APPLICATIONS 5:
SPEECH RECOGNITION

Theme

Speech is produced by the passage of air through various obstructions and routings of the human larynx, throat, mouth, tongue, lips, nose etc. It is emitted as a series of pressure waves. To automatically convert these pressure waves into written words, a series of operations is performed. These involve capturing and representing the pressure waves in appropriate notations, creating feature vectors to represent time-slices of the converted input, clustering and purifying the input, matching the results against a library of known sound vectorized waveforms, choosing the most likely series of letter-sounds, and then selecting the most likely sequence of words.

Summary of contents

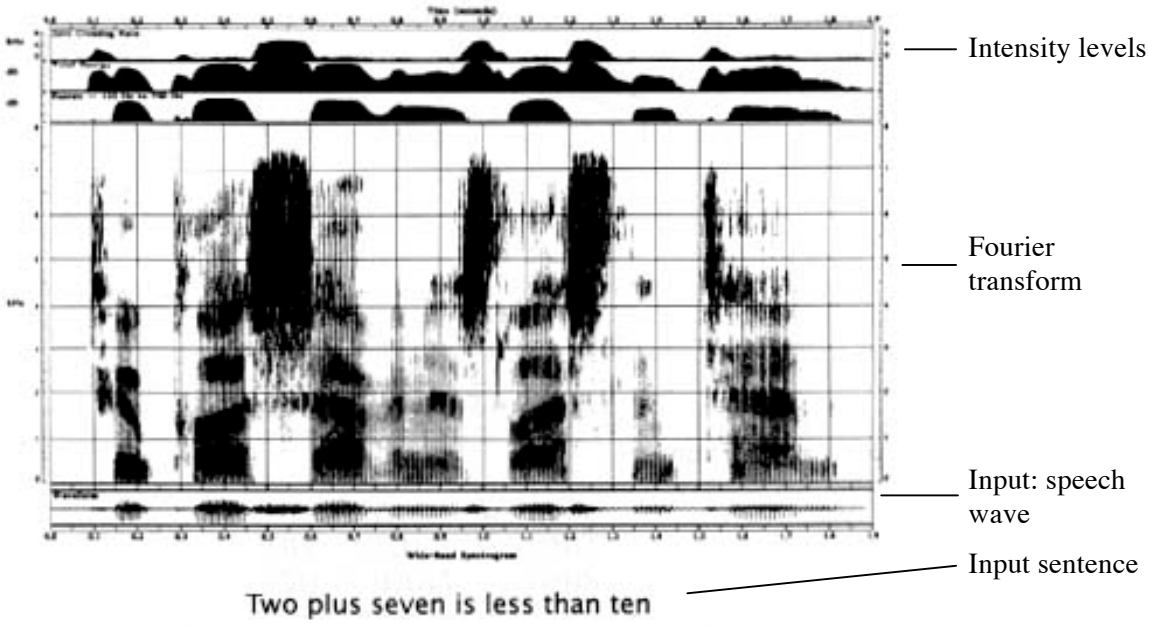
1. Speech Recognition Systems

Introduction

Early speech recognition systems tried to model the human articulatory channel. They didn't work. Since the 1970s, these systems have been trained on example data rather than defined using rules. The transition was caused by the success of the HEARSAY and HARPY systems at CMU.

Step 1: Speech

Speech is pressure waves, travelling through the air. Created by vibrations of larynx, followed by openings or blockages en route to the outside. Vowels and consonants.



Step 2: Internal representation

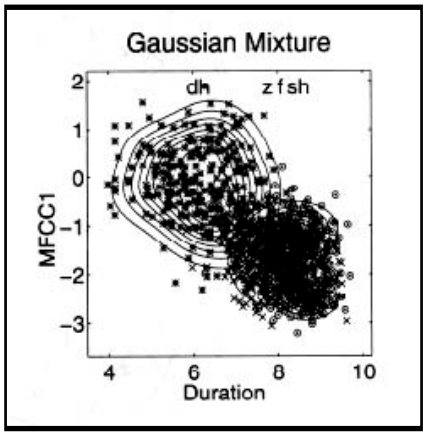
1. The basic pressure wave is full of noise and very context-sensitive, so it is very difficult to work with. First, perform Fourier transform, to represent the wave as a sum of waves at a range of frequencies, within a certain window. This is shown in the speech spectrogram on the previous page. Now work in the Frequency domain (on the y axis), not the waveform domain. Try various windows to minimize edge effects, etc.

2. Decompose (deconvolve) the Fourier-transformed waves into a set of vectors, by cepstral analysis. Chop up the timeline (x-axis) and the frequency space (y-axis) to obtain 'little squares', from which you obtain the quantized vectors. Now certain operations become simpler (like working in log space; can add instead of multiply), though some new steps become necessary. Move a window over the Fourier transform and measure the strengths of the voice natural frequencies $f_0, f_1, f_2 \ldots$

Step 3: Purify and match: **Acoustic Model**

1. After quantizing the frequency vectors, represent them in an abstract vector space (axes: time and MFCC cepstral coefficients). The resulting vectors, one per timeslice, provide a picture of the incoming sound wave. Depending on window size, speech inconsistencies, noise, etc., these vectors are not pure reflections of the speaker's sounds. So purify the vector series by clustering them, using various algorithms, to find the major sound 'bundles'. Here it's possible to merge vectors across time as well, to obtain durations.

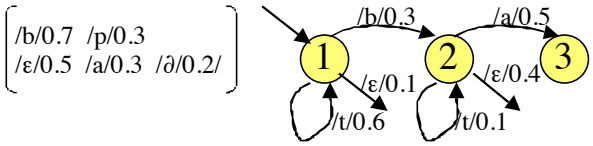
2. Then match the bundles against a library of standard sound bundle 'shapes', represented as durations vs. cepstral coeffecients. To save space, these standard sound bundles are represented as mixtures of Gaussian curves (then only need save two or three parameters per curve)—these are the contour lines in the picture below, for one MFCC coefficient. Try to fit them over the clustered points (like umbrellas). To find the best match (which vector corresponds with which portion of the curve?), use the EM algorithm.



Step 4: Identify sound sequences and words: Lexical Model

Now you have a series of sounds, and you want a series of letters. But unfortunately, sounds and letters do not line up one-to-one. So first represent typical sound sequences in a Hidden Markov Model (like a Finite State Network). For each sound, create all possible links to all other sounds, and arrange these sounds into the HMM. Initialize everything with equal transition probabilities. Then train the transition probabilities on the links using training data, for which you know both the sounds and the correct letters.

Given a new input (= sound sequence), use the Viterbi algorithm to match the incoming series of sounds to the best path through the HMM, taking into account likely sound shifts, etc., as given by the probabilistic sound transitions on the HMM arcs.



A typical large-vocabulary system takes into account context dependency for the phonemes (so phonemes with different left and right context have different realizations as HMM states); to do this it uses cepstral normalization to normalize for different speaker and recording conditions. One can do additional speaker normalization using vocal tract length normalization (VTLN) for male-female normalization and maximum likelihood linear regression (MLLR) for more general speaker adaptation.

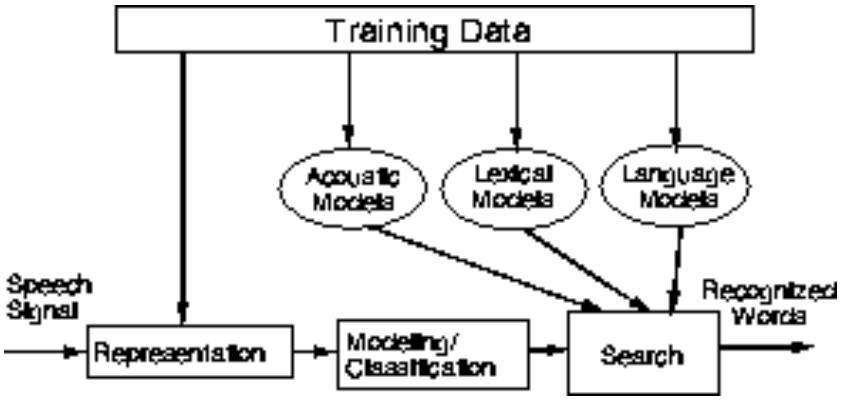
Step 5: Sentences: Language Model

Finally, you have a series of words. But do they form a sentence? At this point you could use a parser and a grammar to see. Trouble is, the system's top word selections often include errors; perhaps the second- or the third-best words are the correct ones. So create an n-gram language model (in practice, a trigram), for the domain you are working in. This language model provides a probabilistic model of the word sequences you are likely to encounter. Now match the incoming word sequence (and all its most likely alternatives) against the n-gram language model, using the Viterbi algorithm, and find what is indeed the most likely sensible sentence.

Output the resulting sequence of words.

Overall System Architecture

The typical components/architecture of an ASR system:



2. Evaluation

Measures

Principal measure is Word Error rate (WER): measure how many words were recognized correctly in known test sample.

$$WER = (S + I + D) * 100 / N$$

where *N* is the total number of words in the test set, and *S, I,* and *D* are the total number of substitutions, insertions, and deletions needed to convert the system's output string into the test string.

WER tends to drop by a factor of 2 every 2 years; in nicely controlled setting in the lab, with limited vocabularies, systems do quite well (WER in the low single digits). But in real life, which is noisy and unpredictable, where people use made-up words and odd word mixtures, it's a different story.

In dialogue systems, people use Command Success Rate (CSR), in which the dialogue engine and task help guide speech recognition; now measure the success for each individual command, and for each task as a whole.

Performance

Corpus	Speech Type	Lexicon Size	Word Error Rate (%)	Human Error Rate (%)
connected digits	spontaneous	10	0.3	0.009
resource mangmnt	read	1000	3.6	0.1
air travel agent	spontaneous	2000	2	–
Wall Street Journal	read	64000	7	1
radio news	mixed	64000	27	–
tel. switchboard	conversation	10000	38	4
telephone call home	conversation	10000	50	–

	pre-1975	1975–1985	1985–1997	present
Unit recognized	sub-word; single word	sub-word	sub-word	sub-word
Unit of analysis	single word	fixed phrases	bigrams, trigrams	dialogue turns
Approaches to modeling	heuristic, ad hoc	template matching	mathematical	mathematical
	rule-based, declarative	data-driven; deterministic	data-driven; probabilistic	data-driven; probabilistic; simple task analysis
Knowledge representation	heterogeneous	homogeneous	homogeneous	unclear
Knowledge acquisition	intense manual effort	embedded in simple structures	automatic learning	learning + manual effort for dialogues

3. Speech translation

The goal: a translating telephone.

Research projects at CMU, Karlsruhe, ARL, etc.

The Verbmobil project in Germany translated between German and French using English as interlingua. A large multi-site consortium, it kept NLP funded in Germany for almost a decade, starting mid-1990s.

The earliest commercial product (PC based), in 2002, came from NEC, for $300. Since 2009 they sell a PDA based version that includes 50,000 words, bigrams, some parsing and a little semantic transfer; see http://www.nec.co.jp/press/en/0901/0503.html.

The YoChina system was deployed by researchers at DFKI in Germany in 2010 for the Beijing Olympics. Your speech is sent back to a server in Germany. See http://itunes.apple.com/app/yochina-language-travel-culture/id401123365?mt=8.

The US Army Phrasalator: English-Arabic-English in a very robust box (able to withstand desert fighting conditions): speech recognition and phrasal (table lookup) translation and output.

In the past five years many commercial handheld translators have become available. They're all phrasal lookup. The larger ones call back to a home server for the translator (e.g., the Chinese-English-German-etc. system built at DFKI in Germany).

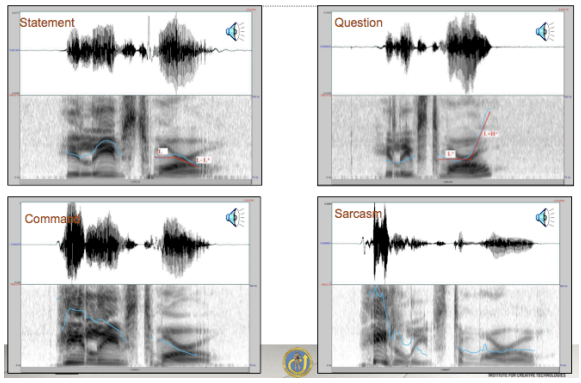
4. Prosody

An increasingly interesting topic today is the recognition of emotion and other pragmatic signals in addition to the words. Human-human speech is foundationally mediated by prosody prosody (rhythm, intonation, etc., of speech). Speech is only natural when it is not 'flat': we infer a great deal about speaker's inner state and goals from prosody.

Prosody is characterized by two attributes:

- Prominence: Intonation, rhythm, and lexical stress patterns, which signal emphasis, intent, emotion
- Phrasing: Chunking of utterances into prosodic phrases, which assists with correct interpretation

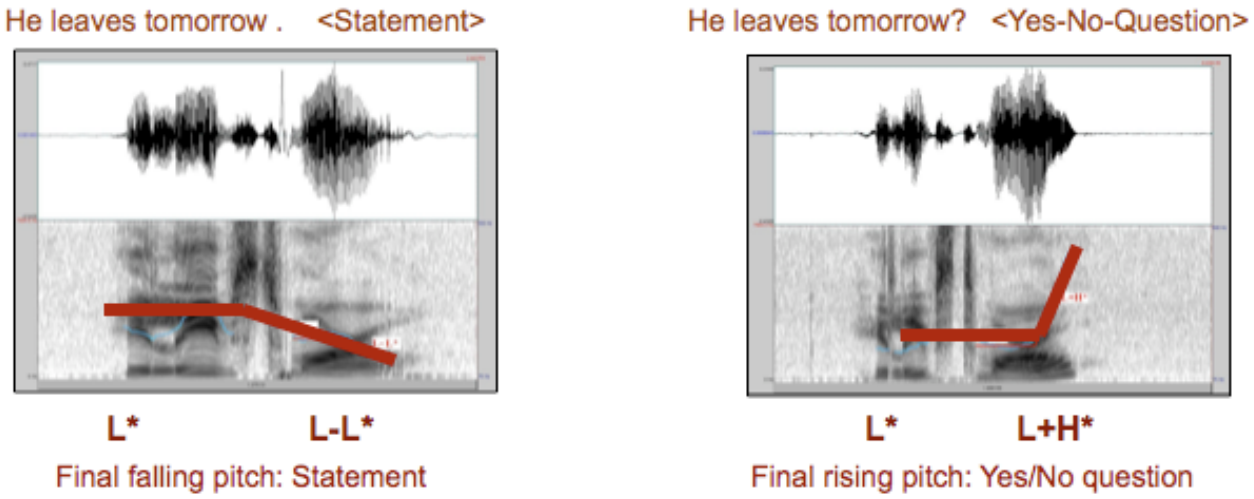
The sentence "he leaves tomorrow", said four ways (statement, question, command, sarcastically):



To handle prosody, you need to develop:

- Suitable representation of prosody
- Algorithms to automatically detect prosody
- Methods to integrate these detectors in speech applications

To represent prosody, you **e**xtract features from the pitch contours of last 200 msec of utterances and then convert the parameters into a discretized (categorical) notation.



Shri Narayanan and students in the EE Department at USC, and others elsewhere, are detecting three features: **pitch** ('height' of voice), **intensity** (loudness), and **breaks** (inter-word spaces). They use the ToBI (TOnes and Break Indices) representation. Procedure: To find the best sequence of prosody labels L

Best prosodic sequence: $L^* = \{l_1, l_2, \cdots, l_n\}$

$$\begin{aligned} L^* &= \arg\max_L P(L|W,S,A) \\ &\approx \arg\max_L \prod_i^n p(l_i|w_{i-k}^{i+k}, s_{i-k}^{i+k}, \mathbf{a}_{i-k}^{i+k}) \end{aligned}$$

$$L^* = \arg\max_L \prod_i^n P(l_i|\mathbf{\Phi})$$

- They assign a prosodic label to each word conditioned on contextual features
- They train continuous density Hidden Markov Models (HMMs) to represent pitch accent and boundary tone — 3 states

They use the following kinds of features:

- Lexical features: Orthographic word identity
- Syntactic features: POS tags, Supertags (Similar to shallow syntactic parse)
- Acoustic features: f_0 and energy extracted over 10msec frames

5. Current status

Applications:

1. General-purpose dictation: Several commercial systems for $100:

- *DragonDictate* (used to be Dragon Systems; by Jim Baker); now at *Nuance* (www.nuance.com)
- IBM *ViaVoice* (from Jim Baker at IBM)
- Whatever was left when Lernout and Hauspie (was Kurzweil) went bankrupt
- Kai-Fu Lee takes SPHINX from CMU to Apple (PlainTalk) and then to Microsoft Beijing
- *Windows Speech Recognition* in Windows Vista

2. Military: Handheld devices for speech-to-speech translation in Iraq and elsewhere. Also used in fighter planes where the pilot's hands are too busy to type.

3. Healthcare: ASR for doctors in order to create patient records automatically.

4. Autos: Speech devices take driver input and display routes, maps, etc.

5. Help for disabled (esp to access the web and control the computer).

Some research projects:
DARPA: ATIS Travel Agent (early 1990s); GALE program (mid-2000s)
MIT: GALAXY global weather, restaurants, etc.
Dutch Railways: train information by phone
DARPA: COMMUNICATOR travel dialogues; BABYLON handheld translation devices

Current topics: Interfaces

Speech systems in human-computer interfaces.

Problems for ASR

- Voices differ (men, women, children)
- Accents
- Speaking speed (overall, specific cadences)
- Pitch variation (high, low)
- Word and sentence boundaries
- Background noise — BIG PROBLEM
- Genuine ambiguity: "recognize speech" vs. "wreck a nice beach"

5. Speech Synthesis

Traditional model
Lexicon of sounds for letters
Problems: flat
Enhance: sentence prosody contour. Also need Speech Act and focus/stress as input

Concatenative synthesis
Record speaker many times; create lexicon of sounds for letters, in word start/middle/end, sentence start/middle/end, stress/unstress, etc. forms
At run time, choose most fitting variant, depending on neighboring options (intensity/loudness, speed, etc.)
Problems: clean sound units for letters, matching disfluencies

Optional Readings

Victor Zue's course at MIT.

Jelinek. F. 1998. *Statistical Methods for Speech Recognition*.

Rabiner, L. 1993. *Fundamentals of Speech Recognition*.

Schroeder, M.R. 2004. *Computer Speech* (2nd ed).

Karat, C.-M., J. Vergo, and D. Nahamoo. 2007. Conversational Interface Technologies. In A. Sears and J.A. Jacko (eds) *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications (Human Factors and Ergonomics)*. Lawrence Erlbaum Associates Inc.