition probabilities	**Q1.15**
Log of observation probability	**Q1.15**

Figure 40: Fixed point number format in a typical HMM implementation

Further, the generic functions like "square root" and "natural logarithm" are implemented in fixed-point whereby different algorithms have to be tested regarding best performances in terms of speed and accuracy. For example, instead of the "natural logarithm", the so-called "bitlog" function can be applied. This function builds on $log_2(\bullet)$, which can be computed much faster than $\ln(\bullet)$ and $log_{10}(\bullet)$. The difference between $log_2(\bullet)$ and $\ln(\bullet)$ is a constant factor. Assuming x is a scaled integer version of the floating-point value $w > 0$:

$$x = \lfloor w \cdot 2^m + sign(w) \cdot 0.5 \rfloor \tag{13.1.3}$$

If $w < 0$ we have

$$x = 2^{n+m} + \lfloor w \cdot 2^m + 0.5 \rfloor \tag{13.1.4}$$

Example: Find x in Q3.5 format ($n = 3; m = 5$) for the decimal number $w = 2.625$.

$$x = \lfloor 2.625 \cdot 2^5 + 0.5 \rfloor = \lfloor 84.5 \rfloor = 84 \rightarrow 01010100 \tag{13.1.5}$$

Example: Find x in Q3.5 format ($n = 3; m = 5$) for the decimal number $w = -2.625$.

$$x = 2^8 + \lfloor -2.625 \cdot 2^5 + 0.5 \rfloor = 256 + \lfloor -83.5 \rfloor = 256 - 84 = 172 \rightarrow 10101100 \tag{13.1.6}$$

$$y = bitlog(x) = 8 \cdot [log_2(x) - 1] = 8 \cdot [log_2(w) + m - 1] \tag{13.1.7}$$

$$log_2(w) = \frac{y}{8} + 1 - m \tag{13.1.8}$$

$$ln(w) = \frac{log_2(w)}{log_2(e)} = \frac{1}{log_2(e)}[\frac{y}{8} + 1 - m] \tag{13.1.9}$$

The square root routine can be bases on an iterative algorithm combined with a small look-up table.

Typically, a floating-point Front-End implementation performs 5% better, compared to the fixed-point version.

13.2 Floating Point format

Floating-point (FP) operands are classified as **single precision** (SP) and **double precision** (DP). Single-precision FP values are 32-bit. Double-precision FP values are 64-bit.

IEEE floating-point numbers consist of normal numbers, denormalized numbers, NaNs (not a number), and infinity numbers. Denormalized numbers are nonzero numbers that are smaller than the smallest nonzero normal number. Infinity is a value that represents an infinite FP number. NaN values represent results for invalid operations, such as (+infinity + (-infinity)).

Normal SP values are always accurate to at least 6 decimal places, sometimes up to 9 decimal places. Normal DP values are always accurate to at least 15 decimal places, sometimes up to 17 decimal places.

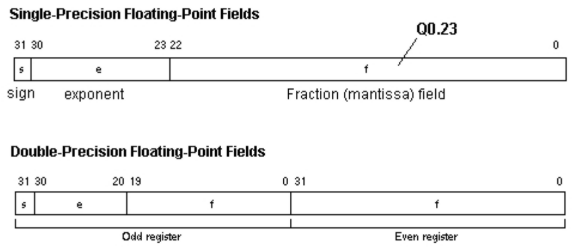

Single-Precision Floating-Point Fields
Q0.23
31 30
23 22
0
s
e
f
sign
exponent
Fraction (mantissa) field
Double-Precision Floating-Point Fields
31 30
20 19
0 31
0
s
e
f
f
Odd register
Even register

Figure 41: Floating point format

14 ASR in the aircraft - Implementation issues

14.1 Noise level in the cockpit

In the case where the microphone is very close to the talker's mouth, the SNR will always be quite high. But the environment is very noisy, so that noise canceling microphones must be used.

Sound levels of cockpit noise routinely exceed 100 dB and often reach 120 dB. The development of turbine-powered aircraft aggravated these effects, especially for f¿2kHz, where human speech is produced. The European Union recently set an ambient noise standard of 80 dB for passenger airliners.

It is very important to evaluate ASR systems in the context of real acoustical environments with real noise sources, rather than using speech that is recorded in a quiet environment into which additive noise is artificially injected.

Ideally, a detailed knowledge of the noise spectrum of the specific aircraft would provide the best matching of ASR system to aircraft. Unfortunately, this level of engineering information is rarely available.

14.2 Noise level in a car

The following picture shows the noise spectrum inside BMW and Volvo cars.

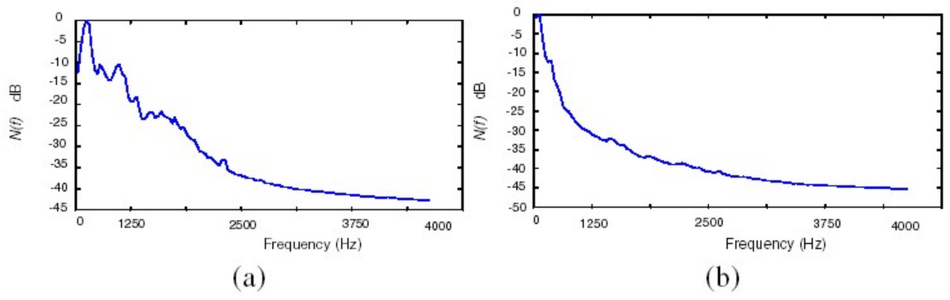


Figure 42: Power spectra of car noise in (a) a BMW at 70 mph, and (b) a Volvo at 70 mph.

The noise in a car is nonstationary, and varied, and may include the following sources: (a) quasi-periodic noise from the car engine and the revolving mechanical parts of the car; (b) noise from the surface contact of wheels and the road surface; (c) noise from the air flow into the car through the air ducts, windows, sunroof, etc; (d) noise from passing/overtaking vehicles. The characteristic of car noise varies with the speed, the road surface conditions, the weather, and the environment within the car.

The simplest method for noise modelling, often used in current practice, is to estimate the noise statistics from the signal-inactive periods.

14.3 Aviation microphones

Aviation microphones are noise canceling mics, built with two microphone cartridges aligned back to back with a common output membrane. It has openings for sound to enter on both the front and back of the microphone body. By allowing the same sound to enter both sides of the mic, the microphone cancels the cockpit noise effectively. When you speak into the mic, your voice mostly enters one side of the mic, causing membrane to resonate, which creates the output signal (your voice) without much of the ambient noise (cockpit sound). The output SNR is quite good. This technology works well until the ambient noise level exceeds 97dB of sound pressure. This sound pressure level is exceeded easily by the noise of a large engine, insufficient sound insulation, an open cockpit, or an open ventilator, doors or windows. When the cockpit noise entering the microphone exceeds 97dB, it causes the membrane to resonate harmonically to the cockpit noise.This resonation makes a noise that sounds like the ”shhhh” of wind. The harmonically vibrating membrane creates this noise, much like drawing your finger over a phonograph needle (remember those?) creates a ”shhhh” noise.

14.4 Microphone adjustment

If there is any one item which will make the intercom play either perfectly or poorly, it is the mechanical adjustment of the microphone element. The microphone must be positioned DIRECTLY in front of the user's mouth – NOT off to one side, NOT at the corner, but right square in the middle of the mouth. As to the distance between lips and mike – if you can't pucker your lips and touch the case, it isn't close enough. All this comes about because of the use of noise-cancelling microphones. The only correct way to use a noise-cancelling microphone is to talk DIRECTLY into the FRONT of the microphone element.

14.5 The task grammar

Per specificare una grammatica si ricorre a un insieme di regole sintattiche che descrivono tutte e sole le sequenze di parole riconoscibili. Per denotare sotto-sequenze si utilizzano delle variabili. Le convenzioni sintattiche sono:

- Una grammatica è un insieme di regole che rappresentano definizioni di variabili, dove la regola finale rappresenta l'intera grammatica

- Una regola ha forma: $\$X = ab...c$, dove $\$X$ è la variabile. Le variabili sono denotate dal prefisso \$.
- Ciascuna regola deve terminare con ";"
- Le variabili vanno definite prima di essere referenziate
- Le parentesi () delimitano oggetti raggrupati
- Le parentesi [] delimitano items opzionali
- Le parentesi `<>` indicano una o più ripetizioni
- Le parentesi `{}` indicano una o più ripetizioni (come "`[<>]`")
- Il simbolo `|` (pipe) separa item alternativi
- Il suffisso %% indica che il riconoscimento della parola non deve essere segnalato in uscita

Typically, an utterance is delimited by silence or often a combination of silence and/or noise. Thus it is common to allow for the "sil" and "any" phonemes at the beginning and end of our grammar specification.

"sil" represents silence (up to, say 5000 ms) and "any" represents the any model (a form of wildcard) that will map to any sound (up to, say, 5000 ms). The "any" symbol is used to implement word-spotting.

14.6 The vocabulary

The vocabulary (or dictionary) is a list the words the system can recognize.

The two main perspectives of the methodology for collecting and validating spoken commands are:

1. a user requires commands to be both intuitive and easy to remember
2. a speech recognition system requires commands to be easily discriminable

ETSI specification ES202-076 V1.1.2 (2002-11) explains various methodologies.

The first step in building a dictionary is to create a sorted list of the required words. In the radio control task discussed here, it is quite easy to create a list of required words by hand.

Example: ADD BACK BLOCK CALL CANCEL CHANNEL CREATE DELETE DISABLE DO EMERGENCY ENABLE ENTER FORWARD FREQUENCY FROM HOLD LAST MESSAGE NO NOT NUMBER PLAY PROGRAM RECORD SEND STOP TO TRANSFER WAIT YES ZERO OH ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE POINT MEGAHERTZ

15 Tools

15.1 The HTK Toolkit

HTK (*Hidden Markov Model Toolkit*) is a toolkit for research in automatic speech recognition and has been used in many commercial and academic research groups for many years. HTK was originally developed at the Cambridge University Engineering Department (CUED). In 1995 the development of HTK was fully transferred to the Entropic Cambridge Research Laboratory Ltd. In 1999 Microsoft bought Entropic. While Microsoft retains the copyright to the existing HTK code, everybody is encouraged to make changes to the source code and contribute them for inclusion in the current release.

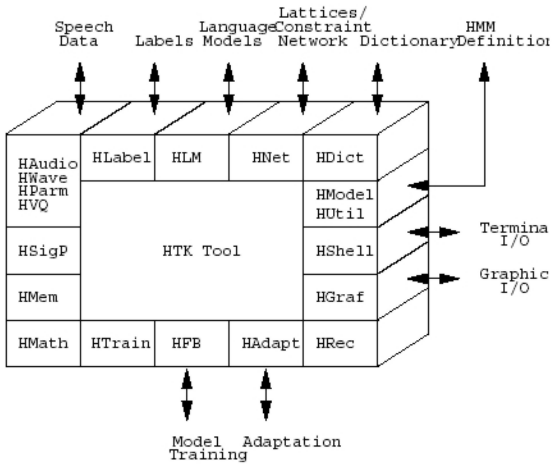


Figure 43: Software architecture of HTK

The picture above illustrates the software structure of a typical HTK tool and shows its input/output interfaces. User input/output and interaction with the OS is controlled by the library module **HShell**. All memory management is controlled by **HMem**. Math support is provided by **HMath**. The signal processing operations needed for speech analysis are in **HSigP**. Each of the file types required by HTK has a dedicated interface module. **HLabel** provides the interface for label files, **HLM** for language model files, **HNet** for networks and lattices, **HDict** for dictionaries, **HVQ** for VQ codebooks definitions and **HModel** for HMM definitions.

All speech input and output at the waveform level is via **HWave** and at the parameterised level via **HParm**. HWave and HLabel support multiple file formats allowing data to be imported from other systems. Direct audio input is supported by **HAudio** and simple interactive graphics is provided by **HGraf**. **HUtil** provides a number of utility routines for manipulating HMMs while **HTrain** and **HFB** contain support for the various HTK training tools. **HAdapt** provides support for the various HTK adaptation tools. Finally, **HRec** contains the main recognition processing functions.

HSLab utility can be used both to record the training and testing speech and to manually annotate it with any required transcriptions.

HTK tools are designed to run with a traditional command-line style interface. Each tool has a number of required arguments plus optional arguments (a single letter name prefixed by a minus sign and followed by the option value)
Example:

```
HFoo -T 1 -f 34.3 -a -s myfile file1 file2
```

This tool has two main arguments called *file1* and *file2* plus four optional arguments. Thus, the value of the -f option is a real number, the value of the -T option is an integer number and the value of the -s option is a string. The -a option has no following value and it is used as a simple flag. Options whose names are a capital letter have the same meaning across all tools. For example, the -T option is always used to control the trace output of a HTK tool. In addition to command line arguments, the operation of a tool can be controlled by parameters stored in a configuration file.

15.1.1 Preparazione dei dati

In order to build a set of HMMs, a set of speech data files and their associated transcriptions are required. Very often speech data will be obtained from database archives, typically on CD-ROMs, or can be recorded and manually annotated using **HSLab**. Before the data can be recorded, a phone set must be defined, a dictionary must be constructed to cover both training and testing and a task grammar must be defined. The tool **HCopy** is used for data parametrization. As the name suggests, HCopy is used to copy and encoding one or more source files to an output file. **HList** can be used to check the contents of any speech file. Typically the labels used in the original source transcriptions will not be exactly as required, for example, because of differences in the phone sets used. **HLEd** is a script-driven label editor which is designed to make the required transformations to label files. **HLStats** display statistics on label files. **HQuant** can be used to build a VQ codebook in preparation for building discrete probability HMM system.

The second step of system building is to define the topology required for each HMM by writing a prototype definition. HTK allows HMMs to be built with any desired topology. The purpose of the prototype definition is only to specify the overall characteristics and topology of the HMM. The actual parameters will be computed later by the training tools. Sensible values for the transition probabilities must be given but the training process is very insensitive to these. An acceptable and simple

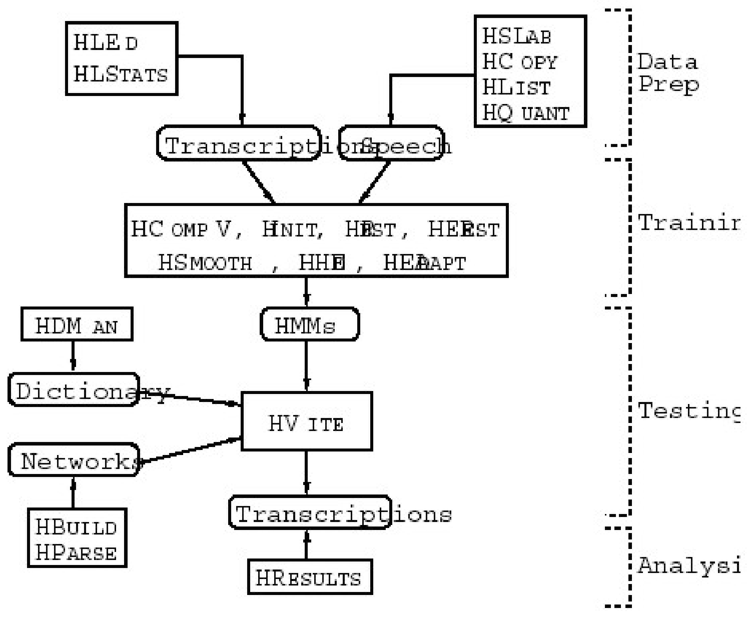


Figure 44: Main processing stages of HTK

strategy for choosing these probabilities is to make all of the transitions out of any state equally likely.

15.1.2 Addestramento

Inizialmente deve essere creato un insieme di modelli fonetici usando file di parlato già segmentati (cioè in cui siano marcati i limiti tra i singoli fonemi). Le HMM vengono generate una alla volta: **HInit** legge tutti i dati in ingresso ed estrapola tutte le occorrenze del fonema richiesto, quindi calcola i valori iniziali dei parametri del modello. Nel primo ciclo i dati sono uniformemente segmentati cioè porzioni uniformi del fonema vengono associate agli stati del modello. Nei successivi cicli si sfrutta l'allineamento di Viterbi. I parametri calcolati da HInit sono nuovamente stimati da **HRest**. Infine si ripete la procedura utilizzando le formule di Baum-Welch.

Una volta creato un insieme iniziale di modelli, **HERest** esegue l'*embedded training* su tutto l'insieme di addestramento vero e proprio. Per ciascuna sequenza di parlato di addestramento i corrispondenti modelli fonetici vengono concatenati. Su ciascuna HMM si applica l'algoritmo *forward-backward* per accumulare le probabilità di occupazione di stato, le medie, le varianze, ecc. Quando tutte le sequenze di addestramento sono state processate, si calcolano le nuove stime dei parametri delle HMMs. HERest, il cuore del tool di addestramento di HTK, può gestire grandi database e può girare in

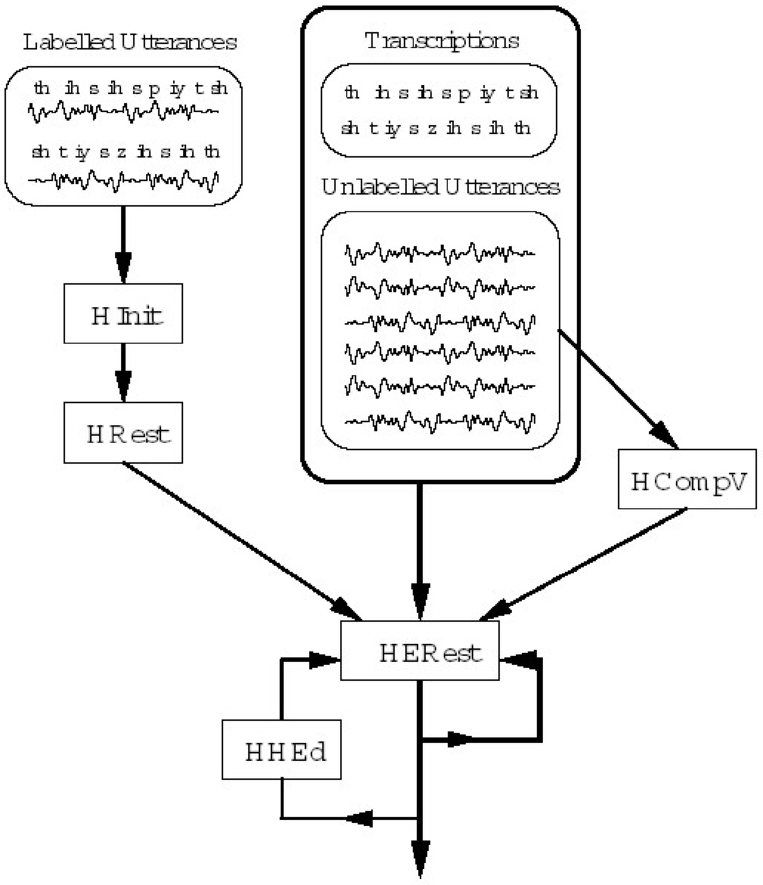


Figure 45: Phoneme model training stages of HTK

parallelo su più macchine connesse in rete.

La filosofia di HTK è quella di raffinare il sistema in modo incrementale: si parte con un insieme semplice di modelli fonetici indipendenti dal contesto e con distribuzioni a singola gaussiana, per poi raffinarli iterativamente con l'aggiunta della dipendenza dal contesto e con l'uso di distribuzioni a mistura di gaussiane multiple. Questo processo è mediato dal tool **HHed**, un editor per la definizione di HMMs.

Per aumentare l'accuratezza del sistema su specifici *speakers* i tools HERest e HVite possono adattare le HMMs alle loro caratteristiche vocali, usando pochi dati di addestramento.

Il vero problema nella realizzazione di sistemi HMM *context-dependent* è sempre l'insufficienza di dati. Più il sistema è complesso, maggiore è la quantità di dati richiesti per ottenere stime robuste dei suoi parametri. Per bilanciare la complessità in funzione dei dati disponibili esistono alcune tecniche. Per i sistemi a densità continua c'è la possibilità di unificare i parametri. Ciò permette di raggruppare insieme i dati in modo tale da ottenere stime più robuste per i parametri comuni. Per i sistemi discreti e quelli con misture legate **HSmooth** è un tool per lo *smoothing* delle distribuzioni.

15.1.3 Riconoscimento

Il tool preposto al riconoscimento è **HVite**, che utilizza la tecnica del *token-passing* per l'algoritmo di Viterbi. HVite riceve in ingresso una rete che descrive le sequenze di parole permesse, un vocabolario che definisce la pronuncia di ciascuna parola (cioè la sua trascrizione fonetica) e un insieme di modelli (HMMs). HVite converte la rete di parole in una rete di fonemi, quindi associa a ciascun fonema la sua HMM. Il riconosciemnto può essere effettuato su una lista di file audio preregistrati, oppure direttamente da una scheda di acquisizione. HVite supporta trifoni cross-word e i token multipli per generare reticoli contenenti ipotesi multiple.

Tipicamente le reti di parole sono semplici loop in cui ogni parola può essere seguita da ogni altra parola, oppure sono grafi diretti che rappresentano una grammatica a stati finiti. Nel primo caso alle transizioni da parola a parola vengono associate probabilità bigram. Per memorizzare le reti di parole si utilizza il formato reticolare standard di HTK. Si tratta di un formato testuale. HTK mette anche a disposizione **HBuild**, per creare delle sotto reti.

In alternativa si può usare una notazione a più alto livello, basata sulla *Extended Backus Naur Form* (EBNF). Il tool **HParse** converte questa notazione nella rete di parole equivalente. A scopo di verifica, il tool **HSGen** serve a produrre delle sequenze di parole in base alla rete che si è definita. HSGen calcola anche la perplessità del task. Per la creazione del vocabolario può essere necessario unire molte sorgenti, effettuando una serie di trasformazioni su ciascuna di esse. A questo scopo si può usare il tool **HDMan**.

15.1.4 Analisi delle prestazioni

Per valutare le performance del sistema realizzato si applica il sistema su sequenze di parlato di test e si controlla la trascrizione prodotta dal sistema. **HResults** è un tool che utilizza la programmazione dinamica per eseguire i confronti. Esso allinea le due trascrizioni e conta le sostituzioni, le delezioni e le inserzioni. Il formato di uscita è conforme al formato del US National Institute of Standards and Technology (NIST). Per le applicazioni word-spotting HResults genera anche i punteggi delle figure di merito (FOM)e la Receiver Operating Curve (ROC).

15.1.5 Definizione della grammatica

Applicando le regole sintattiche standard per la descrizione di una grammatica, abbiamo
$DIGIT = ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE | OH | ZERO;
$NAME = [JOOP] JANSEN | [JULIAN] ODELL | [PHIL] WOODLAND | [STEVE] YOUNG;
(SENT-START (DIAL <$DIGIT> | (PHONE | CALL) $NAME) SENT-END)

In questo esempio la pronuncia di SENT-START e SENT-END è il fonema "sil". La grammatica completa può essere visualizzata in forma di rete, come mostrato nella figura seguente

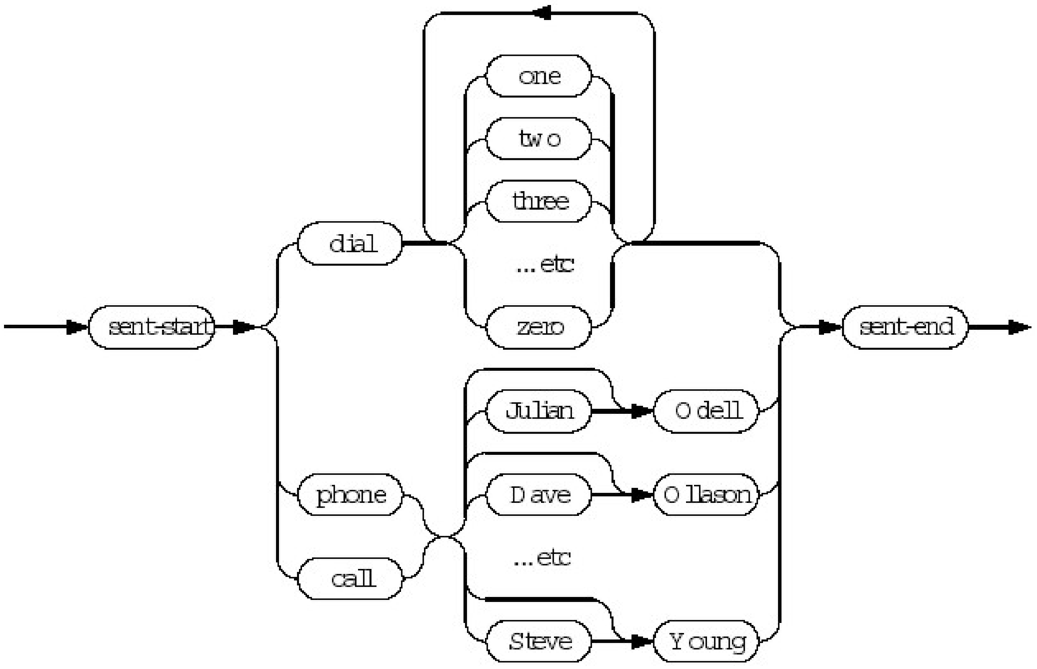


Figure 46: Word-network for a simple voice dial task grammar

Il tool HTK richiede che la grammatica così definita sia tradotta in un'altra notazione, la **HTK Standard Lattice Format** (SLF), dove tutte le parole e le transizioni parola-parola sono elencate esplicitamente. Il tool **HParse** trasforma questa notazione nella rete. Assumendo che il file *gram* contenga la grammatica in SLF, digitando

```
HParse gram wdnet
```

viene creata la rete equivalente nel file *wdnet.*

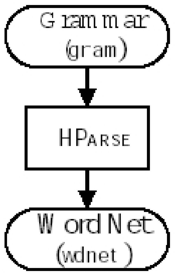


Figure 47: HTK - word-network file generation from the task grammar

15.1.6 Il dizionario

In HTK, il formato della generica linea del dizionario è

```
WORD [outsym] p1 p2 p3 ...
```

che significa che la parola "WORD" è pronunciata come sequenza di fonemi "p1-p2-p3-...". Tra parentesi quadre figura la stringa che il sistema deve emettere in uscita ogni qualvolta riconosce WORD. Se la stringa è omessa allora la parola stessa deve essere emessa, mentre le parentesi vuote indicano che non deve essere emessa alcuna stringa. Le prime linee di un dizionario siffatto saranno del tipo

```
A              ah sp
A              ax sp
A              ey sp
CALL           k ao l sp
DIAL           d ay ax l sp
EIGHT          ey t sp
PHONE          f ow n sp
SENT-END []    sil
SENT-START []  sil
SEVEN          s eh v n sp
TO             t ax sp
TO             t uw sp
ZERO           z ia r ow sp
```

Si noti che A e TO hanno pronuncia multipla, mentre SENT-START e SENT-END hanno il modello sil come pronuncia e non emettono simboli in uscita.

Per costruire un dizionario occorre dapprima una lista ordinata delle parole richieste. Affinchè i modelli acustici siano robusti è necessario addestrarli su un buon numero di frasi contenenti molte parole, e foneticamente bilanciate. Per questi motivi *i dati di addestramento consisteranno di frasi in inglese completamente scorrelate rispetto al compito richiesto* (p.es. comporre un numero di telefono). Ad esempio, un database di partenza come TIMIT potrebbe darci una lista di frasi (*prompts*) del tipo

S0001 ONE VALIDATED ACTS OF SCHOOL DISTRICTS
S0002 TWO OTHER CASES ALSO WERE UNDER ADVISEMENT
S0003 BOTH FIGURES WOULD GO HIGHER IN LATER YEARS
S0004 THIS IS NOT A PROGRAM OF SOCIALIZED MEDICINE
etc

La lista di addestramento desiderata (*wlist*) può essere estratta automaticamente. Come prima cosa i database a disposizione possono richiedere un certo numero di trasformazioni di editing (p.es. trasformare ogni spazio in un a capo). Le utility UNIX **sort** e **uniq** permettono infine di generare una lista di parole ordinate alfabeticamente, una per linea. A questo scopo si può usare lo script **prompts2wlist** nella directory *HTKTutorial*.

Il dizionario deve essere generato dal tool HDMan a partire da un dizionario generico standard, come **BEEP** (*British English Pronouncing dictionary*). Questo dizionario può essere usato così com'è, a meno di eliminare gli stress marks e aggiungere una piccola pausa sp (short-pause) alla fine di ogni pronuncia. Nel caso in cui il dizionario già contenga un marker per la pausa (sil), il comando MP fonderà sil e sp in un unico sil. Le trasformazioni descritte possono essere applicate usando HDMan e uno script (in *global.ded*) con i tre comandi

AS sp
RS cmu
MP sil sil sp

Il comando

```
HDMan -m -w wlist -n monophones1 -l dlog dict beep names
```

crea il dizionario desiderato (*dict*) estraendo dai dizionari generici *beep* e *names* la pronuncia delle soloe parole presenti in *wlist*. Nel nostro esempio *names* è un file

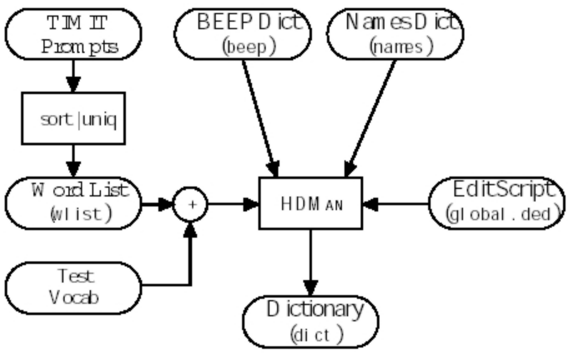


Figure 48: HTK - dictionary generation from the a list of word and a pronunciation dictionaries

editato a mano, con le pronuncie dei nomi propri utilizzati nella grammatica. L'opzione *-l* produce un file di log (*dlog*). HDMan può anche generare un file con la lista dei soli fonemi utilizzati (*monophones1*).

15.1.7 Registrazione dei dati

I dati di training e di test possono essere registrati e labellati usando il tool HSLab. Nel nostro esempio HSLab viene usato solo per la registrazione, dato che le labels già ci sono. HSLab è invocato digitando

```
HSLab noname
```

fd
to do

15.2 The CSLU Toolkit

The CSLU Toolkit is used for research activities related to speech recognition at CSLU. This toolkit allows the development of speech recognizers based on the standard HMM as well as HMM/ANN framework. The Toolkit is written at two levels: the Tcl script level, and the C level. The Tcl level allows for easy implementation and modification of high-level procedures such as speech recognition, file selection, and recognizer evaluation. Computation-intensive procedures, such as wave I/O, neural network training and classification, and Viterbi search, are implemented at the C level. An intermediate level allows the C-level routines to be called from the Tcl-level scripts. The CSLU Toolkit includes several tutorials for developing speech-recognition systems for custom or general-purpose vocabularies, as well as tutorials that explain how speech recognition is done using the HMM/ANN framework, how to read spectrograms, and how robust parsing can be done. Recognizers that are developed using the Toolkit can be quickly plugged in to the Toolkit's " Rapid Application Developer", and used in real-time in a variety of interactive applications. The Toolkit also includes the SpeechView program for displaying, editing, playing, recording, and annotating waveforms and waveform information. We have found this tool extremely useful in the debugging process, as frame-based outputs from recognition scripts can be displayed in SpeechView in synchrony with the waveform and spectrogram.

Collaboration with **Piero Cosi** at C.N.R.'s Institute of Phonetics and Dialectology (IFD) dates back several years, and we have been collaborating both electronically and, with help from C.N.R. and I.F.D., face-to-face. We have investigated the relative performance of standard HMMs and HMM/ANN hybrid systems, techniques for improving HMM/ANN performance, especially for connected digit recognition, and the development of Italian continuous-digit and general-purpose recognizers.

16 Software Utility resources

16.1 Text To Phoneme programs

- *ftp://shark.cse.fau.edu/pub/src/phon.tar.Z*
- *ftp://ftp.doc.ic.ac.uk/packages/unix-c/utils/phoneme.c.gz*
- *ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/synthesis/english2phoneme.tar.gz*

17 Algorithm resources

17.1 FFT algorithm resources

Comprehensive list of FFT software Links to over 65 different pieces of one-dimensional FFT code. *http://tjev.tel.etf.hr/josip/DSP/fft.html*

FFT Software including optimised fft routines and mixed-radix algorithms *ftp://usc.edu/pub/C-numanal/fft-stuff.tar.gz*; *ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/ana* *stuff.tar.gz*

C-source for a very fast arbitrary N FFT routine The C-source is ShareWare. Jens J. Nielsen: jnielsen@internet.dk Available from *ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech* *ftp://ftp.coast.net/simtel/msdos/c/mixfft03.zip*

FFTW FFTW is a C subroutine library for computing the FFT in one or more dimensions. It is not limited to sizes that are powers of two, and includes real-complex and parallel transforms. Also on the FFTW web site are benchmarks comparing the performance and accuracy of many public-domain FFT implementations on a variety of platforms, as well as links to other sources of FFT code and information. Available from *http://theory.lcs.mit.edu/ fftw* Developed by Matteo Frigo and Steven G. Johnson: *fftw@theory.lcs.mit.edu*

18 Free software resources

XVoice *http://www.compapp.dcu.ie/ tdoris/Xvoice/ http://www.zachary.com/creemer/xvoice.html*

CVoiceControl/kVoiceControl *http://www.kiecza.de/daniel/linux/index.html*

GVoice *http://www.cse.ogi.edu/ omega/gnome/gvoice/*

ISIP *http://www.isip.msstate.edu/project/speech/*

CMU Sphinx *http://www.speech.cs.cmu.edu/sphinx/Sphinx.html*

Myers' Hidden Markov Model Software *http://www.itl.atr.co.jp/comp.speech/Section6/Recogn*

IBM ViaVoice *http://www-306.ibm.com/software/voice/viavoice*

JULIUS *http://julius.sourceforge.jp/en/julius.html*

HTK *http://htk.eng.cam.ac.uk*. My user ID: **oscar**; password: **=gJao7uR**

19 Some other information

19.1 The Baum Welch algorithm

The Baum-Welch algorithm is used to find the unknown parameters of a hidden Markov model (HMM). It is also known as the forward-backward algorithm. The Baum-Welch algorithm is an EM (expectation-maximization) algorithm. It can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only emissions as training data.

The algorithm has two steps: (1) calculating the forward probability and the backward probability for each HMM state; (2) on the basis of this, determining the frequency of the transition-emission pair values and dividing it by the probability of the entire string. This amounts to calculating the expected count of the particular transition-emission pair. Each time a particular transition is found, the value of the quotient of the transition divided by the probability of the entire string goes up, and this value can then be made the new value of the transition.

19.2 The Viterbi algorithm

The Viterbi algorithm, named after its developer Andrew Viterbi, is a dynamic programming algorithm for finding the most likely sequence of hidden states – known as the Viterbi path – that result in a sequence of observed events, especially in the context of hidden Markov models. The forward algorithm is a closely related algorithm for computing the probability of a sequence of observed events. Its a subset of a wider topic known as information theory. The Viterbi algorithm was originally conceived as an error-correction scheme for noisy digital communication links, finding universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, keyword spotting, computational linguistics, and bioinformatics. For example, in speech-to-text speech recognition, the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal. The algorithm is not general; it makes a number of assumptions. First, both the observed events and hidden events must be in a sequence. This sequence often corresponds to time. Second, these two sequences need to be aligned, and an observed event needs to correspond to exactly one hidden event. Third, computing the most likely hidden sequence up to a certain point t must depend only on the observed event at point t, and the most likely sequence at point $t - 1$. These assumptions are all satisfied in a first-order hidden Markov model.

19.3 Other key technical problems

Some other key technical problems in speech recognition are:

Speech recognition system are based on simplified stochastic models, so any aspects of the speech that may be important to recognition but are not represented in the models cannot be used to aid in recognition.

Co-articulation of phonemes and words, depending on the input language, can make the task of speech recognition considerably more difficult. In some languages, like English, co-articulatory effects are extensive and far-reaching, meaning that the expected phonetic signal of a whole utterance can be vastly different than a simple concatenation of the expected phonetic signal of each sound or word. Consider for example the sentence ”what are you going to do?”, which when spoken might sound like ”whatchagonnado?”, which has a phonetic signal which is very different from the expected phonetic signal of each word separately.

Intonation and sentence stress can play an important role in the interpretation of an utterance. As a simple example, utterances that might be transcribed as ”go!”, ”go?” and ”go.” can clearly be recognized by a human, but determining which intonation corresponds to which punctuation is difficult for a computer. Most speech recognition systems are unable to provide any more information about an utterance other than what words were pronounced, so information about stress and intonation cannot be used by the application using the recognizer. Researchers are currently investigating emotion recognition, which may have practical applications. For example if a system detects anger or frustration, it can try asking different questions or forward the caller to a live operator.

In a system designed for dictation, an ordinary spoken signal doesn’t provide sufficient information to create a written from that obeys the normal rules for written language, such as punctuation and capitalization. These systems typically require the speaker to explicitly say where punctuation is to appear.

In naturally spoken language, there are no pauses between words, so it is difficult for a computer to decide where word boundaries lie.

Some sets of utterances can sound the same, but can only be disambiguated by an appeal to context: one famous T-shirt worn by Apple Computer researchers made this point: I helped Apple wreck a nice beach, which, when spoken, sounds like I helped Apple recognize speech. Using common sense and context to disambiguate cases like this can be considered a separate field of inquiry: natural language understanding. A general solution of many of the above problems effectively requires human knowledge and experience, and would thus require advanced pattern recognition and artificial intelligence technologies to be implemented on a computer. In particular, statistical language models are often employed for disambiguation and improvement of the recognition accuracies.

19.4 Text normalization

To recognize the words and phrases specified in a grammar, the speech recognition (SR) engine in Microsoft Speech Server 2004 needs to look up the pronunciation of each word in the grammar. If your grammar contains abbreviations like "Mr. Smith", digit strings like "123" or dollar amounts like "$34.05" the SR engine first converts these strings into one or more unambiguous sequence of words in a process called text normalization. For example the speech recognition engine converts the string "123" into the word sequence "one hundred and twenty three". Once the string is converted the SR engine can then look up the pronunciation of each individual word and use this in the recognition process. Converting a string like "123" is not necessarily as straightforward as turning it into "one hundred and twenty three" though. Other valid interpretations might be "one two three", "hundred twenty three", "one twenty three" or "twelve three". Similarly the abbreviation "Dr." might mean "Doctor" or "Drive" and its correct interpretation is based on context, which can be complicated to determine.

Therefore it is always better to spell out phrases like abbreviations, digit strings, or dollar amounts in your grammar explicitly rather than rely on the SR engine to guess the appropriate phrase for them.

19.5 HMM

A hidden Markov model (HMM) is a statistical model where the system being modelled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters, from the observable parameters, based on this assumption. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications.
There are 3 canonical problems to solve with HMMs:

1. Given the model parameters, compute the probability of a particular output sequence. Solved by the forward algorithm

2. Given the model parameters, find the most likely sequence of (hidden) states which could have generated a given output sequence. Solved by the Viterbi algorithm

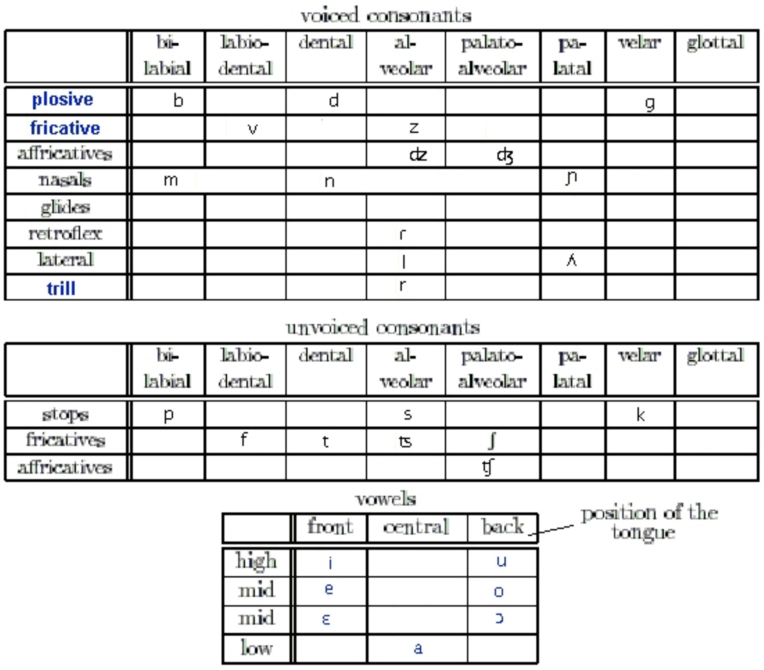
3. Given an output sequence, find the most likely set of state transition and output probabilities. Solved by the Baum-Welch algorithm

19.6 Italian speech sounds

Vowels Italian has seven vowel phonemes: /a/, /e/, /ε/, /i/, /o/, /ς/, /u/. The 'couples' (/e/ - /ε/) and (/o/ - /ς/) are truly different phonemes: compare /'peska/ (fishing) and /'pεska/ (peach). Similarly /'botte/ (barrel) and /'bςtte/ (beatings).

In general, vowel combinations usually pronounce each vowel separately. Diphthongs exist, (e.g. "uo", "iu", "ie", "ai"). The unstressed "u" in a diphthong approximates the English semivowel "w", the unstressed "i" approximates the semivowel "y". E.g.: "buono", "ieri". As a semivowel, "j" is an alternate spelling of "i" (in specific words like "Jesi", "Jacopo").

Triphthongs are limited to a diphthong plus an unstressed "i". (e.g. "miei", "tuoi"). Other sequences of three vowels exist (e.g. noia, febbraio), but they are not triphthongs; they consist of a vowel followed by a diphthong.



voiced consonants

	bi-labial	labio-dental	dental	al-veolar	palato-alveolar	pa-latal	velar	glottal
plosive	b		d				g	
fricative		v		z				
affricatives				dz	dʒ			
nasals	m		n			ɲ		
glides								
retroflex				r				
lateral				l		ʎ		
trill				r				

unvoiced consonants

	bi-labial	labio-dental	dental	al-veolar	palato-alveolar	pa-latal	velar	glottal
stops	p			s			k	
fricatives		f	t	ts	ʃ			
affricatives					tʃ			

vowels

	front	central	back
high	i		u
mid	e		o
mid	ε		ɔ
low		a	

Figure 49: Speech sounds in italian

The phoneme /n/ undergoes assimilation when followed by a consonant, e.g., when followed by a velar (/k/ or /g/) it's pronounced [η, etc.

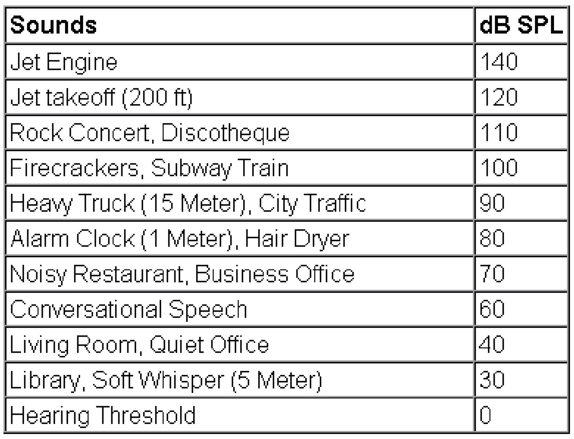
Italian plosives are not aspirated (unlike in English)

Italian has geminate, or double, consonants, which are distinguished by length. Length is distinctive for all consonants except for $/\int/$, $/ts/$, $/dz/$, $/\lambda/$ $/\eta/$, which are always geminate, and /z/ which is always single. Geminate plosives and affricates are realized as lengthened closures. Geminate fricatives, nasals, and /l/ are realized as lengthened continuants. Geminate /r/ is realized as the trill [rr].

19.7 Sound pressure level

The definition of dB SPL is the 20 log of the ratio between the measured sound pressure level and the reference point. This reference point is defined as 0.000002 $\frac{N}{m^2}$, the threshold of hearing. However, the threshold of hearing (and sensitivity to level) changes by frequency and for soft and loud sounds, as discovered by Fletcher and Munson in 1933, shown in the graph below:

Note that human hearing is relatively insensitive to low bass (below 100 Hz), and also compresses at higher sound levels. Here are some typical sounds, and their levels.



Sounds	dB SPL
Jet Engine	140
Jet takeoff (200 ft)	120
Rock Concert, Discotheque	110
Firecrackers, Subway Train	100
Heavy Truck (15 Meter), City Traffic	90
Alarm Clock (1 Meter), Hair Dryer	80
Noisy Restaurant, Business Office	70
Conversational Speech	60
Living Room, Quiet Office	40
Library, Soft Whisper (5 Meter)	30
Hearing Threshold	0

Figure 50: Level of some typical sounds

19.8 The TIMIT Speech Database

The TIMIT Speech Database The TIMIT speech database is designed to provide speech data for the acquisition of acoustic-phonetic knowledge, and for the development and evaluation of speech recognition systems.

TIMIT contains speech from 630 speakers from eight major dialects of American English, each speaking 10 phonetically rich sentences. The TIMIT system includes time-aligned orthographic, phonetic, and word transcriptions as well as speech waveform data for each sentence-utterance. The project was a joint effort among the Massachusetts Institute of Technology, SRI International, and Texas Instruments. The speech was recorded at TI using a Sennheiser head-mounted microphone in a quiet environment, digitizing the speech at a 20 kHz sampling rate and then downsampling to 16 kHz for distribution. The data was transcribed at MIT, and then verified and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST).

Copies of the TIMIT database are available on CD-ROM through the National Technical Information Service (NTIS). Specify the NIST Speech Disc 1-1.1, NTIS Order No. PB91-505065. The domestic price is $100.00 (international price, $300.00). Contact NTIS, Springfield, VA 22161, 703-487-4650, or the Linguistics Data Consortium, University of Pennsylvania, 609 Williams Hall, Philadelphia, PA 19104.

20 Glossary

• ANN - Artificial Neural Network • ASR - Automatic Speech Recognition • AVIOS - American Voice I/O Society • CELP - Code-book Excited Linear Prediction • COLING - COmputational LINGuistics • DTW - Dynamic Time Warping • HMM - Hidden Markov Model • IEEE - Institute of Electrical and Electronics Engineers • JASA - Journal of the Acoustic Society of America • LPC - Linear Predictive Coding • LVCSR - Large Vocabulary Continuous Speech Recognition • LVQ - Learned Vector Quantisation. • MFCC - Mel Frequency Cepstral Coefficients • NLP - Natural Language Processing • NN - Neural Network • TIMIT - A speech corpus with phoneme labels • TTS - Text-To-Speech (i.e. speech synthesis) • VQ - Vector Quantisation