APPLICATIONS OF DYNAMIC PROGRAMMING TO SPEECH AND LANGUAGE PROCESSING

Chin-Hui Lee

Chin-Hui Lee *is a member of technical staff in the Speech Research Department at AT&T Bell Laboratories, Murray Hill, New Jersey. He does research on speech recognition and on speech and signal modeling. He has a B.S. from National Taiwan University, an M.S. from Yale University, and a Ph.D. from the University of Washington, all in electrical engineering. He joined AT&T in 1986.*

Dynamic programming (DP) is a mathematical programming technique for solving certain sequential optimization problems. In this paper, we focus our discussion on DP applications in the areas of speech and language processing. Because of the sequential nature of speech and language events, most applications can generally be formulated as a sequential decision process. DP can then be used to obtain the solution to these processes by decomposing them into a series of smaller local problems and solving them sequentially. It is impossible to give adequate treatment to each such application in this paper. Instead, we briefly illustrate the basic way in which DP techniques have been applied to several key areas of speech and language research in the Information Principles Research Laboratory. The applications discussed in this paper include speaker verification, speech recognition, speech synthesis, speech analysis, speech modeling, speech enhancement, speech segmentation, speech coding, speech transmission, language modeling, and natural language processing. These applications have been proven successful in enhancing human-machine communications.

Introduction

Dynamic programming (DP) is a mathematical programming technique for solving certain optimization problems. Since its introduction in the 1950s by Bellman,[1] DP has been successfully applied to a number of problems, including optimal control, optimal resource allocation, optimal string alignment, and network optimization, to name just a few. In this paper, we focus our discussion on the applications of DP to speech and language processing.

The ways in which the techniques of DP have been applied to speech and language processing are numerous. It is impossible to give adequate treatment to each such application in this paper. Instead, it is our goal to illustrate briefly the basic way in which DP techniques have been applied to several key areas of speech and language research. As nearly as we can determine, the first application of DP to speech processing was outlined by Slutzker.[2] Since that time, the number of proposed applications has grown so rapidly that it is safe to say that, in one form or another, the ideas of dynamic programming have pervaded the methodology of almost every aspect of speech signal processing—particularly the areas of speech and speaker recognition.

Speech signals are often modeled, in the time domain, as a sequence of segments, and these segments are generally modeled as a sequence of feature vectors. The feature vector typically consists of either spectral parameters, temporal parameters, or some suitable combination. Markov models have been used to characterize the statistical properties of the feature vectors. Because of this sequential nature, a speech signal can be considered a multistage process. In each stage, we have a frame or a segment of speech, which is characterized by a sequence of speech codewords or models. Most speech applications are then formulated as string alignment problems, in which a sequence of input speech segments (or frames) is matched with a collection of speech models to determine the optimal speech code sequence or to perform speaker or speech recognition. For language processing, there is also a sequential nature of words, phrases, sentences, and paragraphs appearing in text. Stochastic language models have been successfully formulated to perform syntactic and semantic analysis of text and speech.

In this paper we review some of the DP applications to speech and language processing research done in the Information Principles Research Laboratory. Most of the applications presented herein have been pioneered by the people listed under "Acknowledgments," who have used them in many successful laboratory experiments and in some commercial products. Other applications are still in the research stage, but we reasonably expect that they will be perfected and will become part of the evolving technology. The reader is referred to special *AT&T Technical Journal* issues on speech processing technology[3] and artificial intelligence[4] for a broad perspective on other issues related to speech and language processing technologies.

This paper is organized as follows. In the following section, we give an overview of the dynamic programming formulation that is common to all applications discussed in this paper. We then briefly present the essentials of a number of DP applications to speech processing (the applications include speaker verification, speech recognition, speech synthesis, speech analysis, speech modeling, speech enhancement, speech segmentation, speech coding, and speech transmission). Following that, we present a DP-based language processing application for assigning the most likely sequence of parts of speech to a sentence. Finally, we discuss ways of applying these speech and language processing techniques to future research in the areas of automatic speech recognition and natural language processing.

Dynamic Programming Formulation of Sequential Problems

Consider an N-stage *decision process* with *state* vector $\mathbf{x}$, and *decision* vector $\mathbf{q}$. At each stage i, we denote the decision by $\mathbf{q}_i$ and the state vector after decision $\mathbf{q}_i$ has been made by $\mathbf{x}_i$. A sequence of decisions (called a *policy*), is generally expressed as a sequence of transformations T_i such that $\mathbf{x}_{i-1} = T_i(\mathbf{q}_i, \mathbf{x}_i)$, for which the sequence of the decisions and the states $\{\mathbf{q}_i, \mathbf{x}_i\}$ corresponding to a policy can be retrieved by a traceback procedure. At each stage, the decision is based on optimizing a well-defined *objective function* over a set of *admissible* policies. In this paper, we focus our attention on *additive* objective functions; i.e., the objective function at the current stage can be expressed as the sum of the *contributions* from the current and previous stages. The

contribution $c_i(\mathbf{q}_i, \mathbf{x}_i)$ at stage i is usually composed of two parts: (1) the transitional cost of going from state $\mathbf{x}_{i-1}$ at stage $(i-1)$ to state $\mathbf{x}_i$ at stage i, denoted by $a_i(\mathbf{x}_{i-1}, \mathbf{x}_i)$; and (2) the state occupancy cost at stage i, denoted by $d(i, \mathbf{x}_i)$, which does not depend on the state of the previous stage $\mathbf{x}_{i-1}$. The problem of determining the optimal sequence $\{\mathbf{q}_i, \mathbf{x}_i\}$ over a discrete time period (or a finite grid) is then formulated as

$$\underset{\{\mathbf{q}_i, \mathbf{x}_i\}}{\text{optimum}} \sum_{i=1}^{N} c_i(\mathbf{q}_i, \mathbf{x}_i) = \underset{\{\mathbf{q}_i, \mathbf{x}_i\}}{\text{optimum}} \sum_{i=1}^{N} \left[d(i, \mathbf{x}_i) + a_i(\mathbf{q}_i, \mathbf{x}_i) \right] \quad (1)$$

DP provides a method of determining the optimal solution to problem (1) in a sequential manner. This approach is based on the *principle of optimality*,[1] which states, "An optimal policy has the property that whatever the initial state and the initial decision, the remaining decisions must constitute an optimal policy with regard to the new state of the process resulting from the first decision." This principle of optimality is applied to generate a set of *functional (recurrence) equations.* Assume that the optimal policy at stage $(i-1)$ has been found and the optimal objective function is denoted by $D(i-1, \mathbf{x}_{i-1})$. Then, for every state, the optimal objective function at stage i can be expressed in a functional equation of the form

$$D(i, \mathbf{x}_i) = \underset{\mathbf{q}_i}{\text{optimum}} \left[D(i-1, \mathbf{x}_{i-1}) + c_i(\mathbf{q}_i, \mathbf{x}_i) \right] \quad (2)$$

where $\mathbf{x}_{i-1} = T_i(\mathbf{q}_i, \mathbf{x}_i)$. With initial condition $D(0, \mathbf{x}_0) = 0$, the functional equation (2) can be constructed systematically at each stage and solved by successive iteration. This recursive computational procedure for solving multistage decision problems is a direct consequence of the principle of optimality.

An important feature of the DP recursion is that the optimization at stage i does not require the values of the decision variables which determine the value of the optimal objective function $D(i-1, \mathbf{x}_{i-1})$. This "bootstrapping" nature of the DP recursion means that, at each stage, the optimal policy depends on a small number of decision variables, and only the values of the objective functions need to be retained for further calculation.

In the following sections we focus our attention on several applications of DP in the areas of speech and language processing. Each application has been formulated as a problem of finding the optimal path of a decision network in an appropriate parameter space. The discussion will emphasize the sequential nature of the applications, the mapping of the optimization problems into multistage decision processes, and the formulation of the DP functional equation for each application. Details of the algorithms presented can be found in the references given in each section.

Speaker and Speech Recognition—Dynamic Time Warping

In both automatic speech and speaker recognition, the central task is the comparison of a pair of utterances, one unknown and the other a labeled prototype. Speech utterances are analyzed and represented as sequences of feature vectors. Labeled prototypes are usually actual sequences of feature vectors referred to as *reference template patterns.* A fundamental problem in the central comparison task occurs when the comparison is between two tokens of the *same* utterance. Repeated utterances of the same word or phrase can never be expected to have precisely the same duration, nor can corresponding events in each utterance be expected to occur at precisely the same relative times. For this reason, it is crucial to find an optimal alignment in time between such utterances so that corresponding events can be compared precisely. In the following we outline DP-based procedures for time alignment and then describe and differentiate the speech and speaker recognition tasks.

The time alignment problem is generally formulated as a graph search algorithm; i.e., we find the best path through a finite grid, in which the axes of the grid

correspond to the time scales of the unknown and the reference speech patterns. The specific procedure that is generally used has been called *dynamic time warping* (DTW).[5] The DP functional equation for determining the best path through the grid is of a form similar to equation (2), i.e.,

$$D(i, j) = d(i, j) + \min_{\{k\}} \left[D(i-1, k) + a(k, j) \right] \quad (3)$$

In the above formulation, i is the frame index for the unknown (test) pattern, and it corresponds to the stage index i in equation (2); j is the frame index of the reference pattern used in the match, and it corresponds to the state $\mathbf{x}_i$ in equation (2). The set $\{k\}$, corresponding to the state space $[\mathbf{x}_{i-1}]$ at stage $(i-1)$, consists of the set of frames from which a valid path to frame j can come. $D(i, j)$ is the (global) accumulated distance to the grid point (i, j); $d(i, j)$ is the local spectral distance between frame i and frame j, and this distance corresponds to the state occupancy cost $d(i, \mathbf{x}_i)$ in equation (1). Finally $a(k, j)$ is the penalty associated with a transition from frame k to frame j, and this penalty corresponds to the state transition cost $a_i(q_i, \mathbf{x}_i)$ in equation (1). Typically, $\{k\}$ is chosen to ensure that the overall path or warping function is monotonically increasing with appropriate limits on the maximum and minimum slopes, i.e., the range of warping that is allowed. Equation (3) is solved sequentially for $i = 1, \ldots, T$ (where T is the length, in frames, of the test pattern), over the range of j from 1 to R (where R is the length, in frames, of the reference pattern). The initial conditions for the DP recursion are defined by setting $D(0, 1) \equiv 0$, and setting $D(0, k)$ to be infinity for $k \neq 1$. In Figure 1, we show an example of how the DTW procedure is used for aligning two utterances corresponding to the same word. In the top left and the bottom of the figure, we plot the energy contours of the reference and the test patterns respectively. In the top right, we show the resulting optimal alignment path. The dashed lines indicate good alignment of the energy peaks in the two utterances.

For speech recognition, an unknown speech sample is compared with each one of a labeled inventory of speech prototypes. Each prototype may represent speech segments, typically isolated words or phrases. An overall spectral distance is computed for each comparison after appropriate time alignment. The unknown speech sample is identified with the label of the prototype having the smallest time-aligned distance. For speaker recognition, however, the speech prototypes are labeled by talker. Two types of speaker recognition systems are referred to in the literature, namely speaker identification and speaker verification.[6,7] Those two systems differ with respect to the decision task. In speaker identification, the decision task is analogous to the decision usually carried out in speech recognition, namely choosing the prototype which best matches the input sample. In speaker verification, an identity claim is either asserted or solicited from the user. The decision task is to determine whether or not the input sample matches the prototype associated with the speaker (whose identity is claimed) well enough to accept the claim. The decision is made by comparing the overall distance measured in the prototype comparison with a threshold distance. The application area for which speaker verification is usually considered is control of access to privileged information, transactions, or premises. The practical significance of the verification task, as contrasted to the identification task, is that only one comparison is carried out between the input sample and the prototype for the claimed identity. Here again, time alignment is critical to ensure that comparison distances are consistently small when the input and prototype speakers are the same.

Connected Word Recognition—Level Building

In the case where the unknown utterance corresponds to a text pattern with more than a single spoken word, the DP iteration used in isolated word recognition must be extended so that multiple patterns can be combined to give the best match to the string. One way in which this can be accomplished is to add level informa-

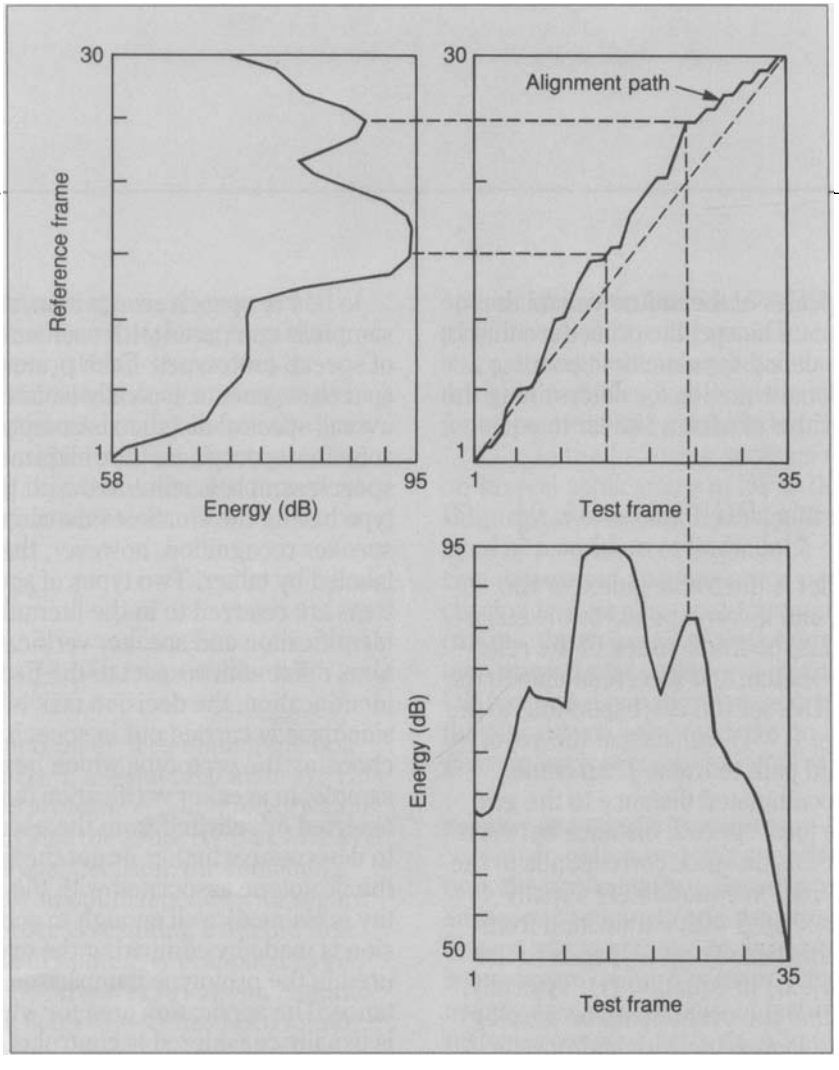

Figure 1. An example of using the DTW procedure for time alignment.

tion to the frame-based DP recursion of equation (3). Now the DP recursion consists of two interacting layers of multistage processes, one corresponding to the frame index i and the other corresponding to the level index l which defines the word position in a string. Such a DP-based algorithm for connected word recognition is formulated as follows. We define $D_l(i, j)$ as the accumulated distance to frame i of the test pattern and frame j of the reference pattern, at level l within the string. The level l can vary from 1 to L, where L is the maximum expected string length. The DP iteration (3) is then modified so that for $j > 1$, and for all reference patterns

$$D_l(i, j) = d(i, j) + \min_{\{k\}} \left[D_l(i-1, k) + a(k, j) \right] \quad (4)$$

and for $j = 1$, i.e., the first frame of any reference pattern at any level l, we have

$$D_l(i, 1) = d(i, 1) + \min \left\{ \min \left[D_{l-1}(i-1, m) \right], D_l(i-1, 1) + a(1, 1) \right\} \quad (5)$$

Equation (5) defines the intraword DP recursion, which is similar to that in equation (3), and equation (5) defines the interword DP recursion for going from a word at level $(l-1)$ to a word in the next level l. In the formulation of equation (5), the second minimization inside the bracket is done over all possible words that can appear in the previous level and terminate at some frame m, while

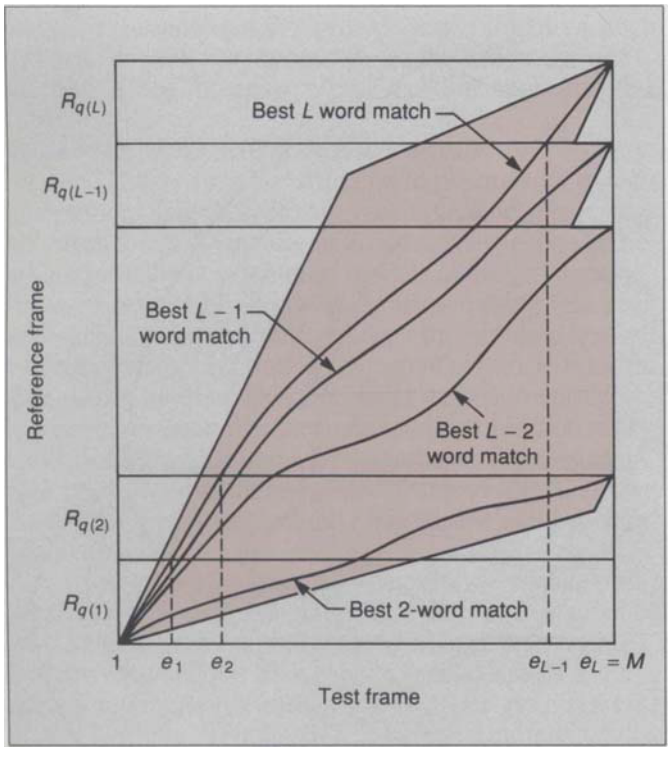

Figure 2. An example of using level building for connected word recognition.

the first minimization is performed to extend a path either by picking up from a path that terminates at some frame m of a word in the previous level (interlevel) or by picking up from a path that terminates at the first frame of the current level (intralevel). The initial conditions are defined so that $D_0(0, 1) \equiv 0$ for all the first frames of all the words that can appear in the first position of a string, and in all other cases, $D_0(0, k)$ is set to be infinity. By iterating the above pair of DP functional equations in an appropriate manner, a number of interesting and versatile algorithms for speech recognition based on concatenating words (or units smaller than words) have evolved.

This DP procedure for recognizing a connected sequence of words, known as *level building*, has been applied successfully for connected digit recognition, and for syntax-directed connected word recognition.[8,9]

In Figure 2, we show an example of using level building to obtain the optimal warping paths for strings of up to L words. In the horizontal axis we plot the frame index of the test utterance, and in the vertical axis we list all possible concatenations of reference patterns to form a string of up to L words. For example, in Figure 2, $q(l)$ denotes the index of the word in the lth position of a string, $R_{q(l)}$ is the reference pattern for that word, and the concatenation of reference patterns, $\{R_{q(l)}, 1 \le l \le K\}$, forms a string of K words. The horizontal lines represent the boundaries between two reference patterns in consecutive levels. Therefore the intersections of the warping paths and the horizontal lines indicate the word segmentation boundaries in the test utterance for the corresponding warping path. For example, the set $\{e_l, 1 \le l \le L\}$, shown in Figure 2, corresponds to word boundaries for the best L word match. The shaded region enclosed by the dashed lines represents the space of all admissible warping paths. It is interesting to note, for this particular example, that the warping path corresponding to a single word match does not exist, because the test utterance is too long; hence a single word match is unreasonable.

The level building algorithm is basically a level- (or word-) synchronous DP searching procedure, such that the frame recursion is embedded in each of the level recursions. The algorithm requires that all the distance pairs $d(i, j)$ for DP iterations be accessible for computation at any level, which poses a memory management problem for large recognition tasks and for real-time implementation of the algorithm. Since most recognition tasks can be accomplished by mapping the level (or grammar) information onto a finite state decision network,[10] connected word recognition problems can be formulated as optimal path searching problems through a task-oriented network (e.g., a subnetwork of the Bell Laboratories airline reservation system[11] of the type

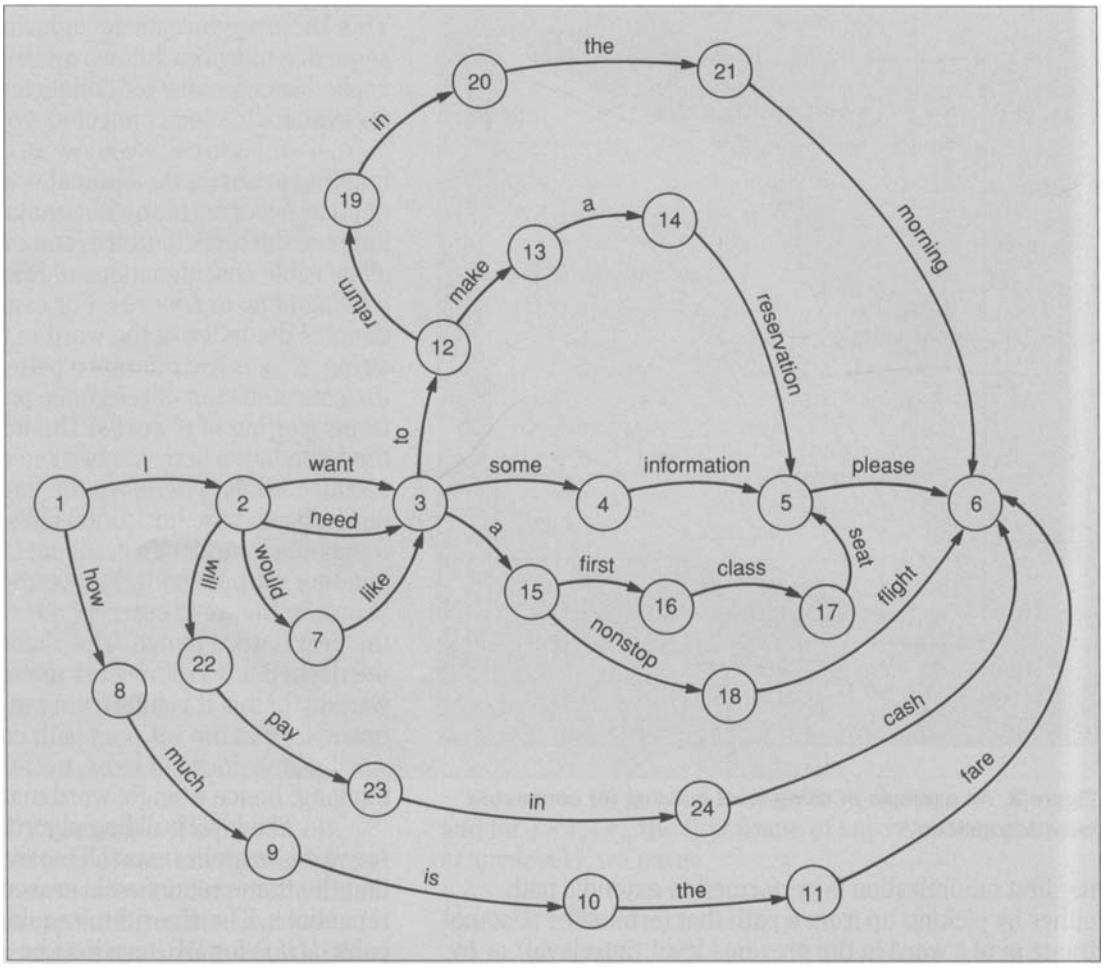

Figure 3. A subnetwork for the AT&T Bell Laboratories airline reservation system task.

shown in Figure 3). Based on the concept of maximum likelihood network decoding, the frame synchronous network search algorithm[12] uses the same pair of DP functional equations as in level building. However, rather than search on a level-by-level basis, the approach keeps track of the optimal path to any node (including the intraword nodes and grammar nodes) of the finite state network, at every time instance, so that the best path is obtained on a frame-by-frame basis. Since only the best path information at the previous frame, plus the distance vector at the current frame, is required for performing the DP recursions, the algorithm is more efficient than level building in memory management. The algorithm is also highly modular, which makes it easy to implement

on special-purpose parallel processor architectures, such as the AT&T BT-100 processor[13] or the graph search machine (GSM) processor,[14] for real-time applications.

Speech Synthesis Using Articulatory Models

We now turn our attention to the areas of speech analysis and speech synthesis research. One promising approach to improving the naturalness of speech synthesis is to employ physiological models of the glottis and the vocal tract.[15] While speech synthesis using this approach is quite straightforward,[16] the problem lies in estimating an appropriate sequence of parameters for the articulatory models. This parameter estimation problem has been formulated as a constrained optimization problem[17] in which a sequence of tract shapes is found that matches the sequence of spectral feature vectors, so that the synthesized speech closely mimics the input speech segment.

Instead of exhaustively searching over all possible tract shape sequences, dynamic programming can be used to obtain the optimal shape sequence on a frame-by-frame basis. For a tract shape space of size M and a speech segment of N frames, the decision space is a grid of $M \times N$ points. If we let $D(i, j)$ be the accumulated distance associated with the optimal path through the grid to the point (i, j), then the resulting DP functional equation for solving the optimization problem is of the same form as equation (3), in which $d(i, j)$ is the acoustic distance between the current speech frame i and tract shape j and $a(k, j)$ is the cost of making a transition from tract shape k to shape j as defined in Reference 17.

In the following setup, the articulatory tract shape space consists of $M = 2523$ vocal tract shapes. Each frame of speech is modeled by a vector of linear prediction coefficients (LPCs)[18] which characterize the smoothed spectrum of the speech frame. The LPC vector is then linked to a tract shape (also referred to as a code vector) in the articulatory space (also referred to as a codebook). The sentence "Why were you away a year, Roy?" as spoken by a male speaker was used to create the tract shape sequences for speech synthesis. Three estimation approaches were compared. The first scheme estimated the sequence of tract shapes directly from the articulatory codebook *without* imposing any shape transition cost. The quality of the synthesized speech using this discontinuous tract shape sequence is rather poor. The second scheme used a local dynamic codebook access procedure[17] that contains a shape transition cost component in the objective function and for which the shape selection was made locally at every frame. Much smoother shape transitions were observed. The third approach used a delayed-decision dynamic codebook access strategy[19] that is similar to the second approach except that the decision for the optimal shape sequence was made only at the end of utterance. The DP-based approach, in addition to producing the desired smoothed shape transitions, yielded the best quality of synthesized speech among the three approaches evaluated.

Speech Analysis—Estimation of Formant Trajectories

A *formant* is generally defined as a resonance (manifested as a complex pole pair) of the vocal tract transfer function. The formant frequencies as functions of time, or *formant trajectories*, are often important for speech research. Speech synthesis studies have demonstrated that the first three formants are of primary importance in the representation of voiced speech. However, the spectral information in a short segment of speech is often insufficient for determining the formant trajectories. In the following, we describe a DP-based algorithm[20] to solve for a set of formant trajectories that optimally satisfy a set of formant continuity constraints. The results of applying the algorithm to estimating the first four formant trajectories are consistent with estimates provided by human experts.

Given a speech segment of T spectral frames, the DP-based formant-tracking algorithm attempts to find the best set of formant trajectories such that the

total cost of selecting the trajectories is minimized. For every speech frame i, we have available a set of I_i candidate frequency-bandwidth pairs obtained from a short-time spectral analysis (e.g., LPC analysis). Let $d(i, j)$ be the cost of selecting the candidate $j \in I_i$ at frame i and $a_i(k, j)$ be the transition cost at frame i between candidates $k \in I_{i-1}$ and $j \in I_i$. Then the minimum accumulated cost $D(i, j)$, for selecting candidate j at frame i, can be obtained by the DP recursion:

$$D(i, j) = d(i, j) + \min_{k \in I_{i-1}} \left[D(i-1, k) + a_i(k, j) \right] \quad (6)$$

After all T frames are processed, a set of optimal formant trajectories is obtained that can then be retrieved using a traceback procedure. The local cost $d(i, j)$ is evaluated on the basis of a weighted sum of the assigned bandwidth and the deviation of the assigned frequencies from the nominal formant frequencies. The transition cost $a_i(k, j)$ is a function of how stationary the signal is and how much the frequency changes in interframe formant transitions. A detailed description of the algorithm and the method for estimation of the parameters needed in evaluating the costs $d(i, j)$ and $a_i(k, j)$ can be found in Reference 20.

Speech Modeling Using Hidden Markov Models

The speech signal can be considered a sequential process in which acoustic manifestations of linguistic codes are observed. For convenience, acoustic signals for a particular linguistic code are usually modeled by a parameterized stochastic process. Suppose there are N distinct sound classes, corresponding to N distinct linguistic codes, in an utterance represented by $\{\mathbf{y}_t\}_{t=1}^{T}$. The feature vector $\mathbf{y}_t$ usually consists of spectral information observed at frame t. Given a sequence of feature vectors, the problem of speech modeling is to find the best characterization of these N distinct signal classes and to determine how they interact with each other and evolve to give the observed speech utterance. The interaction and evolution of signal classes define the global "structure" of the utterance while the characterization of each signal class models the local acoustic events in the utterance. The inherent uncertainty in the acoustic signal necessitates the use of stochastic modeling techniques.

In the statistical hidden Markov model (HMM) framework, an N-state Markov chain with transition matrix $A = [a_{ij}]$, is used to characterize the underlying transitional nature of the N linguistic codes (states) in an utterance. The linguistic information, which is represented by a state sequence $\{s_t\}$, is not directly observed (it is hidden). Instead, we observe a sequence of feature vectors $\{\mathbf{y}_t\}$ from which inference about the underlying Markov structure is made. Parameterized stochastic models $\{P(\mathbf{y}_t \mid s_t = i, i = 1, \ldots, N)\}$ are usually used to describe the local acoustic event (within a state). Let B denote the set of parameters that characterizes the state output probability $P(\mathbf{y}_t \mid s_t = i)$. If the state output probability is Gaussian, for example, then the set B consists of the mean vectors and the covariance matrices associated with each state. Therefore, an HMM is defined by the parameter set $\boldsymbol{\lambda} = (A, B)$. A three-state HMM, with its associated parameter set $\boldsymbol{\lambda}$, is shown in Figure 4. On the basis of the Markov structure, the probability of observing the utterance $Y = \{\mathbf{y}_t\}$ along with a particular state sequence $S = \{s_t\}$ can be expressed as

$$P(Y, S) = \prod_{t=1}^{T} a_{s_{t-1}s_t} P(\mathbf{y}_t \mid s_t) \quad (7)$$

where a_{ij} is the transition probability from state i to state j. The probability of the observations Y is obtained by summing the right-hand side of equation (7) over all possible state sequences, giving

$$P(Y) = \sum_{\{S\}} \prod_{t=1}^{T} a_{s_{t-1}s_t} P(\mathbf{y}_t \mid s_t) \quad (8)$$

The modeling problem then becomes that of

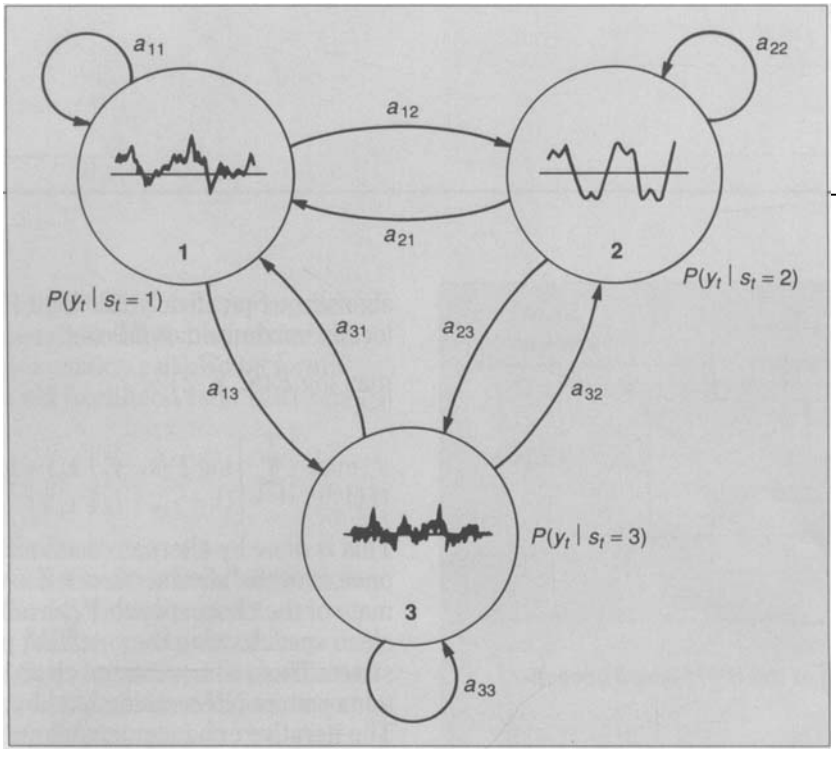

Figure 4. A three-state hidden Markov model.

finding the state transition probabilities and the parameters defining the N sound classes so that either equation (7) or equation (8) is maximized. The maximization of equation (8) is solved by a procedure known as the Baum-Welch reestimation algorithm.[21] We will only discuss the maximization of equation (7), which is of direct relevance to the DP technique. The criterion of maximizing the joint state-observation probability (7) is formally stated as

$$\max_{\lambda} \left\{ \max_{S} \log P(Y, S \mid \lambda) \right\} =$$

$$\max_{\lambda} \left\{ \max_{\{s_t\}} \left[\sum_{t=1}^{T} \log a_{s_{t-1}s_t} + \sum_{t=1}^{T} \log P(\mathbf{y}_t \mid s_t, \lambda) \right] \right\} \quad (9)$$

Instead of enumerating all N^T possible state sequences, the maximization over the state sequences is achieved by a DP-based Viterbi algorithm.[22,23] The algorithm is similar to the DTW time-alignment procedure described under "Speaker and Speech Recognition—Dynamic Time Warping." Conceptually, a set of functional equations similar to equation (3) can be systematically constructed to solve for the optimal path (state sequence) through a grid of $N \times T$ points. After the optimal state sequence is decoded, the maximization of equation (9) over the parameter space can easily be accomplished by traditional maximum likelihood estimation techniques. The entire iterative procedure to solve equation (9) has been called the segmental k-means algorithm.[24] The algorithm has been extensively and successfully applied to obtain statistical reference models that are crucial for high-performance speech recognition tasks.[25]

Model-Based Speech Enhancement

Statistical approaches for enhancing speech signals degraded by noise require knowledge of the joint probability density function (PDF) of the clean speech and the noise process. For the case of speech signals which have been degraded by statistically independent additive noise, considered in Reference 26, the PDFs of the speech signal and the noise process must be known. Since the PDF of the speech signal is not known and the PDF of the noise process is rarely available, parametric models for the unknown PDFs have been used. The

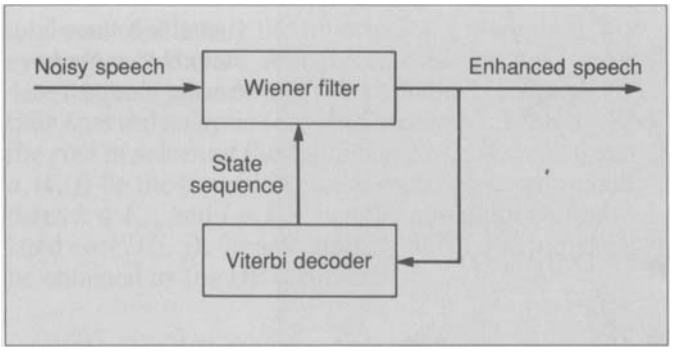

Figure 5. A block diagram of the HMM-based speech-enhancement algorithm.

parameter sets of these models are estimated by using long sequences of data (known as training data) from the two processes. Useful models for the speech signal are Markov sources or HMMs. In particular, HMMs with mixtures of Gaussian autoregressive (AR) output PDFs have proven useful in speech recognition and enhancement applications. Models for the noise process depend on its specific type. For the case of Gaussian noise with a theoretically flat power spectral density, a Gaussian AR model of low order is typically used.

The estimation of the parameter set $\boldsymbol{\lambda}_s$ of the HMM for the speech signal is done by the segmental k-means approach discussed under "Speech Modeling Using Hidden Markov Models." That approach locally maximizes the joint PDF of state and observation sequences using dynamic programming and maximum likelihood formulas. The parameter set $\boldsymbol{\lambda}_n$ of the AR model for the noise process is estimated from AR modeling of the average of the autocorrelation vectors in the training sequence from the noise process.

Given the models for the clean speech and the noise process, $\{P(S, Y \mid \boldsymbol{\lambda}_s)\}$ and $\{P(N \mid \boldsymbol{\lambda}_n)\}$, respectively, the clean speech signal $Y = \{y_t\}$ is estimated from the given noisy speech $Z = \{z_t = y_t + n_t\}$ by a procedure that is consistent with the segmental k-means training algorithm. Specifically, the joint PDF $P(S, Y, Z)$ is locally maximized as follows:

$$\max_{\{s_t, y_t\}} \log P(S, Y, Z) =$$

$$\max_{\{s_t, y_t\}} \left\{ \sum_{t=1}^{T} \left[\log P(s_t, y_t \mid \boldsymbol{\lambda}_s) + \log P(z_t - y_t \mid \boldsymbol{\lambda}_n) \right] \right\} \quad (10)$$

This is done by alternate maximization of $\log P(S, Y, Z)$, once over the state sequence S assuming that an estimate of the clean speech Y is available, and then over the clean speech using the resulting most likely sequence of states. Thus, a sequence of clean speech sample functions with nondecreasing likelihood values is generated. The iterative enhancement algorithm is terminated when a convergence criterion is satisfied, e.g., if the value of the likelihood function (10) in two consecutive iterations is less than or equal to a given threshold. The most likely sequence of states, say $\hat{S}$, is estimated by applying a DP-based Viterbi algorithm to the current estimate of the clean speech, say $\hat{Y}$. The estimation of the clean speech vectors is done by time-varying Wiener filtering of the noisy speech. The Wiener filters are constructed from the AR covariance matrices of the HMM associated with the most likely state sequence and the AR covariance matrix of the noise model. A block diagram of the algorithm is given in Figure 5. This algorithm provided a signal-to-noise ratio (SNR) improvement of approximately 4 to 6 dB, when enhancing speech degraded by additive white Gaussian noise at 10-dB input SNR. And the perceived quality of the enhanced speech was significantly better than that of the original noisy speech.

Autoregressive Maximum Likelihood Speech Segmentation

For a given utterance of T frames, the objective of speech segmentation is to partition the utterance into M consecutive segments with boundaries $\{t_1, \ldots, t_M\}$, such that the speech frames in each segment have similar acoustic or phonetic properties. Based on the maximum likelihood autoregressive formulation,[27] the follow-

ing speech segmentation algorithm solves for segment boundaries so that the overall likelihood is maximized. Equivalently, speech segmentation can also be formulated to minimize the overall likelihood ratio (LR) segment distortion, i.e.,

$$\min_{\{t_1, \cdots, t_M\}} \sum_{m=1}^{M} \left[d_{LR}(t_m, t_{m+1} - 1) \right] \qquad (11)$$

where $d_{LR}(t_m, t_{m+1} - 1)$ is defined as the LR distortion of the mth segment.[27] At every time frame, let $\{D(m, t_{m+1} - 1),\ 1 \le m \le M\}$ denote the set of the minimum accumulated LR segment distortions obtained so far. Then at the next time frame t, the DP recursion to solve for $D(m, t_{m+1} - 1)$ is simply

$$D(m, t_{m+1} - 1) =$$

$$\min_{\{t_{m+1} < t\}} \left[D(m - 1, t_m - 1) + d_{LR}(t_m, t_{m+1} - 1) \right] \qquad (12)$$

After the last frame T ($T \ge M$) is processed, we obtain the overall minimum LR distortion $D(M, T)$, as well as the set of optimal segmentation boundaries $\{t_1, \ldots, t_M\}$, which can be efficiently retrieved through a backtracking procedure.[28] The algorithm has served as an initialization procedure for an acoustic segment model-based approach to speech recognition.[29]

Speech Transmission—Combined Source-Channel Coding

A major problem encountered in digital speech transmission over a mobile radio channel is the exceptionally high error rate caused by signal fading. For most speech coders, the bits in a quantized speech segment exhibit widely varying sensitivity to channel errors. For example, the dynamic bit allocation sub-band speech coder in Reference 30 exhibits more than 40-dB variation in error sensitivity, necessitating the use of an efficient unequal error protection code. It has been shown that convolutional codes are suitable for combined source and channel coding, so that channel redundancy is used to match the error protection capability with the error sensitivity of the speech bits.

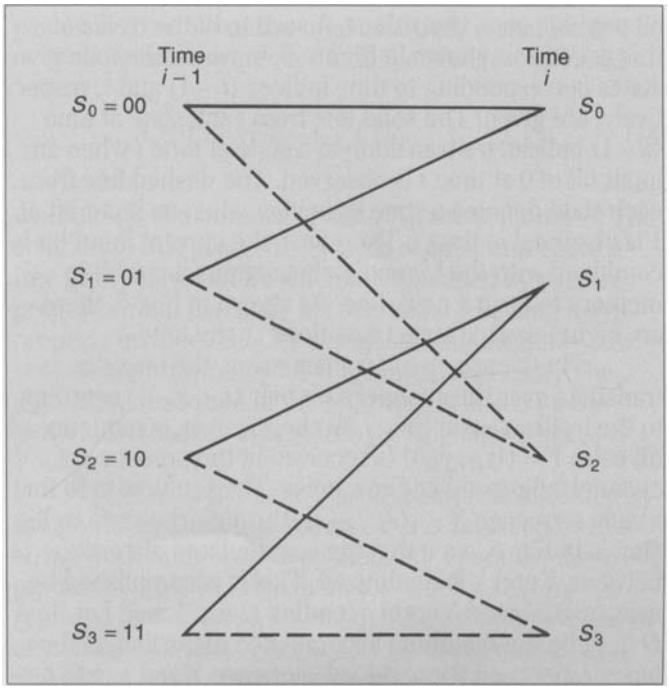

Figure 6. A section of the state trellis corresponding to a binary convolution coder.

Consider a binary convolutional encoder with two delay elements (memory $M = 2$). At time index i, the outputs of the encoder are determined by the current input and the two previous inputs. These two previous bits represent the state of the encoder at that time. Since the input takes on binary values, there are four states in this encoder. Therefore, every input bit sequence is associated with a state sequence; i.e., the input bit sequence can be uniquely determined from a state sequence. Starting from the initial state, all possible state sequences form a *trellis*, which is basically a network consisting of

all possible state transitions. A section of the trellis of this encoder is shown in Figure 6, in which the four states corresponding to time indices $(i-1)$ and i, respectively, are given. The solid line from each state at time $(i-1)$ indicates a transition to a state at time i when an input bit of 0 at time i is observed. The dashed line from each state denotes a state transition when an input bit of 1 is observed at time i. Therefore, the current input bit is combined with the immediately previous input bit in memory to form a new state. As shown in Fig. 6, there are eight possible state transitions at any time.

In the process of transmission, the encoder transmits over the channel a bit pair (x_{i1}, x_{i2}), according to the trellis state at time i. At the receiver, a sequence of bit pairs $Y = \{(y_{i1}, y_{i2})\}$ is received in the presence of channel fading and additive noise. The problem is to find a code sequence $X = \{(\hat{x}_{i1}, \hat{x}_{i2})\}$, through the state trellis shown in Fig. 6, such that the accumulated distortion between X and Y is minimized. This is accomplished by using a DP-based Viterbi decoding algorithm.[22] Let $D(i, j)$ be the minimum accumulated distortion, at time index i, between the received sequence Y and a path terminating in trellis state j. Then $D(i, j)$ can be solved sequentially by using the same DP recursion in equation (6), where k is one of the two trellis states at time index $(i-1)$ that are connected to state j at time index i. The state cost $d(i, j)$ is equal to zero in this case. The incremental cost $a_i(k, j)$ is a function of the states k and j and the received symbols at time i. Finally, the trellis will terminate in a known state, and a backtracking procedure through the record gives the estimated state sequence, which in turn uniquely determines the transmitted bit sequence.

A Stochastic Parts Algorithm

We now turn our attention to language modeling. There are two important aspects in analyzing the structure of a sentence: (1) categorization of words into *parts of speech*, and (2) grouping of words into *structural constituents* of the sentence. The second aspect is related to syntax and parsing, which we will address under "Natural Language Processing—Syntax and Parsing." We now discuss the issue of assigning parts of speech to words in a sentence.

It is well-known that the part of speech (e.g., noun, verb, adjective) depends on context in written text. The word "table," for example, can be a verb in some contexts (e.g., "The chairman will table the motion") and a noun in others (e.g., "The table is ready"). Part-of-speech tagging is an important practical problem with potential applications in many areas including speech synthesis, speech recognition, proofreading, query answering, machine translation, and text retrieval from large databases (e.g., newspapers). In text-to-speech synthesis, for example, it is necessary to know the part of speech of certain words in order to pronounce them correctly (e.g., "to conSTRUCT" vs. "the CONstruct"). Other important examples can be found in Reference 31.

The stochastic tagging algorithm discussed in Reference 31 assigns the most likely part of speech sequence $P = \{p_i\}$ to each word in an input sentence $W = \{w_i\}$, so that the conditional probability $\Pr(P|W)$ is maximized. The conditional probability can be approximated by the product of (1) lexical probabilities $\Pr(p_i \mid w_i)$ and (2) contextual probabilities $\Pr(p_i \mid p_{i+1}, \cdots, p_{i+n})$. Both probabilities can be estimated from training data such as the Tagged Brown Corpus,[32] which is a corpus of approximately 1 million words with part-of-speech tags assigned laboriously by hand over many years. In the following, we let $n = 2$, so that the contextual probability is approximated by the so-called *trigram* probability. The optimization problem can then be formulated as

$$\max_{\{p_i\}} \sum_{i=1}^{N} \left\{ \log \Pr(p_i \mid w_i) + \log \Pr(p_i \mid p_{i+1}, p_{i+2}) \right\} \quad (13)$$

The solution is then obtained by dynamic programming, in which the search is performed starting from the end of the sentence and moving backward to the beginning of

the sentence.

Conceptually, the DP search enumerates all possible assignments of parts of speech to input words. If each word were at most k (≈ 10) ways ambiguous, then for an N-word sentence ($N \approx 100$), the search would need to consider k^N ($\approx 10^{100}$) possible part-of-speech sequences. In fact, it is not necessary to enumerate all of them because the scoring function cannot see more than n (≈ 2) words away. Thus, only k^n ($\approx 10^2$) paths need to be preserved, and the search space is considerably reduced.

Natural Language Processing—Syntax and Parsing

As speakers, readers, and writers of English, all of us are aware—although possibly not on a conscious level—that an arbitrary sequence of perfectly valid words will not necessarily form a meaningful sentence. This phenomenon is no accident. Constraints on word order in natural language serve two crucial purposes. First, they aid in determining the meaning of messages, and, second, they decrease the likelihood that one word will be mistaken for another, thereby changing the intended meaning of the message. We would like to understand and exploit both of these functions for automatic speech recognition. Specifically, we focus our attention on syntax and parsing. For a review of other important issues in applying structural methods to speech recognition, Reference 33 is suggested. Reference 34 discusses the theory of formal languages and grammars and related topics on syntax and parsing.

The classification accuracy of a speech recognizer can be improved by exploiting constraints on word order imposed by a task language and expressed by a formal grammar G. The method for doing so is called *probabilistic parsing*. In the probabilistic parsing formulation, we are given a word lattice $W = (w_i, t_{i-1}, t_i)$ of all possible ways of classifying a particular speech segment in a time interval (t_{i-1}, t_i), as a particular word w_i, and some measure of the cost, $\mathbf{C}[w_i \mid t_{i-1}, t_i]$, for making the assignment. The purpose is to find the most likely sentence $\hat{S}$ such that the total cost $\mathbf{C}(\hat{S})$ is minimum over all possible sentences in the language $L(G)$, i.e.,

$$\mathbf{C}(\hat{S}) = \min_{S \in L(G)} \left\{ \sum_{i=1}^{|S|} \mathbf{C}[w_i \mid t_{i-1}, t_i] \right\} \tag{14}$$

where $|S|$ is the number of words in the sentence S. The solution to this problem, for both regular and context-free grammars, can be efficiently obtained by dynamic programming. Since the set of context-free languages properly includes the set of regular languages, we present an algorithm only for the context-free case. The algorithm described in the following is based on the general DP-based context-free parsing algorithm of Younger.[35]

Let $\alpha \rightarrow \beta$ denote a production rule specified by the grammar such that the string α is replaced by the string β when the rule is applied. For example, there is a production rule in English under which a sentence is replaced by a *noun phrase* followed by a *verb phrase*. For any string (phrase) A, let $\psi_{ij}(A)$ be the minimum-cost string that spans the ith to the jth word position in S, and let $\phi_{ij}(A)$ be its cost. Initially, we try to find all one-word solutions in word position i such that

$$\phi_{ii}(A) = \min_{\{A \rightarrow a\}} \{\mathbf{C}[a \mid t_{i-1}, t_i]\} \tag{15}$$

where the minimum is taken over all possible one-word production rules $\{A \rightarrow a\}$, and the one-word string is obtained as the word that satisfies equation (15), i.e., for $1 \le i \le |S|$,

$$\psi_{ii}(A) = \bar{a} = \underset{\{A \rightarrow a\}}{\operatorname{argmin}} \{\phi_{ii}(A)\} \tag{16}$$

We now try to combine words into phrases (strings) of size greater than 1 by finding production rules that generate these phrases; i.e., for $1 \le i, j \le |S|$, and $i \le l < j$, we solve for strings of size greater than 1 by iterating the following two equations,

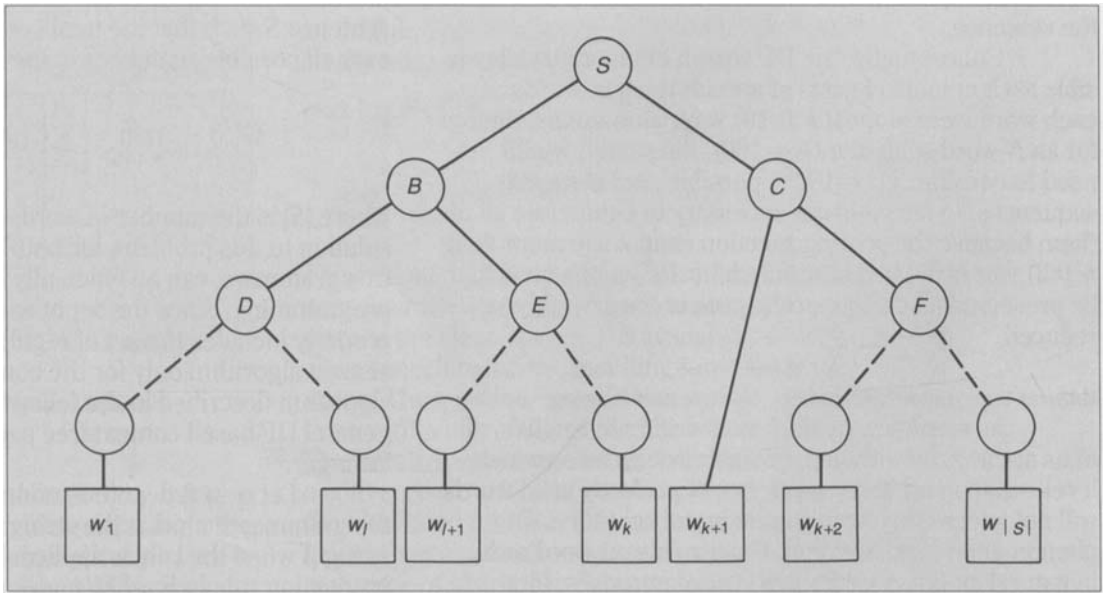

Figure 7. An example of a parse tree for a sentence.

$$\phi_{ij}(A) = \min_{\{A \to BC\}} \{ \min_{i \le l < j} [\phi_{il}(B) + \phi_{l+1j}(C)] \} \qquad (17)$$

and

$$\psi_{ij}(A) = (\bar{B}, \bar{C}, \bar{l}) = \underset{\substack{\{A \to BC\} \\ i \le l < j}}{\operatorname{argmin}} \{\phi_{ij}(A)\} \qquad (18)$$

At first, equations (17) and (18) are solved for $j = (i + 1)$ to find all production rules that will combine any two phrases B and C of sizes less than or equal to 2 into a two-word phrase A. Then, we set $j = (i + 2)$ and find all production rules that will combine any two phrases into a three-word phrase. We keep iterating equations (17) and (18) until we find all production rules that will combine two phrases into a sentence S of $|S|$ words. Finally, we have available a sentence S and a sequence of minimum-cost production rules that produces the sentence S. The minimum-cost sentence $\hat{S}$ can then be retrieved as $\hat{S} = \psi_{1|S|}(S)$, and the minimum cost is simply $\mathbf{C}[\hat{S}] = \phi_{1|S|}(S)$.

The process of recovering the sequence of production rules from the sentence S is one of constructing a *parse tree*, illustrated in Figure 7. The root of the tree is the sentence S, so we begin by examining $\psi_{1\,|S|}(S) = (B, C, k)$. This means that there are two subtrees (phrases) whose roots are B and C and which account for w_1 through w_k and w_{k+1} through $w_{|S|}$, respectively. Now we examine $\psi_{1k}(B)$ and $\psi_{k+1|S|}(C)$, each of which indicates the formation of two subtrees. The two branching subtrees from B, namely D and E, span words from positions 1 to l and positions $l + 1$ to k respectively, while those branching from C account for the single terminal symbol (word) w_{k+1} and a subtree F for the rest of the sentence. This reconstruction process goes on until all the phrases are replaced by words, and at that point we have obtained a recognized sentence $\hat{S}$ and a parse tree associated with the sentence.

The version of this algorithm appropriate to regular languages reduces to a formulation very similar to equation (9). This method has been used in the

conversational-mode speech recognition system described in Reference 11. The algorithm exactly as described in equations (15) to (18) for context-free languages has been implemented and used in a large-vocabulary speech recognition task.[36]

Summary

In this paper, we have briefly discussed a number of applications of dynamic programming in the areas of speech and language processing. As should be clear from this brief summary, optimization by means of dynamic programming pervades the field of speech and language processing. In fact, it is arguable that no other single mathematical technique is so widely employed in this field. Most of the applications presented herein have been pioneered by the people noted under "Acknowledgments," who have used them in many successful laboratory experiments and in some commercial products. Other applications are still in the research stage, but we reasonably expect that they will be perfected and will become part of the evolving technology.

Acknowledgments

The author is indebted to the following people whose contributions to this paper made its completion possible: K. W. Church, Y. Ephraim, B.-H. Juang, S. E. Levinson, L. R. Rabiner, A. E. Rosenberg, J. Schroeter, N. Seshadri, F. K. Soong, and D. Talkin. The author also thanks these colleagues for allowing him to use part of their published papers, and hence their contributions to speech and language processing research are reflected in the list of references for each application. Special thanks are due to L. R. Rabiner for reading and offering comments on earlier versions of the paper.

References

1. R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.
2. G. S. Slutzker, "Non Linear Method of Analysis of Speech Signal," *Fifth Conference for Young Specialists of Scientific Research*, Institute of Radio Communications, U.S.S.R., October 1967.
3. Special issue on Speech Processing Technology, *AT&T Technical Journal*, Vol. 65, No. 5, September/October 1986.
4. Special issue on Artificial Intelligence, *AT&T Technical Journal*, Vol. 67, No. 1, January/February 1988.
5. L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson, "Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-26, No. 6, December 1978, pp. 575-582.
6. B. S. Atal, "Automatic Recognition of Speakers From Their Voices," *Proceedings of the IEEE*, Vol. 64, No. 4, April 1976, pp. 460-475.
7. A. E. Rosenberg, "Automatic Speaker Verification: A Review," *Proceedings of the IEEE*, Vol. 64, No. 4, April 1976, pp. 475-487.
8. C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 2, April 1981, pp. 284-297.
9. C. S. Myers and S. E. Levinson, "Speaker-Independent Connected Word Recognition Using a Syntax-Directed Dynamic Programming Procedure," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-30, No. 4, August 1982, pp. 561-565.
10. J. K. Baker, "Stochastic Modeling for Automatic Speech Understanding," *Speech Recognition*, D. R. Reddy (ed.), Academic Press, New York, 1975, pp. 521-542.
11. S. E. Levinson and K. L. Shipley, "A Conversational Mode Airline Information and Reservation System Using Speech Input and Output," *Bell System Technical Journal*, Vol. 59, No. 1, January 1980, pp. 119-137.
12. C.-H. Lee and L. R. Rabiner, "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, 1988, pp. 410-413.
13. A. L. Gorin, D. B. Roe, and L. R. Rabiner, "High-Performance Real-Time Speech Recognition," *Defense Computing*, November 1988.
14. S. Glinski et al., "The Graph Search Machine (GSM): A Programmable Processor for Connected Word Speech Recognition and Other Applications," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Dallas, April 1987, pp. 519-522.
15. K. Ishizaka and J. L. Flanagan, "Synthesis of Voiced Sounds from a Two-Mass Model of Vocal Cords," *Bell System Technical Journal*, Vol. 51, No. 6, 1972, pp. 1233-1268.
16. M. M. Sondhi and J. Schroeter, "A Hybrid Time-Frequency Domain Articulatory Speech Synthesizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 7, July 1987, pp. 955-967.
17. J. Schroeter, J. N. Larar, and M. M. Sondhi, "Speech Parameter Estimation Using a Vocal Tract/Cord Model," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Dallas, April 1987, pp. 308-311.

18. J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.
19. J. Schroeter and M. M. Sondhi, "Dynamic Programming Search of Articulatory Codebooks," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, May 1989, pp. 588-591.
20. D. Talkin, "Speech Formant Trajectory Estimation Using Dynamic Programming with Modified Transition Costs," *Journal of the Acoustical Society of America*, Sup. 1, Vol. 82, Fall 1987, p. S55.
21. L. E. Baum and J. A. Eagon, "An Inequality with Application to Statistical Estimation Probabilistic Functions of Markov Processes and to a Model for Ecology," *Bulletin of the American Mathematical Society*, Vol. 73, 1967, pp. 360-363.
22. G. D. Forney, Jr., "The Viterbi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, March 1973, pp. 268-278.
23. L. R. Rabiner and B.-H. Juang, "An Introduction to Hidden Markov Models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, Vol.3, No.1, January 1986, pp. 4-16.
24. L. R. Rabiner, J. G. Wilpon, and B.-H. Juang, "A Segmental *k*-means Training Procedure for Connected Word Recognition," *AT&T Technical Journal*, Vol. 65, No. 3, May/June 1986, pp. 21-31.
25. L. R. Rabiner, J. G. Wilpon, and F. K. Soong, "High Performance Connected Digit Recognition Using Hidden Markov Models," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988, pp. 119-122.
26. Y. Ephraim, D. Malah, and B.-H. Juang, "On the Application of Hidden Markov Models for Enhancing Speech Signals," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988, pp. 533-536.
27. F. Itakura and S. Saito, "A Statistical Method for Estimation of Speech Spectral Density and Formant Frequencies," *Electronics and Communication*, Vol. 53-A, 1970, pp. 36-43.
28. T. Svendsen and F. K. Soong, "On the Automatic Segmentation of Speech Signals," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Dallas, April 1987, pp. 77-80.
29. C.-H. Lee, F. K. Soong and B.-H. Juang, "A Segment Model Based Approach to Speech Recognition," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988, pp. 501-504.
30. R. V. Cox, J. Hagenauer, N. Seshadri, and C.-E. Sundberg, "A Sub-Band Coder Designed for Combined Source and Channel Coding," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988, pp. 235-238.
31. K. W. Church, "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Second Conference on Applied Natural Language Processing*, Austin, Texas, 1988.
32. W. Francis and H. Kucera, *Frequency Analysis of English Usage*, Houghton Mifflin, Boston, 1982.
33. S. E. Levinson, "Structural Methods in Automatic Speech Recognition," *Proceedings of the IEEE*, Vol. 73, No. 11, November 1985, pp. 1625-1650.
34. J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, Massachusetts, 1969.
35. D. M. Younger, "Recognition and Parsing of Context Free Languages in Time n^3," *Information and Control*, Vol. 10, No. 2, 1967, pp. 189-208.
36. S. E. Levinson, A. Ljolje, and L. G. Miller, "Large Vocabulary Speech Recognition Using a Hidden Markov Model for Acoustic/Phonetic Classification," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988, pp. 505-508.

(Manuscript received December 29, 1988)