SPEECH PROCESSING

Speech processing refers to the manipulation or transformation of speech signals, mostly by computer, to more useful forms of information. Usually, of course, signals identified as speech are produced by humans, and they do the processing necessary to produce or generate speech; similarly, humans receive speech via their ears and process it in their brains to understand the spoken message. In this article, we deal with ways that computers can simulate either production and/or perception of speech. There are many applications for such computer simulation using speech: speech synthesis from text (e.g., a reading machine for the visually handicapped, or a voice response system from a database), speech recognition (e.g., voice control of machines, or entry/interrogation of databases), speaker verification (e.g., allowing access to secure systems by voiceprint), and speech coding (i.e., compressing speech into a compact form for storage or transmission, and subsequent reconstruction of the speech). All of these uses require converting the original sound waves of pressure (which actually constitute speech) into a compact digital form suitable for computer manipulation, or vice versa.

Such processing usually involves analysis and/or synthesis of speech. Recognition of speech (where the output is the text usually associated with the speech) or of speakers (where the output is the identity of the source of the speech, or a verification of a claimed identity) requires analysis of the speech, to extract relevant (and usually efficient) features that characterize how the speech was produced (e.g., features related to vocal tract shape). Text-to-speech synthesis converts normal text into an understandable synthetic voice. In speech coding, both the input and output are in the form of speech; thus text is not directly involved, and the objective is simply a compression of the information rate, for efficient and secure transmission.

SPEECH ANALYSIS

The first step in speech analysis is to convert the continuously-varying speech waves into a bit stream [i.e., analog-to-digital (A/D) conversion], so that computers may receive and process the speech signal. Two key factors here are the sampling rate (number of samples/second) and the sampling precision (number of bits/sample). The rate is directly proportional to the preserved bandwidth of the speech; the Nyquist sampling theorem specifies that the rate be at least twice the highest frequency in the speech (1). The minimum rate used in practical applications is usually 8000 samples per second, thus preserving from 0 Hz to 4 kHz (in practice, an analog

lowpass filter usually precedes the A/D converter, and there is a gradual cutoff of frequencies before 4 kHz); this low rate is suitable (and is standard) for telephone applications, because the switched network only preserves the 300–3200 Hz range. The highest rate typically employed is 44,000 samples/s, which preserves frequencies well beyond the normal hearing range and is used for audio on compact disks (2–5).

The choice of bits/sample is not governed by a fixed rule, such as the Nyquist law, but rather corresponds to how much distortion is tolerable in a given speech application. To be unable to hear distortion after A/D conversion, we typically need 11 bits/sample. This retains about 60 dB (decibels) in signal-to-noise ratio, which allows for a typical 30 dB range between strong vowels and weak fricatives in speech (speech sounds include vowels such as /i, a, u/, nasals such as /m, n/, fricatives such as /f, s, z/, and stops such as /p, t, g/). Hence, 12 bit A/D converters are acceptable, although 16 bit systems are more common. In the telephone network, where bandwidth is at a high premium, 8 bit logarithmic (nonuniform) quantization is used on most digital links.

The objective of speech analysis is to transform speech samples into a more informative representation, for specific objectives other than simply furnishing speech to human ears. The transmission rate of basic digital speech (via simple or logarithmic A/D conversion) is typically 64 kbit/s or higher. If we compare that rate to the amount of fundamental information in the signal, we can see a large discrepancy. An average speaking rate is approximately 12 phonemes (individual sounds) per second, and most languages have an inventory of approximately 32 phonemes. Thus the phonemic sequence of speech can be sent in approximately 60 bits/s ($12 \log_2 32$). This does not include intonational or speaker-specific aspects of the speech, but also ignores the fact that phoneme sequences are not random (e.g., Huffman coding can reduce the rate). The overall information rate in speech is about 100 bits/s. Current speech coders are far from providing transparent coding (i.e., speech without loss of quality) at such rates, but some complex systems can reduce the rate to about 8 kbit/s (at 8000 samples per second) without losing quality.

Typical analysis methods try to extract features that correspond to some well-known aspects of speech production or perception. For example, experiments have shown that speakers can easily control the intensity of speech sounds (and that listeners can easily detect small changes in intensity) (6). Thus intensity (or a related parameter, energy) is often determined in speech analysis, by simply summing a sequence of (squared) speech samples. Similarly, the positions of resonance (or formant) peaks in the amplitude spectrum of speech have been correlated with shapes of the vocal tract in speech production; they are also well discerned perceptually (peaks are much more salient than spectral valleys). Many analysis techniques try to extract compact parameterizations of these spectral peaks, either modeling the underlying resonances of the vocal tract (including resonance bandwidths) or some form of the detail (both broad and fine) in the amplitude spectrum.

Another feature that is often examined in speech analysis is that of the fundamental frequency (F0) of the vocal cords, which vibrate during voiced (periodic) sounds such as vowels. (In unvoiced sounds, the cords do not vibrate.) The rate F0 is directly controlled in speech production, and the resulting periodicity is easily discriminated by listeners. It is used in tone languages directly for semantic concepts, and cues many aspects of stress and/or syntactic structure in other languages. Although F0 (or its inverse, the pitch period—the time between successive closures of the vocal cords) is rarely extracted for speech recognizers, it is often used in low-bit-rate speech coders, and must be modeled well for speech synthesizers.

SPEECH CODING

The objective of speech coding is to represent speech as compactly as possible (i.e., minimal bits/second), while retaining high intelligibility and naturalness, with minimal time delay in inexpensive hardware. There are many practical compromises possible here, ranging from zero-delay, cheap, transparent, high-rate pulse-code modulation (PCM) to systems that trade off naturalness for decreased bit rates and increased complexity (5). We can distinguish higher rate *waveform-based* coders (usually yielding high-quality speech) from lower rate *parametric* coders. The former coders reconstruct speech signals sample by sample and usually represent each speech sample directly with at least a few bits, whereas the latter coders typically discard phase information and process blocks (or frames) of many samples at a time, which allow the average number of bits/samples to be below one (i.e., speech coding under 8 kbits/s).

Coders Exploit Structure in Speech

Coders typically try to identify structure in signals, extract it for efficient representation, and leave the remaining (less predictable) signal components either to be ignored (in low-rate systems) or coded using simple techniques (in high-rate systems). There are many sources of structure in speech signals (which, indeed, account for the three orders of magnitude difference between the 64 kbits/s of log PCM speech and an approximately 100 bits/s theoretical limit). Sounds or phonemes average a lengthy 80 ms (e.g., perhaps ten pitch periods), largely due to the difficulty of articulating speech more rapidly (but also due to losses in human perception at higher speaking rates). However, identification of each sound is possible from a fraction of each pitch period (at least for speech in noise-free environments). The repetition of information in multiple periods helps build redundancy in human speech communication, allowing reliable communication even in difficult conditions.

The spectra of most speech sounds show a regular structure, which is the product of a periodic excitation (due to vocal cord vibration) and the set of resonances of the vocal tract (which appears as a series of peaks and valleys, averaging about one peak every 1 kHz). Rather than code the speech waveform sample-by-sample or the spectrum point-by-point, efficient coders extract from the speech a number of parameters directly related to the overall amplitude, F0, and the spectral peaks, and use these for transmission.

One simple spectral measure sometimes used in speech analysis is the zero-crossing rate (ZCR), which provides a basic estimate of the frequency of major energy concentration in the signal. The ZCR for speech is just the number of times the speech signal crosses the time axis (i.e., changes algebraic sign) in a given time period (e.g., taking an overly simple case, a sinusoid of 100 Hz has a ZCR of 200/s). Background

noise often has a flat or broad lowpass spectrum and thus a ZCR corresponding roughly to a fixed frequency in the middle-to-low range of the signal bandwidth. For speech obstruents (stops and fricatives), on the other hand, the ZCR is either high (corresponding to the high-frequency concentration of energy in fricatives and stop bursts) or very low (if a voicebar dominates). Weak fricatives, which cause the most detection difficulties, have high ZCRs. Information about energy and ZCR has low calculation cost and can be used in speech endpoint detection (see below).

Exploiting Simple Structure

Advanced coders require examination of at least several successive speech samples as a block of data (which entails more computation and delay). The simplest coders (PCM) represent each sample independently. Using a uniform quantizer, however, is inefficient because most waveform samples tend to be low in size, and thus the quantizer levels assigned to large samples are rarely used. Optimal encoding occurs when all levels are equally used on the average. Because speech samples typically follow a gamma probability density function (their likelihood decaying roughly as an exponential with increasing amplitude), a logarithmic compression prior to uniform quantization is best (e.g., the μ-law or A-law log PCM, common in telephone networks) (1).

Many coders adapt in time, exploiting slowly changing characteristics of the speech, related to either the shape or movements of the vocal tract. For example, the step size of the quantizer can be adjusted to follow excursions of the speech signal integrated over time periods ranging from 1 to 100 ms. When the signal has large energy, larger step sizes are needed to avoid clipping (which causes highly nonlinear and severe distortion). By reducing the step size for low-energy samples, the quantization noise is proportionally reduced without clipping.

Exploiting Detailed Spectral Structure

Many speech coders use a form of linear prediction to estimate a current sample based on a linear combination of previous samples (7). This is useful when the speech spectrum is nonuniform (unlike speech, pure white noise, with independent samples and a flat spectrum, achieves no gain through such prediction). The simplest prediction is found in delta modulation, where one previous speech sample directly provides the estimate of the current sample, the assumption being that a typical sample changes little from its immediately prior neighbor. This becomes truer if we sample well above the Nyquist rate, as is done in delta modulation, which allows the use of a one-bit quantizer (trading off sampling rate for bit precision) (8).

The power of linear prediction becomes more apparent when the order of the predictor (i.e., the number of prior samples used) is approximately 10, and when the predictor adapts to movements of the vocal tract (e.g., is updated every 10–30 ms). This occurs in two very popular speech coders, the adaptive differential PCM (ADPCM) and linear predictive coding (LPC). They exploit the fact that the major excitation of the vocal tract occurs at the time that the vocal cords close (once per pitch period), which causes a sudden increase in speech amplitude, after which the signal decays exponentially (with a rate inversely proportional to the bandwidth of the strongest resonance—usually also the lowest-frequency formant). Each pitch period approximately corresponds to overlapped impulse responses of the vocal tract and is directly related to the resonances of the vocal tract. Because there are about four such formants (in a typical 0–4 kHz bandwidth employed in 8000 samples per second speech) and since each formant can be directly characterized by a center frequency and a bandwidth, a predictor order of ten or so is adequate (higher orders give more precision, but offer diminishing returns for further increases in computation and bit rate).

The ten to sixteen LPC parameters derived from such an analysis form the basis of many coders, as well as for speech recognizers. These parameters are the multiplier coefficients of a direct-form digital filter, which when used in a feedforward fashion (an all-zero filter), converts a modeled speech signal into a *residual* or *error* signal, which is then suitable either for simple coding (with fewer bits, typically 3–4/sample, in ADPCM) or further parameterization (as in LPC systems).

Using the coefficients in a feedback fashion, the all-pole filter acts as a speech synthesizer, when excited by an impulse train (impulses spaced every 1/F0 ms) or a noise signal (simulating the frication noise generated at a narrow constriction in the vocal tract). This latter excitation is a very simple model of the LPC residual signal, which allows very low bit rates in basic LPC systems (9). The result is intelligible but synthetic-quality speech (less natural than the *toll-quality* speech, as in the telephone network with log PCM, or other medium-rate, waveform-coding methods). The LPC model, with its dozen spectral parameters parameterizing the vocal tract shape (filter) and its three excitation parameters (F0, amplitude, and a single "voicing" bit noting a decision whether the speech is periodic or not) modeling the residual, operates around 2.4 kbits/s (using updates every 20 ms). The LPC coefficients are usually transformed into a more efficient set for coding, such as the reflection coefficients (which are the multipliers in a lattice-form vocal tract filter, and can actually correspond to reflected energy in simple three-dimensional vocal tract models, modeling the two traveling waves, one going up the vocal tract, the other down). Another popular set is the line-spectral frequencies (LSF) that displace the resonance poles in the spectral z-plane onto the unit circle, which allows more efficient differential coding in one dimension around the circle, rather than the effectively two-dimensional coding of the resonances inherent in other LPC forms. These latter structures guarantee stable synthesis filters, as well.

Exploiting Structure across Parameters

The most efficient coders go beyond simple temporal and spectral structure in speech to exploit correlations across parameters, both within successive frames of speech data and across frames. Even relatively efficient spectral representations such as the LSFs do not produce orthogonal parameter sets; that is, there remain significant correlations among sets of parameters describing adjacent frames of speech. Shannon's theorem states that it is always more efficient to code a signal in vector form, rather than code the samples or parameters as a succession of scalars. Thus, we group related parameters as a block and represent the ensemble with a single index. For example, to code a speech frame every 20 ms, a set of 10 re-

flection coefficients might need 50 bits as scalar parameters (about 5 bits each), but need only 10 bits in vector quantization (VQ). In the latter, we chose 1024 ($= 2^{10}$) representative points in 10-dimensional space (one dimension/parameter). The points are usually chosen in a training phase (coder development) by examining many minutes of typical speech (ranging over different speakers and phonemes). Each frame provides one point in this space, and the chosen points for the VQ are the centroids of the most densely populated clusters. Because there are only approximately 1000 different spectral patterns that are easily discriminated in speech perception (ignoring the effects of F0 and overall-amplitude), a 10 bit system is often adequate. A 10 bit VQ represents effectively a practical limit computationally as well, because a search for the optimal point among much more than 1024 possibilities every 20 ms exceeds current hardware capacity. Basically, VQ trades increased computation in the analysis stage for lower bit rate in transmission or storage. The most common use today for VQ is in code-excited linear prediction (CELP) speech, in which short sequences of LPC residual signals are stored in a codebook to excite an LPC filter. CELP provides toll-quality speech at low rates, and has become very popular in recent years (8).

FUNDAMENTAL FREQUENCY (F_0) ESTIMATION (PITCH DETECTORS)

Both low-rate speech coders and text-to-speech synthesizers require estimation of the F0 of voiced speech signals, as well as the related estimation of the presence versus absence of periodicity (voicing). At first glance, it seems a simple task to detect periodicity and measure the period. However, despite hundreds of algorithms in the literature (10), no one pitch detector (so-called due to the close correlation of F0 and perceived pitch) can handle more than approximately 95% of speech without errors. Environmental noise often obscures the periodicity, and the interaction of phase, harmonics, and spectral peaks often creates ambiguous cases where pitch detectors can make mistakes.

The basic approach to F0 estimation simply looks for peaks in the speech waveform, spaced at intervals roughly corresponding to typical pitch periods. Periods can range from 2 ms (for small infants) to 20 ms (for large males), but each individual speaker usually employs about an octave range (e.g., 80–160 Hz for a typical adult male). Many estimators use heuristics, such as the fact that F0 cannot change abruptly (except when voicing starts or ceases) in successive periods. Because F0 is roughly independent of which phoneme is being uttered and the structure of formants (and related phase effects) in a voiced spectrum can obscure F0 estimation, we often eliminate from analysis the frequencies above 900 Hz (via a lowpass filter), thus retaining one strong formant containing several harmonics to supply the periodicity information. At extra cost, more spectral flattening can be provided by autocorrelation or by LPC inverse filtering, to further reduce F0 estimation errors. Other pitch detectors do peak-picking directly on the harmonics after a Fourier transform of the speech (10).

AUTOMATIC SPEECH RECOGNITION

Automatic speech recognition (ASR) is a pattern recognition problem; in principle, one could store all possible speech signals and ask linguists to transcribe each with its correct corresponding text. Given faster computation and the decreasing cost of computer memory, one might wonder if this radically simple approach will eventually solve the ASR problem. To see that this is not true, just consider the immense number of possible utterances. Typical utterances last a few seconds; at 10,000 samples per second, we potentially have, say, $2^{30,000}$ signals. Even with very efficient coding (e.g., 100 bits/s), we would have an immense 2^{300} signals. Most of these would not be recognizable as speech, but it is impossible a priori to just consider ones that might eventually occur as an input to an ASR system (even enlisting millions of speakers talking for hours would only scratch the surface of possibilities).

Thus we must find ways to process each speech signal to be recognized so that the amount of information is reduced from perhaps 64 kbits/s to a much lower figure. Obvious candidates are low-rate coders, since they generally preserve enough information to reconstruct intelligible speech. (Higher-rate waveform coders preserve too much information, which may be needed for naturalness but not for intelligibility; the latter is the essence of ASR.)

Unlike speech coding, where the objective of speech analysis is to reproduce speech from a compact form, speech recognition instead transforms speech into its corresponding textual equivalent. The direct relationship between the spectral envelope of speech and vocal tract shape (and hence the phoneme being uttered) has led to intense use of efficient representations of spectral envelope for ASR. (The relative independence of F0 and phonemes, on the other hand, has led ASR to largely ignore F0, except with tone languages, despite its use for semantic and syntactic information in human speech recognition.) One difficulty has been how to extract compact yet relevant information about the envelope for ASR. Simple energy is useful, but is often subject to variations (e.g., automatic gain control, variable mouth-to-microphone distance, varying channel gain) that are irrelevant for phonemic distinctions; as a result, energy is often not used for ASR, but change in energy between frames is used.

Linear predictive coding parameters were once popular for ASR, but they have been largely replaced by the mel-frequency cepstral coefficients (MFCCs) (11–12). Mostly due to empirical results showing superior ASR performance, the MFCC enjoy widespread use in ASR. The mel-scale refers to a frequency-axis deformation, to emphasize the lower frequencies more than higher ones (which is quite difficult to do in LPC analysis). This follows critical-band spacing in audition (4), where perceptual resolution is fairly linear below 1 kHz, but becomes almost logarithmic above that. The cepstrum is the inverse transform of the log-amplitude of the Fourier transform of speech. The amplitude spectrum of speech in decibels is warped by triangular filters spaced at critical bands, and then coefficients are produced by weightings from increasingly higher frequency sinusoids (in the inverse Fourier transform). The first 10 or so MFCCs provide a good spectral envelope representation for ASR. The first MFCC C_0 is effectively the overall speech energy (and is often omitted from use); C_1 provides a simple measure of the balance between low- and high-frequency energy (the one-period sinusoid of the inverse transform weights low frequencies positively and high ones negatively). Higher coefficients provide the increasingly finer spectral details needed to distinguish, say, the vowels /i/ and /e/.

Timing Problems in Automatic Speech Recognition

The major difficulty for ASR is the large amount of variability in speech production. In text-to-speech synthesis, one synthetic voice may suffice, and all listeners must adjust to its accent. However, ASR systems must accommodate different speaking styles, by storing many different speakers' patterns or by integrating knowledge about different styles. Variations occur at several levels: for example, timing, spectral envelope, intonation. There is much freedom in the timing of articulations and how the vocal tract moves. In the past, ASR commonly stored templates consisting of successive frames of spectral parameters (e.g., LPC coefficients), and compared them with those of an unknown utterance. Because utterances of the same text could easily have different numbers of frames, the alignment of frames could not simply be one-to-one. Nonlinear dynamic time warping (DTW) was popular because it compensated for small speaking rate variations. Unfortunately, it was still computationally expensive and extended awkwardly to long utterances; in addition, it was difficult to improve the model with more training speech.

General Stochastic Approach

In the early 1980s, the hidden Markov model (HMM) approach gradually took over as the dominant method for ASR. It traded large amounts of computation during an initial training phase for a more flexible and faster model at recognition time. Furthermore, it provided a mathematically elegant and computationally practical solution to the serious problems of variability in speech production. Dynamic time warping had partially solved the timing problem by allowing nonlinear time paths in the recognition search space but was unsatisfactory with regard to variations in pronunciation (e.g., differences in spectrum due to perturbations in vocal tract shape, different speakers, phonetic contexts). Essentially, DTW was a deterministic approach to a stochastic problem.

With HMMs, speech variability was handled in terms of probabilities. The likelihood of a speaker uttering a certain sound in a certain context was modeled by probability distributions, which were estimated from large amounts of training data speech. The current dictum of "there's never enough data" or "there's no data like more data" rules the current approaches toward ASR. Given sufficient computer power and memory, ASR systems tend to have improved recognition accuracy as more and more data are included in the stochastic models to make them more reliable. Although computer resources are never infinite, this approach is feasible to improve systems for speaker-independent ASR, where training speech is obtained from hundreds of different speakers, and the system accepts input speech from all users.

For alternative speaker-dependent recognizers, which need training for each individual user and only employ models trained on that speaker's voice at recognition time, the endless possibility of increasing the training data is much less feasible, given most users' reluctance to provide more than a few minutes of speech. Speaker-dependent systems have better recognition accuracy, because the HMM models are directly related to each user's speech, whereas speaker-independent HMMs must model more broadly across the diversity of many speakers (as a result, such models are less discriminative). A third category, speaker-adaptive recognizers, start from speaker-independent models and adapt the models while each new speaker uses the system.

It is clear that stochastic modeling has an important place in speech processing, given the large amount of variability in speech production. Humans are incapable of regenerating exactly the same utterance twice. With effort, they can produce utterances that sound virtually the same to listeners, but at the sampling and precision levels at which speech analyzers function, there are always differences that result in variable parameters. If the ASR parameters are well chosen and the variation is minimal, system performance can be high if proper stochastic models are used. However, all too often, variability is large and beyond the modeling capability of current ASR systems. Future ASR must combine deterministic knowledge modeling (e.g., the so-called expert system approach to artificial intelligence tasks, like ASR) with well-controlled stochastic models to accommodate the inevitable variability. Currently, the pendulum has swung far away from the expert-system approach common in the mid-1970s (11).

Most ASR is done using the maximum likelihood (ML) approach, in which statistical models for both speech and text are estimated based on prior training data. After training establishes the models, an input speech signal is analyzed at recognition time and the text with the highest likelihood of corresponding with the speech is chosen as the recognition output. Thus, given a signal S, we choose text T which maximizes the a posteriori probability, using Bayes Rule,

$$P(T|S) = P(S|T)P(T)/P(S)$$

It is impossible to get good estimates for $P(T|S)$ because of the extremely large number of possible speech signals S. Instead, we develop estimates for $P(S|T)$ (the acoustic model) and for $P(T)$ (the language or text model). When maximizing across possible texts T, we can ignore the denominator $P(S)$ term, because it is the same for all T. Even for large vocabularies, the number of text possibilities is much smaller than the number of speech signals; thus it is more practical to estimate $P(S|T)$. For each possible text, the statistics are obtained from speakers repeatedly uttering that text (in practice, such estimates can be obtained for small text units, such as words and phonemes, while using speech of sentences). The a priori likelihood of a text T being spoken is $P(T)$, which is obtained by examining computerized textual databases.

There are efficient methods to develop such statistics and to evaluate the large number of possibilities when searching for the maximum likelihood. (The search space for a vocabulary of thousands of words and for an utterance of several seconds, at 100 frames per second, is quite large.) The forward–backward method examines all possible paths, summing many small likelihoods, while the more efficient Viterbi method looks for the single best path (11). The latter is much faster and is commonly used at the recognition phase, because it tends to sacrifice little in recognition accuracy. The forward-backward method is popular in the training phase, where computational speed is less important. When trying to discriminate similar words, however, the ML approach sometimes fails, because it does not examine how close alternative possible texts are. In addition, all speech frames are treated as equally important, which is not the case in human speech perception. Alternative methods such as Maximum Mutual Information Estimation or Linear Disciminant Analysis are

more expensive, but are more selective in examining the data, focusing on the differences between competing similar texts, when examining a speech input.

Hidden Markov Model Approach Details

The purpose of a Markov model is to model random dynamic behavior with a mathematical method in which different states represent some aspect of the behavior. The basic, first-order Markov model has several states, connected by transitions among states. The likelihood of leaving each state is modeled by a probability distribution, and each state is modeled by a probability density function (PDF). For speech, these two sets of distributions (using the abbreviation PDF for both continuous and discrete distributions) attempt to model the variable timing and articulation of utterances, respectively. Very roughly, each state should correspond to a vocal tract shape (or equivalently a speech spectral envelope with formants resulting from the vocal tract shape). Each transition then can model the likelihood that the vocal tract moves from one position to another. In practice, the correspondence between vocal tract shape and HMMs is often weak.

The PDF for transitions out of a given state usually has a simple form: a relatively high likelihood of remaining in that state (e.g., 0.8), a smaller chance of moving to the next state in the time chain (e.g., 0.15), and a yet smaller chance of skipping to the state after the next one. The time course of speech does not allow backward transitions (e.g., we proceed through an HMM modeling a word, starting with its first sound and proceeding to its last sound). Because each state models roughly a vocal tract shape (or often an average of shapes), we must stay in each state for several 10 ms frames typically (hence the high self-loop probability). Allowing an occasional state to be skipped accounts for some variability in speech production, especially for rapid, unstressed speech; for example, the training speech may be clear and slow, thus creating states that need not always be visited in later (perhaps fast) test speech.

The transition PDF described here leads to an exponential PDF for the duration of state visits, which is an inaccurate model for actual phoneme durations. There is too much bias toward short sounds—very few phonemes last only 1–2 frames. Some more complicated HMM approaches allow direct durational modeling but at the cost of increased complexity.

In practice, every incoming speech signal is divided into successive 10 ms frames of data for analysis (as in speech coding). During the training phase, many frames are assigned to each given HMM state, and the state's PDF is simply the average across all assigned frames. Similarly, the PDF describing which state B follows any given state A in modeling the dynamics of the speech simply follows the likelihood of moving to a nearby vocal tract shape (B) in one frame, given that the previous speech frame was assigned to state A. This simple approach in which the model takes no direct account of the history of the speech (beyond one state in the past) is called a first-order Markov model. It is clear, when dealing with typical 10 ms frames, that there is certainly significant correlation across many successive frames (due to coarticulation in vocal tract movements). Thus the first-order assumption is an unrealistic approximation used to simplify the model and minimize computer memory. Attempts to use higher-order models have not been successful to date, because of the immense increase in complexity needed to accommodate reasonable amounts of coarticulation. The HMM is called hidden because the vocal tract behavior being modeled is not directly observable from the speech signal (if the input to an ASR system were x-rays of the actual moving vocal tract during speech, we could use direct Markov models, but this is far from practical).

The PDF to model states in an HMM is usually presumed to have the form of a Guassian PDF. Such a PDF for an N-dimensional feature vector $\mathbf{x}$ for a spoken word i is

$$P_i(\mathbf{x}) = (2\pi)^{-N/2}|W_i|^{-1/2}\exp\left[-\frac{(\mathbf{x}-\boldsymbol{\mu_i})^{\mathrm{T}}W_i^{-1}(\mathbf{x}-\boldsymbol{\mu_i})}{2}\right] \quad \text{(PDF)}$$

where W_i is the covariance matrix. $|W_i|$ is the determinant of W_i and $\boldsymbol{\mu_i}$ is the mean vector for word i. Most systems use a fixed W matrix (instead of individual W_i for each word) because (1) it is difficult to obtain accurate estimates for W_i from limited training data, (2) using one W matrix saves memory and computation, and (3) W_i matrices are often similar for different words. If one chooses independent parameters (i.e., no relationship among the numbers in the vector $\mathbf{x}$), the W matrix simplifies to a diagonal matrix, which significantly simplifies calculation of the PDF (which must be repeatedly done, for each speech frame and for each HMM). Thus many ASR systems assume (often without much justification, other than reducing cost) a diagonal W. Unless an orthogonalization procedure is performed, which is itself quite costly (e.g., a Karhunen–Loeve transformation), most commonly used feature sets (LPC coefficients, MFCCs) have significant correlation.

The elements along the main diagonal of W indicate the individual variances of the speech analysis parameters. The use of W_i^{-1} in the Gaussian PDF notes that those features with the smallest variances are the most useful for discriminating sounds in ASR. We usually try to choose parameter sets that lead to small variances and widely spaced means μ_i, so that similar sounds can be consistently discriminated.

The Gaussian form for the state PDF is often appropriate for modeling many physical phenomena; the idea follows from basic stochastic theory: the sum of a large number of independent, identically distributed random variables approaches a Gaussian PDF. Natural speech, coming from human vocal tracts, can be so treated, but only for individual sounds. If we try to model too many different vocal tract shapes with one HMM state (as occurs in multispeaker, or context-independent ASR), the Gaussian assumption is much less reasonable (see below for the discussion of mixtures of Gaussian PDFs).

HMMs can be used to model different units of speech. The most popular approaches are to model either phonemes or words. Phonemic HMMs require fewer states than word-based HMMs, simply because phonemes are shorter and have less spectral variation than words. If we ignore coarticulation with adjacent sounds, many phonemes could be modeled with just one state each. Inherently dynamic phonemes, such as stops and diphthongs, would require more states (e.g., a stop such as /t/ would need at least a state for the silence portion, a state for the explosive release, and probably another state

to model ensuing aspiration). There is a very practical issue of how many states are appropriate for each HMM. There is no theoretical answer; too few states lead to diffuse PDFs and poor discriminability (especially for similar sounds such as /t/ and /k/) because of averaging over diverse spectral patterns. Too many states lead to excessive memory and computation, as well as to *undertraining,* in which there are not enough speech data in the available training set to provide reliable model parameters for any given state.

Consider the case in which the (very practical) task simply distinguishes "yes" versus "no" (or perhaps just the ten digits: 0, 1, 2, . . ., 9); then it is efficient to create an HMM for every word in the allowed vocabulary. There will be several states in each HMM, to model the several phonemes in each word. When there is a rough correspondence between the number of states and the number of distinguishable sounds in the speech unit being modeled, HMMs seem to give good accuracy. If the model creation during the training phase is done well, this produces state PDFs with minimal variance. At recognition time, such tight PDFs will more easily discriminate words with different phoneme sequences.

As the size of the vocabulary increases, it becomes much less practical to have models for every word. Even in speaker-independent ASR, where we could ask thousands of different people to furnish speech data to model the many tens of thousands of words in any given language such as English, the memory and search time needed for word-based HMMs goes well beyond current computer resources. Thus, for vocabularies larger than 1000 words, most systems employ HMMs that model phonemes (or sometimes "diphones", which are truncated sequences of pairs of phonemes, used to model coarticulation during transitions between two phonemes). Diphones are obtained by dividing a speech waveform into phoneme-sized units, with the cuts in the middle of each phone (thus preserving in each diphone the transition between adjacent phonemes). In text-to-speech applications, when diphones are concatenated in a proper sequence (so that spectra match on both sides of a boundary), smooth spectral transitions usually result because the adjoining sounds at the boundaries are spectrally similar. For example, to synthesize the word straight, the six-diphone sequence /#s-st-tr-re-et-t#/ would be used (where # denotes silence).

Using phonemic HMMs has the big advantage of small memory and search time, as well as a fixed set of models, which does not have to be updated every time we add a word to the vocabulary. If the models are *context independent* (e.g., one model for each phoneme), however, their discriminability is small; the states modeling the initial and final parts of each phoneme will certainly result from a wide range of phonetic contexts, and the varying spectral patterns will cause broad PDF's of poorer discriminability.

Today, more advanced ASR systems employ *context-dependent* phonemic HMMs (of which diphone models are one example), in which each phoneme has many HMMs, depending on its immediate neighbors. A simple and common (but expensive) technique uses triphone models, where for a language with N phonemes we have N^3 models; that is, each phoneme has N^2 models, one for each context (occuring before and after the phoneme). There is much evidence from the speech production literature about the effects of coarticulation, which are significant even beyond a triphone window. (A diphone approach is a compromise between a simple context-independent scheme and a complex triphone method; it has N^2 models.) For English, with about 32 phonemes, the triphone method leads to tens of thousands of HMMs (which need to be adequately analyzed and stored in the training phase, and all must be examined at recognition time). Many systems adopt a compromise by grouping or clustering similar models according to some set of phonetic rules (e.g., the decision-tree approach). If the clustering is done well, one could even expand the analysis window beyond triphones, to accommodate further coarticulation (e.g., in the word "strew," lip rounding for the vowel /u/ affects the spectrum of the initial /s/; thus a triphone model for /u/ looking leftward only at the neighbor /r/ is inadequate).

Language Models for Automatic Speech Recognition

In the recent past, less successful ASR methods concentrated entirely on the acoustic input to make decisions. It was thought that all the required information to translate speech to text could be found in the speech signal itself; thus no cognitive modeling of the listener was necessary. Some simple experiments modeling language changed that approach, with positive results in recognition accuracy (11). Indeed, it can be easily observed that words in speech (as in text) rarely occur in random order. Both syntactically (word order structure) and semantically (meaning), there is much redundancy in word order. For example, when talking about a cat, one may well find the semantically related words "large" or "brown" just beforehand. As for syntax, there are many restrictions on English word sequences, for example, the common structure of article + adjective + noun for noun phrases, and strict order in verb phrases such as "may not have been eaten."

Thus researchers developed language models, in which the likelihood of a given word in the context of a prior sequence is evaluated and used in the decision process of speech recognition. The most common approach is that of a trigram language model, in which we estimate the probability that any given word in text (or speech) will follow the preceding two words (written or spoken). Textual redundancy in English (and indeed many languages) goes well beyond a trigram window of analysis, but practical issues of computer memory and the availability of training data have so far limited most models to a three-word window. We note here that training for language models is almost exclusively done on written texts (and rarely on transcriptions of speech), despite the inappropriateness of written compositions (i.e., most speech is spontaneous, and rarely from written texts), for the simple reason of the cost of transcribing large amounts of speech. Researchers find it much easier to use the many textual databases available today.

Trigram models incorporate various types of practical text redundancies automatically (no semantic or syntactic analysis is needed). This lack of intelligent analysis, however, leads to serious inefficiencies. As the allowed vocabulary increases, for general applications not limited to specific topics of conversation, the availability of sufficient text to obtain reliable statistics is a major problem. Depending on what we count as a word (e.g., do "eat, eats, eating, eaten" count as four?), there are easily hundreds of thousands of words in English; taking a conservative estimate of 10^5 words, there are 10^{15} trigrams. As a result, many acceptable trigrams (and even many bigrams and unigrams) do not occur in any training text (even

ones containing millions of words). The estimation of probabilities simply divides the number of occurrences of each specific trigram (or bigram) by the total number of occurrences of each word being modeled in the training text. For sequences found frequently, the estimates are reasonable, at least for the purposes of modeling (and recognizing speech from) text from the same source; one may often create different language models for different types of text, for example, medical and legal applications.

The serious problem with the above approach is what to do about unseen sequences (e.g., the very large number of trigrams not existing in a given training text, but legal nonetheless in the language). Assigning them all a probability of zero is inappropriate, because such word sequences will never appear in a recognizer's output, even when they are actually spoken. One approach simply assigns all unseen sequences a very small probability, and reduces the probabilities of all the others by a compensatory amount (to make the total probability equal to one). This is marginally better than assigning zero likelihoods, but treats all unseen sequences as equally likely, which is a very rough and poor estimate. A more popular way is the *back-off* approach, in which the overall likelihood for a word is a weighted combination of trigram, bigram, and unigram probabilities, with the weighting adjusted when estimating the likelihood of unseen trigram sequences. If no trigram (or bigram) estimate is available, we simply assign its weighting to be zero, and use existing (bigram and) unigram statistics. Of course, words that never appear at all in a training set have no statistics to back-off to; their likelihoods must be estimated in another way.

Although the trigram method has considerable utility, there is considerable waste as well. The memory for trigram statistics for large-vocabulary applications is quite large, and many probabilities are poorly estimated due to sparse training data. It is more efficient to cluster similar words together, to reduce memory and make the fewer statistics more reliable. One extreme case is the tri-POS (tri-part-of-speech) approach, where all words are classified by their simple syntactic category (such as noun, verb, preposition, etc.). Depending on how many categories are used, the statistics can be very much reduced in size while still retaining some powerful syntactic information and predicting common word-class sequences. Unfortunately, the semantic links across words are lost with this purely syntactic language model. One can easily extend the tri-POS method to windows of more than three words, due to the reduced model size, or one can extend the word categories to include semantic labels as well. Word-class language models using perhaps hundreds of hybrid syntactic-semantic classes may well be a good compromise approach between the overly-simple tri-POS way and the huge trigram approach.

Practical Issues in Automatic Speech Recognition

Segmentation. In addition to the serious issue of speech variability for ASR, there is another aspect of recognizing speech that makes the task more difficult than synthesizing speech: segmentation. In text-to-speech applications, the input text is clearly demarcated into separate words, and the listener must adapt to the accent of the synthetic speech. In ASR, on the other hand, the machine must adapt to the users' different styles of speech, and there are few clear boundaries in a speech signal. In particular, cues to word boundaries in speech are very few. Segmenting a long utterance into smaller units (ideally words) is hard, but it helps to reduce computation and lower error rates. Periods of silence are unreliable cues to linguistic boundaries. Long pauses are usually associated with sentence boundaries, but they often occur internal to a sentence, and even within words (as in hesitations). Short pauses may be confused with phonemic silences (i.e., closures during unvoiced stops).

Segmenting speech into syllables is easier due to the rise and fall of speech energy between vowels and consonants, but many languages (e.g., English) allow many different types of syllables (ranging from those with only a vowel to those with several preceding and ensuing consonants). Languages such as Japanese and Chinese are much easier to segment because of their consistent alternation of consonants and vowels. Segmenting speech into phonemes is also difficult if the language allows sequences of similar phonemes, such as vowels, or large variations in phonemic durations (e.g., English allows severe reduction of unstressed phonemes, such that recognizers often completely miss brief sounds).

Many commercial recognizers require speakers to adopt an artificial style of talking, pausing briefly (at least 0.25 s) after each word, to facilitate segmentation. Silences of more than 100 ms should not be confused with stop closures and allow reliable segmentation of speech into words. In order of increasing recognition difficulty, four styles of speech can be distinguished: *isolated-word* or *discrete utterance* speech, *connected-word* speech, *continuous* read speech, and *spontaneous* speech. The last two categories require little or no adaptation of speaking style. Continuous speech allows the most rapid input (e.g., 150–250 words/min) but is the most difficult to recognize. Requiring the speaker to pause after each word for isolated-word recognition is unnatural for speakers and slows the rate at which speech can be processed (e.g., to approximately 20–100 words per minute), but it alleviates the problem of isolating words in the input speech signal and eliminates coarticulation between words.

The search space for word units is much smaller than for longer utterances, and leads to faster and more accurate recognition. Unfortunately, few speakers prefer to talk in isolated word style. There is also the serious issue of *endpoint detection,* where the beginning and end times for speech must be determined; against a background of noise, it is often difficult to discriminate weak sounds from the noise. Endpoint detection is more difficult in noise: speaker-produced (lip smacks, heavy breaths, mouth clicks), stationary environmental (fans, machines, traffic, wind, rain), nonstationary environmental (music, shuffling paper, door slams), and transmission (channel noise, crosstalk). The variability of durations and amplitudes for different sounds makes reliable speech-versus-silence detection difficult; strong vowels are easy to find, but the boundaries between weak obstruents and background noise are often poorly estimated. Such endpoint location uses energy as the primary measure to cue the presence of speech, but also employs some spectral parameters as well. Endpoint decisions are required very often in isolated-word speech. For these reasons, most current research is focusing on more normal continuous speech.

Noise. In many applications, the speech signal is subject to distortion before it is received by a recognizer (13). Noise

may occur at the microphone (e.g., in a telephone booth on the street) or in transmission (e.g., fading over a portable telephone). This leads to less accurate parameters during speech analysis, and hence to poorer recognition, especially if there is a mismatch in conditions between the training and testing speech. It is very difficult to anticipate all distortion possibilities during training, and thus ASR accuracy certainly decreases in such cases.

In an attempt to normalize across varying conditions, many systems calculate an average parameter vector over several seconds (e.g., over the utterance), and then subtract this from each frame's parameters, applying the net result to the recognizer (13). This approach takes account of differences in average energy and the filtering effects of the transmission microphone and channel, which change slowly over time. Such "mean subtraction" is simple and useful in noisy conditions, but can add delay if we must determine the mean for several seconds of speech and only then start to recognize the speech. A similar method called RASTA uses a highpass filter (with a very low-frequency cutoff) to eliminate very slowly varying aspects of a speech signal, as being specific to the transmission channel and irrelevant to the speech message.

More traditional speech enhancement techniques can also be tried (14). These include spectral subtraction (or related Wiener filtering), where the average amplitude spectrum observed during estimated silent periods is subtracted from the spectrum of each speech frame. Thus stationary noise can be partly suppressed. Frequency ranges dominated by noise can be eliminated from consideration in the analysis to determine relevant parameters. Such processed speech sounds more pleasant, but intelligibility is not enhanced by this removal of noise.

Sometimes used for speech enhancement, "comb filtering" takes an F0 estimate for each frame and suppresses those parts of the speech between harmonics. Again the output speech sounds more pleasant, but this method requires a reliable F0 detector, and the resulting comb filter must dynamically follow changes in F0.

Much more powerful speech enhancement is possible if several microphones can record speech in a noisy environment. For example, in a noisy plane cockpit, one microphone close to the pilot's lips captures a signal containing speech plus some noise, whereas another outside the helmet captures a version of the noise corrupting the first signal. Using adaptive filtering techniques very similar to those for echo cancellation at 2/4 wire junctions in the telephone network, we can improve significantly the intelligibility of such speech.

Artificial Neural Networks

In the last decade, artificial neural networks (ANN) have attracted significant interest to attempt to solve problems of pattern recognition (15). These ANNs very roughly simulate the behavior of neurons in the human nervous system. As such, each node of an ANN typically accepts a set of inputs, computes a linear combination of weighted input values, compares the result against a threshold, and provides a binary output (1 if the linear combination of inputs exceeds the threshold, 0 otherwise). By combining such nodes in a network, quite complicated nonlinear decision spaces can be constructed, which have proven useful in numerous pattern recognition applications, including speech recognition (15). The ANN accepts a long vector of L parameters (e.g., for 10 MFCCs over a 100 frame utterance, we have a vector of $L = 1000$ dimensions), coming from the usual speech analysis methods discussed above, and these numbers provide the input to a set of M nodes. Each node makes a linear combination of a selected set of values from the input vector. The conceptually simplest method is to assign one such output node to each possible output of the recognition process (e.g., if the utterance must be one of the ten spoken digits, $M = 10$), and we design the weights in the ANN such that only one of the M outputs will be 1; the label on that output node will provide the textual output. Thus, with proper training of the ANN to select the appropriate weights, when one says "six" and the 1000 resulting analysis parameters are fed through the ANN, ideally only one of the 10 output nodes (i.e., the one corresponding to "six") will show a 1 (the others will show a 0). This assumes, however, that each of the vocabulary words has a simple distribution in L-dimensional speech space, such that simple hyperplanes can partition the space into 10 appropriate clusters (and furthermore that a training algorithm examining many utterances of the ten digits can determine the correct weights). It is very rare that these assumptions are true, however. A partial solution has been to add on additional layers of nodes in the ANN. We allow the number M of nodes in the second layer to be larger, and allow their binary outputs of those M nodes to feed through another layer of weights and N nodes. Such a double-layer ANN can partition the speech space into the much more arbitrary shapes corresponding to practical systems. In general, for speech it appears that a three-tier system is needed: the original input vector feeds the lowest layer, and the information propagates through three weightings (two hidden layers of nodes) to finally appear at the output layer (with one node for each textual label).

For static patterns, ANNs have provided excellent classification when properly trained. However, speech is dynamic, and the success of ANNs has been more limited for ASR. So-called time-delay ANNs feedback values within ANN layers to try to account for the fact that small delays (variability in timing) in speech signals occur often in human speech communication and cause no difficulty in perception. Shifting or scaling the set of inputs to a basic ANN, however, usually causes changes in the ANN outputs. As a result, ANNs have typically found application in ASR, not as a replacement for HMMs, but as an additional helper to the basic HMM scheme (e.g., ANNs are sometimes used in the training phase for better parameter estimation).

SPEAKER VERIFICATION

Speech analysis has been shown to be useful in coding, where we regenerate speech after efficient compression, and in ASR, where we convert it to text. Another application is speaker verification, where we determine if speakers are who they claim to be. Using fingerprints or retical scans may be more accurate in identifying people, but identification via speech is much less intrusive and is feasible over the telephone. Many financial institutions, as well as companies furnishing limited access to computer databases, would like to provide automatic customer service by telephone. Since personal number codes

(keyed on a telephone pad) can be lost, stolen, or forgotten, speaker recognition might provide a viable alternative. The decision process can be much simpler than for ASR, because a given speech signal is converted into a simple binary digit (yes, accept the claim, or no, refuse it). However, the accuracy of such verifiers has only recently achieved the point of commercialization.

In ASR, variation due to different speakers in speech signals corresponding to the same spoken text is viewed as noise to be either eliminated by speaker normalization or (more commonly) accommodated through HMMs drawn from a large number of speakers. When the task is to identify the person talking rather than what the person is saying, the speech signal must be processed to extract measures of speaker variability instead of segmental features. What to look for in a speech signal to distinguish speakers is much more complex than for ASR. In ASR we look for spectral features to distinguish different basic vocal tract positions and can exploit language models to raise accuracy. One common verification approach indeed employs ASR techniques, except associating models with speaker names instead of word or phoneme labels. When two speakers utter the same word or phoneme, we compare timing and spectral patterns to see which speaker provides the more precise match. In ASR, the competing phoneme or word candidates are often fairly distant in spectral space, making the choice relatively easy. In verification, however, many speakers are quite similar in vocal tract shape and in speaking style. However, analysis over many frames renders the task feasible.

For speech recognition, much is known about the speech production process linking a text and its phonemes to the spectra and intonation of a corresponding speech signal. Each phoneme has specific articulatory targets, and the corresponding acoustic events have been well studied (but still remain far from being fully understood). For speaker recognition, the acoustic aspects of what characterizes the differences between voices are obscure and difficult to separate from signal aspects that reflect ASR. There are three sources of variation among speakers: differences in vocal cords and vocal tract size and shape, differences in speaking style (including variations in both target positions for phonemes and dynamic aspects of coarticulation such as speaking rate), and differences in what speakers choose to say. Automatic speaker recognizers exploit only the first two variation sources, examining low-level acoustic features of speech, since a speaker's tendency to use certain words and syntactic structures (the third source) is difficult to quantify or control in an experiment.

Unlike the clear correlation between phonemes and spectral resonances, there are no acoustic cues specifically or exclusively dealing with speaker identity. Most of the parameters and features used in speech contain information useful for the identification of both the speaker and the spoken message. The two types of information, however, are coded quite differently. Unlike ASR, where decisions are made for every phone or word, speaker recognition requires only one decision, based on the entire test utterance, and there is no simple set of acoustic cues that reliably distinguishes speakers. Speaker recognizers typically utilize long-term statistics averaged over whole utterances or exploit analyses of specific sounds. The latter approach is common in *text-dependent* applications where utterances of the same text are used for training and testing.

There are two classes of errors in any binary classification problem, such as speaker verification: *false rejections* and *false acceptances.* A false rejection occurs when the system incorrectly rejects a true speaker (one whose identity claim is true). A false acceptance occurs when the system incorrectly accepts an impostor. The decision to accept or reject usually depends on a threshold: if the distance between a test and a reference pattern exceeds the threshold (or equivalently, using a PDF for the claimed speaker, the value is too low), the system rejects a match. Depending on the costs of each type of error, systems can be designed to minimize an overall cost by biasing the decisions in favor of less costly errors. Low thresholds are generally preferred because false acceptances are usually more expensive (e.g., admitting an impostor to a secure facility might be disastrous, whereas excluding some authorized personnel is usually only annoying). Many researchers adjust system parameters so that the two types of error occur equally often. This is called an *equal error rate condition.*

A recent popular method is called Gaussian mixture models (16), which follows the popular modeling approach in ASR, but eliminates separate HMMs for each phoneme as unnecessary. A GMM is essentially a one-state Markov model, but allows the PDF to be complicated. In ASR, we can often justify that each state's PDF may be approximated by a Gaussian distribution (especially for triphone HMMs, where context is well controlled). When creating one PDF for all speech from a speaker, however, the PDF is far from Gaussian. To retain the simple Gaussian statistics (complete specification by only the mean and variance), however, both ASR and speaker verification often allow a state's PDF to be a weighted combination of Gaussian PDFs. Using the same set of Gaussians for all speakers, each speaker is characterized by the corresponding set of weights. Evaluation is thus quite rapid, even when all the frames of an utterance contribute to the overall speaker decision.

Such an approach ignores dynamic speech behavior (as well as ignoring intonation, as do ASR systems) and further assumes that both training and testing utterances are roughly similar in phonetic composition. This latter can be assured by text-dependent verification, where the speaker is asked to utter words already used during the training phase (e.g., one commercial system trains on two-digit numbers, such as "74," and then asks each candidate speaker to utter a few such numbers at test time). Accuracy is much higher for these systems than for text-independent verifiers, although the latter are needed for forensic work (e.g., there is no text control in wiretapped conversations). The former, however, that risk an imposter may play a recording of the desired speaker (over the telephone, or if there is no camera surveillance at a secure facility). Such impostors, of course, cause problems with text-independent verifiers (which have lower accuracy) as well, but imposters could not anticipate the requested text to speak with the much larger vocabulary in the latter systems.

In any pattern recognition task (including ASR), training and test data should be kept separate, but this is all the more important in the case of speaker recognition. Speakers tend to vary their style of speech over time (e.g., morning versus evening, Monday versus Saturday, healthy versus ill). Train-

ing a speaker verifier in one session, and then testing a few months later rarely gives good results. Ideally, speaker recognizers need months of training (or at least an adaptive system, allowing periodic updates while being used). If the same utterances are used to train and test a recognition system, artificially high accuracy results, because the model parameters are often heavily tuned toward the training data. For good results, the training data must be sufficiently diverse to represent many different possible future input speech signals. With shared training and test data, it is difficult to know whether the system has been designed to take advantage of specific speech or speaker characteristics, that may not be reliable for new data. Given K utterances per speaker as data, one common procedure trains the system using $K - 1$ as data and one as test, but repeats the process K times treating each utterance as test once. Technically, this leave-one-out method designs K different systems, but it verifies whether the system design is good using a limited amount of data, while avoiding the problems of common training and testing data.

TEXT-TO-SPEECH SYNTHESIS

Our last speech processing application concerns generation of speech from text (17–19). Here the speech processing occurs both in the development stage of the system, where we record spoken units from one or more speakers, and again at synthesis time, where we concatenate selected stored units to produce the synthetic output voice. The units involved range from full sentences (e.g., talking cars and ovens) to shorter phrases and words (e.g., telephone directory assistance) to phonemes (e.g., for unlimited text applications).

We distinguish here true text-to-speech (TTS) systems, which accept any input text in a chosen language (including new words and typographical errors), from voice-response systems of very limited vocabulary. The latter are essentially voice coders of much simpler complexity but are also inflexible and quite limited in applications. The recent increase in commercial synthesizers is due to both advances in computer technology and improvements in the methods of speech synthesis.

The simplest applications with small vocabularies are just speech coders, playing back the speech when needed. Simple playback is impossible for general TTS, because no speaker can record all possible utterances. In the latter case, small units such as phonemes or diphones are concatenated, and significant adjustments must be made at unit boundaries to avoid highly disjointed (jumpy) speech. The number and complexity of such adjustments vary in indirect proportion to the number of boundaries, which in turn also varies inversely to quality (e.g., TTS using smaller units sounds less natural).

The critical issues in current synthesis research concern tradeoffs among the conflicting demands of maximizing speech quality and minimizing memory space, algorithmic complexity, and computation speed. Although simple TTS is possible in real time with low-cost hardware, there is a trend toward using more complex programs (tens of thousands of lines of code, megabytes of storage) to improve quality. Text-to-speech systems constructively synthesize speech from text using linguistic processing and concatenating small speech units (e.g., phonemes). Real-time TTS systems produce speech that is generally intelligible, but lacks naturalness (19).

Most synthesizers reproduce speech of bandwidth ranging from 300–3000 Hz (e.g., for telephone applications) to 100–5000 Hz (for higher quality). Frequencies up to 3 kHz are sufficient for vowel perception, because vowels are adequately specified by the formants F_1–F_3. The perception of some consonants, however, is slightly impaired if energy in the 3 to 5 kHz range is omitted (6). Frequencies above 5 kHz are useful to improve speech clarity and naturalness but do little to aid speech intelligibility. If we assume that the synthesizer reproduces speech up to 4 kHz, a rate of 8000 samples per second is needed. Because linear PCM requires 12 bits per sample for toll-quality speech, storage rates near 100 kbits/s result, which are prohibitive except for synthesizers with very small vocabularies.

The memory requirement for a simple synthesizer is often proportional to its vocabulary size. Because of decreasing memory costs, it is less imperative to minimize memory than computational complexity. Nonetheless, storing all possible speech waveforms (even with efficient coding) for synthesis purposes is impractical for TTS. The sacrifices usually made to reduce complexity and memory for large-vocabulary synthesizers involve simplistic modeling of spectral dynamics, vocal tract excitation, and intonation. Such modeling yields quality limitations that are the primary problems for current TTS research (6).

Steps Necessary to Produce Speech from Text

A series of steps are required in TTS to convert the textual message into an acoustic message. The linguistic processing is, in a sense, an inverse of the procedures for ASR. First, a preprocessing stage normalizes the input text so that it is a series of spelled words (retaining punctuation marks). Thus all abbreviations and digits are converted to words, typically by a look-up table or simple programs (sophisticated systems, however, might distinguish the several ways of saying “1997”, “$19.97” and “1,997”); word context may be necessary to handle cases like “St. Mark St.” Then, the words are converted into a string of phonemes (and often rudimentary intonation parameters as well), usually with a combination of a dictionary and pronunciation rules. With the continuing reductions in cost of computer memory, it is popular to use a large dictionary containing most words in the chosen language and their phonemic pronunciations. This way has the advantage that additional information can be stored in the dictionary, including lexical stress (i.e., which syllable is stressed), part-of-speech, and possibly even semantics.

The alternative approach of letter-to-phoneme rules has the advantage of handling new or foreign words, as well as mistyped words, for which few dictionaries are suitable. Complex languages like English, which is derived from both Romance and Germanic languages, require many hundreds of these rules to pronounce letter sequences correctly (e.g., consider the ways of uttering “-ough-:” rough, cough, though, through, thought, drought). Other languages are much simpler; for example, Spanish employs only one rule per letter. In any event, developing TTS capability requires establishing a dictionary and pronunciation rules for the language. Hence commercial TTS exists only for approximately 20 of the world’s languages, whereas many commercial ASR systems (those based on word units and without a language model) can handle virtually all languages, because most languages

employ similar versions of stops, fricatives, vowels, and nasals.

The next TTS step is intonation assignment, specifying a duration and amplitude for each phoneme, as well as an F_0 pattern for the utterance. This is much more difficult than the prior two TTS steps, and requires a syntactic and semantic analysis of the input text. Most languages use intonation in complex and different ways to cue many aspects of speech communication. Different classes of phonemes have different durational trends (e.g., a vowel tends to be longer when followed by a voiced consonant). Many languages (including English) stress only a small number of syllables in an utterance. Determining where to stress involves not just lexical stress from a dictionary, but also a judgment about the semantic importance of the words in a sentential context; for synthetic speech comparable to that of humans, this would require a natural language processor beyond current capabilities. Syntactic structure is often cued via intonation in natural speech; for example, in English, long word phrases often start with an F0 rise and end with a fall. Questions in many languages are cued with a large final F0 rise, if they request a yes/no answer (but not questions with the "wh" words: what, when, why, who, how). Tone languages (e.g., Chinese, Thai) employ four or five different F0 patterns to distinguish different words with the same phoneme sequence. Lastly, speech uttered with emotion often has large changes in intonation.

The last TTS step concatenates speech units, using the specified intonation, and does the necessary adjustments to the model parameters at unit boundaries. Few manipulations are needed for phrasal concatenation, but smaller units require smoothing parameters across the boundary, for several frames. Such smoothing is relatively straightforward when the units contain spectral parameters, such as LPC coefficients or formants, although improved quality occurs with more complicated smoothing rules. Storing small units with waveform coding (e.g., log PCM or ADPCM) is often not suitable (despite the higher general quality of such speech) because smoothing the available parameters does not approximate well the coarticulation and F0 patterns found in human speech production.

Smoothing of spectral parameters at unit boundaries is most important for short units (e.g., phonemes), because of the large number of boundaries per second in the synthesized speech. For units of equal size, smoothing is much simpler when the joined units have approximately matching spectra at the boundaries. Because diphones concatenate spectra from similar sections of two realizations of the same phoneme, their smoothing rules are simple. Systems that link phonemes, however, must use complex smoothing rules to represent coarticulation. Not enough is understood about coarticulation to establish a complete set of rules that describe how the spectral parameters for each phoneme are modified by its neighbors. Diphone synthesizers try to circumvent this problem by storing the parameter transitions from one phoneme to the next since coarticulation primarily influences only the immediately adjacent phonemes. However, coarticulation often extends over several phonemes. Using only average diphones or those from a neutral context leads to lower-quality synthetic speech. Improved quality is possible by using multiple diphones depending on context, effectively storing triphones of longer duration (which may substantially increase memory requirements). Some coarticulation effects can be modeled by simple rules, such as lowering all resonant frequencies during lip rounding, but others such as the undershoot of phoneme target positions (which occurs in virtually all speech) are much harder to model accurately.

It is typically because of this last step that TTS yields poorer quality than many speech coders, because TTS is forced to employ synthetic-quality coding techniques. The basic excitation for vocal tract filters in TTS (either LPC or formant-based approaches, which constitute most commercial methods) is a simple train of periodic pulses (for voiced speech) or white noise (often using a random number generator for simplicity) for unvoiced speech. The combination of overly simple excitation and the limited modeling accuracy of the LPC or simple formant models has led to intelligible, but slightly unnatural, synthetic speech (6). Use of more complicated excitation waveforrrms can raise quality.

Recently, some commercial systems have been successful in concatenating waveform-coded small units, with limited numbers of perceptually annoying spectral jumps at unit boundaries; for example, the pitch-synchronous overlap-and-add (PSOLA) method outputs successive smoothed pitch periods (20). While such speech sounds more natural, PSOLA cannot easily produce alternative voices. It is typically based on one speaker uttering a large inventory of diphones; for a language like English with about 32 phonemes, about 1000 diphones must be uttered. One cannot simply adjust some synthesizer parameters as uniformly as possible to get other synthetic voices (as is possible with formant synthesizers).

CURRENT TECHNOLOGY

Today's speech coders can deliver toll quality (i.e., equivalent to the analog telephone network) at 8 kbits/s with minimal delay (and even at 4 kbits/s, if delay is not an issue). The favored approach is that of CELP, for which there are several standards accepted internationally (e.g., for use in digital cellular telephony) (5). The digital links in most telephone networks still employ simpler 64 kbits/s log PCM coding; 24–32 kbits/s ADPCM and delta modulation are also still popular because of their relative simplicity, compared to CELP. The last few years have seen significant reductions in transmission rate; a decade ago, toll quality was limited to 16 kbits/s and above. We are still far from an ultimate limit of 100 bit/s. Despite the reducing cost of computer memory and speed, further research in coding is needed because of the increasing use of wireless telephony, where limited bandwidth is an important factor.

The lack of standards in the area of human–machine applications for speech (i.e., synthesis and recognition) has hindered easy comparisons across commercial systems. Nonetheless, we can say that several companies offer unlimited text-to-speech for several languages (typically the major European languages, as well as Japanese and Chinese). The synthetic speech is largely intelligible, but is easily discerned as synthetic and lacking naturalness. The remaining problems for TTS lie in the areas of excitation, coarticulation, and intonation. The majority of synthesizers (both LPC and formant-based) employ a source-filter model of the vocal tract, where the excitation is modeled with few parameters and only coarsely approximates the actual residual signal. Relatively simple models for the movements of vocal tract resonances

lead to an inadequate representation of coarticulation. Waveform-based synthesizers, on the other hand, do not always accept boundary smoothing well. All synthesizers suffer from our lack of understanding of the complex relationships between text and intonation. Thus synthesizers are clearly in use for many languages, but wider public acceptance awaits improvements in quality.

People have always been very impressed by a machine able to recognize what they say. Speech recognizers are increasingly entering the commercial market, but their severe limitations (compared to humans) will continue to hinder wider acceptance. The need to pause between words, restrict the choice of words, and do prior training, as well as frequent recognition errors, have significantly limited the use of ASR. Systems eliminating all these restrictions (i.e., continuous, speaker-independent, large-vocabulary ASR) still suffer from high cost and frequent errors, especially if they are used in noisy or telephone environments, conditions that occur often in practical applications. Progress in ASR in recent years has been more attributed to general improvements in computers and to the wider availability of training data, than to algorithmic breakthroughs. The basic HMM approach using MFCCs largely was developed more than 15 years ago (11). Even more recent additions, such as delta coefficients, mean subtraction, Gaussian mixtures, and language models, have been in wide use for several years.

Future systems will likely integrate more structure into the stochastic approach. It is clear that the expert-system approach to ASR common in the early 1970s will never replace stochastic methods, because individual human phoneticians can never assimilate enough information from hundreds of hours of speech; probabilistic computer models can improve with larger amounts of training data. The simple stochastic models in current widespread ASR use, however, are too unstructured and allow too much freedom (similar deficiencies hold for more recent neural network approaches to ASR). For example, the MFCCs, although appropriately scaling the frequency axis to account for perceptual resolution, do not take account of the wide perceptual difference between resonances and spectral valleys. First-order HMMs ignore the high degree of correlation across many frames of speech data (compensating by using delta coefficients is only a very rough use of speech dynamics). Intonation is widely ignored (e.g., F0) or treated as noise (e.g., durational factors), despite solid evidence of its use in human speech perception. Of course, in a practical world, you use whatever works, and current systems, despite their flaws, provide sufficiently high accuracy for small vocabularies (e.g., recognizing the digits in spoken telephone or credit-card numbers, or controlling computer menu selections by speech). Practical use of speech in telephone dialogs must await advances in both the quality of synthetic speech and the recognition accuracy of spontaneous conversations.

BIBLIOGRAPHY

1. J. Deller, J. Proakis, and J. Hansen, *Discrete-time Processing of Speech Signals,* Englewood Cliffs, NJ: Prentice Hall, 1993.
2. P. Noll, Digital audio coding for visual communications, *Proc. IEEE,* **83**: 925–943, 1995.
3. J-P. Adoul and R. Lefebvre, Wideband speech coding, in W. Kleijn and K. Paliwal (eds.), *Speech Coding and Synthesis,* New York: Elsevier Science, 1995, Ch. 8.
4. J. Johnston and K. Brandenburg, Wideband coding—perceptual considerations for speech and music, in S. Furui and M. Sondhi (eds.), *Advances in Speech Signal Processing,* New York: Marcel Dekker, 1992, pp. 109–140.
5. A. Gersho, Advances in speech and audio compression. *Proc. IEEE,* **82**: 900–918, 1994.
6. D. O'Shaughnessy, *Speech Communication: Human and Machine.* Reading, MA: Addison-Wesley, 1987.
7. P. Kroon and W. Kleijn, Linear predictive analysis by synthesis coding, in R. Ramachandran and R. Mammone (eds.), *Modern Methods of Speech Processing,* Norwell, MA: Kluwer, 1995, pp. 51–74.
8. A. Spanias, Speech coding: a tutorial review, *Proc. IEEE,* **82**: 1541–1582, 1994.
9. R. Cox and P. Kroon, Low bit-rate speech coders for multimedia communication, *IEEE Comm. Mag.,* **34**(12): 34–41, 1996.
10. W. Hess, Pitch and voicing determination, in S. Furui and M. Sondhi (eds.), *Advances in Speech Signal Processing,* New York: Marcel Dekker, 1992, pp. 3–48.
11. L. Rabiner and B. Juang, *Fundamentals of Speech Recognition,* Englewood Cliffs, NJ: Prentice-Hall, 1993.
12. J. Piconi, Signal modeling techniques in speech recognition, *Proc. IEEE,* **81**: 1215–1247, 1993.
13. J-C. Junqua and J-P. Haton, *Robustness in Automatic Speech Recognition,* Norwell, MA: Kluwer, 1996.
14. Y. Ephraim, Statistical-model-based speech enhancement systems, *Proc. IEEE,* **80**: 1526–1555, 1992.
15. N. Morgan and H. Bourlard, Continuous speech recognition, *IEEE Signal Proc. Mag.,* **12** (3): 25–42, 1995.
16. D. Reynolds and R. Rose, Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans. Speech Audio Process.* **SAP-3**: 72–83, 1995.
17. D. Klatt, Review of text-to-speech conversion for English. *J. Acoust. Soc. Am.,* **82**: 737–793, 1987.
18. J. van Santen, Using statistics in text-to-speech system construction. *Proc. ESCA/IEEE Workshop on Speech Synthesis,* **2**, 240–243, 1994.
19. T. Dutoit, *From Text to Speech: A Concatenative Approach,* Norwell, MA: Kluwer, 1997.
20. E. Moulines and F. Charpentier, Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones, *Speech Commun.,* **9**: 453–467, 1990.

DOUGLAS D. O'SHAUGHNESSY
INRS-Telecommunications

SPEECH PROCESSING. See also COMPANDORS.