Chapter 5
The Viterbi Search

5.1 Introduction

We have now discussed, at least on a basic level, acoustic processing, acoustic modeling, and language modeling. The remaining component of a speech recognizer is the hypothesis search (see section 1.2). Several methods are possible; the ones we discuss in this chapter are based on the Viterbi algorithm [1] introduced in sections 2.4 and 2.5.[1] Tree search methods will be presented in the next chapter.

5.2 Finding the Most Likely Word Sequence

We will now describe a method of search for the most likely word sequence $\mathbf{W}$ to have caused the observed acoustic data $\mathbf{A}$. This method, like all the known others for large vocabulary sizes $|\mathcal{V}|$, cannot actually be guaranteed to find the most likely $\mathbf{W}$. It gives very good results, however, from the practical point of view (i.e., it is very rare that the maximizing sequence

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{A}|\mathbf{W})P(\mathbf{W}) \tag{1}$$

is the one actually spoken but is not found by our search method).[2]

1. This will demonstrate again the usefulness of the HMM formulation. We will take further advantage of the fact that embedding HMMs into an HMM leads to a new HMM. In a sense, the subject of this chapter is already known to the reader even before we delve into it.

2. As we know from chapter 2, the Viterbi algorithm (which is the basis of our method) finds the most likely path through an HMM. But (1) demands that we find for each candidate word string $\mathbf{W}$ the probability of the set of paths that

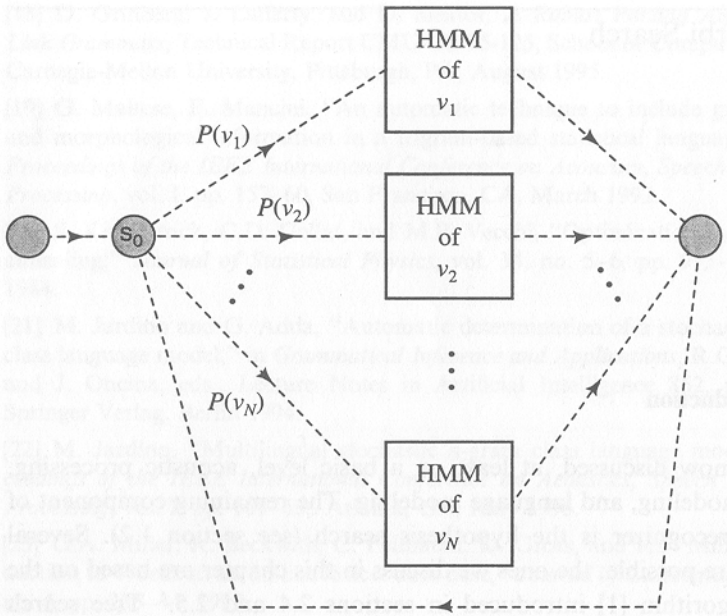

Figure 5.1
Composite model of speech generation when word generation is assumed to be memoryless

First of all, consider the simplest language model, one in which all histories $w_1, w_2, \ldots, w_{i-1}$ are equivalent (see section 4.1). Then

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i) \tag{2}$$

It is immediately obvious that in this case finding $\hat{\mathbf{W}}$ amounts to searching for the most likely path through the graph of figure 5.1. This statement must be interpreted literally—we don't care what happens inside the HMM boxes.

correspond to that **W**, and then identify the word string whose set of paths has the highest probability. Fortunately, in practice it is very rare that the word string corresponding to the most probable set of paths is the one actually spoken but the one corresponding to the most probable path is not. How to estimate the error rate due to the Viterbi approximation and other shortcuts is discussed in section 6.8.

The slightly more complicated bigram language model assigns probabilities by the formula

$$P(\mathbf{W}) = P(w_1) \prod_{i=2}^{n} P(w_i|w_{i-1}) \tag{3}$$

Then $\hat{\mathbf{W}}$ is given by the most likely path through the graph of figure 5.2 (those of its null transitions not labeled by probability expressions have probability 1). Note that while figure 5.2 is somewhat more complicated than figure 5.1, the number of acoustic models (HMMs) in both is the same. The total number of states is proportional to the vocabulary size $|\mathcal{V}|$. Thus, except in cases where estimates of $P(v_i|v_j)$ cannot be obtained, one would always use the bigram model and thus conduct the search through the graph of figure 5.2 rather than that through figure 5.1.[3]

For a trigram language model,

$$P(\mathbf{W}) = P(w_1)P(w_2|w_1) \prod_{i=3}^{n} P(w_i|w_{i-2}, w_{i-1}) \tag{4}$$

the graph is considerably more complex—the number of states is proportional to $|\mathcal{V}|^2$. For $|\mathcal{V}| = 2$, figure 5.3 illustrates the transition structure. There we have marked the multiple outgoing transitions by the *generation probabilities* to which they correspond. In general, similar graphs can be derived for any equivalence classifier Φ, as long as the number of equivalence classes is finite.

How do we find the most likely paths through these graphs? No practical algorithms exist for finding the exact solution. However, if we replace the boxes in figures 5.1 through 5.3 by the corresponding HMM models, then all the figures simply represent (huge) HMMs in their own right. The Viterbi algorithm will therefore find the most likely path through such HMMs. This path will lead through a sequence of word models whose identity can then specify the recognized word string.[4]

5.3 The Beam Search

The only problem with the Viterbi approach is that for practical vocabularies (e.g., $|\mathcal{V}| = 20{,}000$) even the bigram HMM represented in figure 5.2

3. This is because the language model (3) is more accurate than model (2).

4. We have already explained in footnote 2 why this does not necessarily correspond to the string $\hat{\mathbf{W}}$ defined by (1).

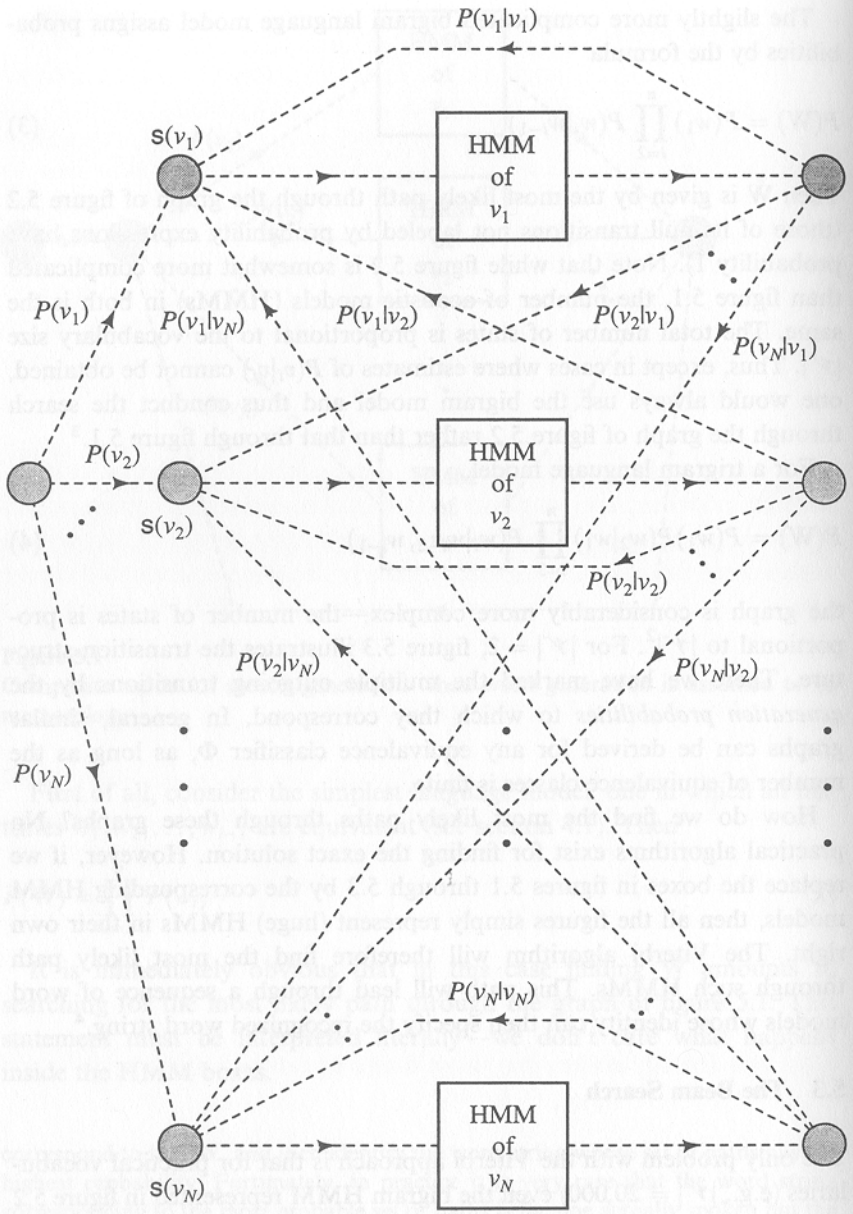

Figure 5.2
Composite model of speech generation when the generated words are assumed to depend only on the identity of the preceding word (bigram language model)

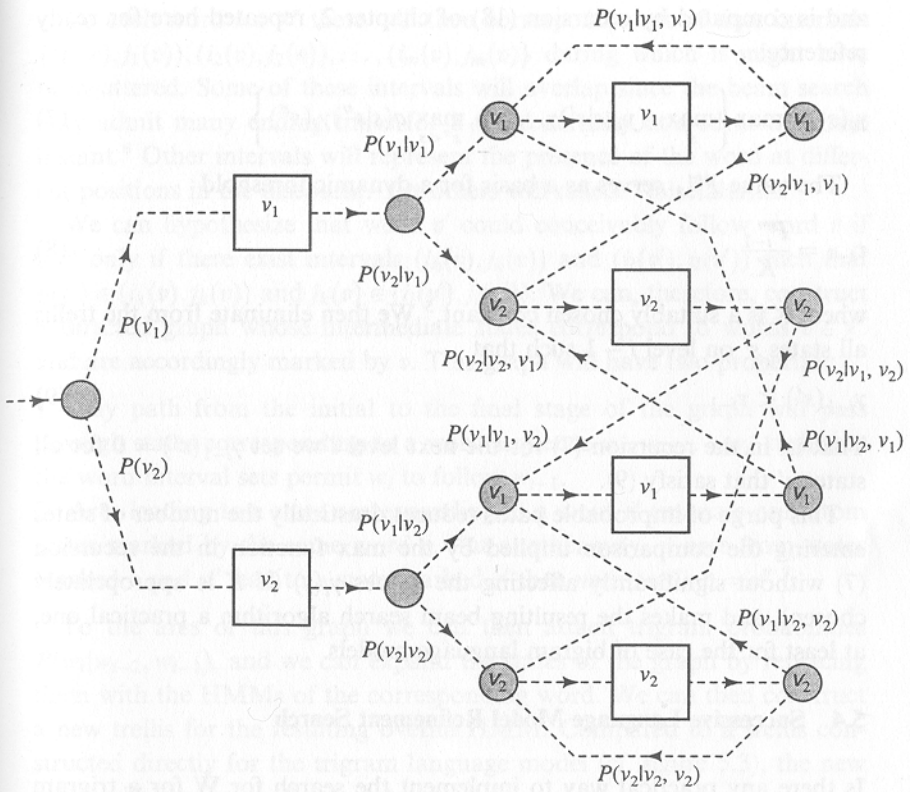

Figure 5.3
A section of the transition structure of speech generation based on a trigram language model when the vocabulary consists of two words only

has too many states. One short cut is the so-called *beam search* [2]. The idea is simple. Imagine the trellis (see section 2.3) corresponding to figure 5.2. Before carrying out the Viterbi purge (step 4 of the algorithm described in section 2.4) at stage i, we determine the maximal probability P_{i-1}^m of the states at stage $i-1$

$$P_{i-1}^m \doteq \max_s \gamma_{i-1}(s) \tag{5}$$

In (5) $\gamma_{i-1}(s)$ is defined in chapter 2

$$\gamma_i(s) \doteq \max_{s_1,\ldots,s_l} P(s_1, s_2, \ldots, s_{l-1}, s_l = s, y_1, y_2, \ldots, y_i|s_0) \tag{6}$$

and is computed by recursion (18) of chapter 2, repeated here for ready reference:

$$\gamma_i(s) = \max\left\{ \max_{s'} r(y_i, s|s')\gamma_{i-1}(s'), \max_{s''<s} q(s|s'')\gamma_i(s'') \right\} \tag{7}$$

The value P^m_{i-1} serves as a basis for a dynamic threshold

$$\tau_{i-1} = \frac{P^m_{i-1}}{K} \tag{8}$$

where K is a suitably chosen constant.[5] We then eliminate from the trellis all states s' on level $i-1$ such that

$$\gamma_{i-1}(s') < \tau_{i-1} \tag{9}$$

That is, in the recursion (7) for the next level i we set $\gamma_{i-1}(s') = 0$ for all states s' that satisfy (9).

This purge of improbable paths reduces drastically the number of states entering the comparison implied by the max function in the recursion (7) without significantly affecting the values $\gamma_i(s)$ (if K is appropriately chosen), and makes the resulting beam search algorithm a practical one, at least for the case of bigram language models.

5.4 Successive Language Model Refinement Search

Is there any practical way to implement the search for $\hat{\mathbf{W}}$ for a trigram language model? One ingenious method has been suggested by researchers at SRI [3]. We will describe the main idea.

The HMMs for each word in the vocabulary have a final state (see the far right column of states in figure 5.2). Final states of words, therefore, occur at various stages of the trellis. As the beam search proceeds, some of these states will be killed by the thresholding (9). Consider now those word final states in the trellis that remain alive. A trace-back (see Viterbi algorithm step 5 in section 2.4) from any of these final states, say from word v at (time) stage j, will lead to the initial state of the same word model, say at stage i. The pair (i,j) then identifies a time interval during which it is reasonable to admit the hypothesis that the word v might have been spoken.

5. The value of the constant K must be determined experimentally for each different application. A good start is $K \approx 100$.

To each word $v \in \mathcal{V}$ there will then correspond a set of time intervals $\{(i_1(v),j_1(v)), (i_2(v),j_2(v)), \ldots, (i_m(v),j_m(v)\}$ during which it might have been uttered. Some of these intervals will overlap since the beam search may admit many ending times for a word actually uttered at a certain instant.[6] Other intervals will represent the presence of the word at different positions in the discourse. Yet others will reflect "false alarms."

We can hypothesize that word v' could conceivably follow word v if and only if there exist intervals $(i_k(v),j_k(v))$ and $(i_l(v'),j_l(v'))$ such that $i_l(v') \in (i_k(v),j_k(v))$ and $j_k(v) \in (i_l(v'),j_l(v'))$. We can, therefore, construct a directed graph whose intermediate states correspond to words $v \in \mathcal{V}$ and are accordingly marked by v. This graph will have two properties:

1. Any path from the initial to the final stage of the graph will pass through states corresponding to a word sequence $w_1, w_2, \ldots, w_n$ such that the word interval sets permit w_i to follow w_{i-1}.
2. Arcs leading to a state corresponding to a word v' emanate only from states marked by the same word v. Put another way, if arcs from states marked v and v'' lead to a state marked v' then *necessarily* $v = v''$.[7]

To the arcs of this graph we can then attach trigram probabilities $P(w_i|w_{i-2}, w_{i-1})$, and we can expand the states of the graph by replacing them with the HMMs of the corresponding word. We can then construct a new trellis for the resulting overall HMM. Compared to a trellis constructed directly for the trigram language model (cf. figure 5.3), the new trellis will have many fewer states.

Recapitulating, to find $\hat{\mathbf{W}}$ we conduct two successive beam searches. The first, on the trellis of the bigram HMM, results in time intervals indicating the possible presence of particular words. The knowledge of these intervals gives rise to a new HMM, associated with trigram probabilities. A second beam search is then conducted over the trellis of this second HMM and results in the final choice of $\hat{\mathbf{W}}$.

6. These intervals will certainly overlap with intervals corresponding to other words $v' \in \mathcal{V}$.

7. This property assures that the transitions out of a state corresponding to v' can be associated with a unique trigram probability: Given any state s associated with v', all previous states are necessarily associated with the same word v, and so the identity of s determines the next word distribution $P(w_i|w_{i-1} = v, w_{i-2} = v')$. Figure 5.3 satisfies this property if its boxes are interpreted as states. For instance, it is possible to reach the first (second) state marked v_1 in the next-to-last column only by being previously in either of the two states marked v_1 (v_2).

Obviously, this two-pass strategy can be extended further if better and better, but more complex, language models are available. After each pass a new transition graph is prepared, allowing for the application of the next-order language model.

5.5 Search Versus Language Model State Spaces

So far have assumed that the language model used determines the complexity of the composite HMM. So the schematic structure of figure 5.2 was necessitated by a bigram language model, and that of figure 5.3 by a trigram model. If such rigidity could not be relaxed then a Viterbi-based search would be practically impossible for language models whose predictive power stems from the consideration of events in a more distant past. The only way to accommodate these models would be via the previous section's multi-pass strategy.

Fortunately this is not the case. We are perfectly free to adopt a history equivalence classification for the composite HMM that differs from that for the language model.[8] So, for instance, we are free to use the structure of figure 5.2 with a trigram language model: Observe that once the search reaches a particular stage of the corresponding trellis, the Viterbi decision assures that only a single path will lead into each state of that stage. Thus the language model probability of the next word can depend on the entire past, starting at the initial stage. So although it is true that the composite HMM of figure 5.2 will require a decision among all of the live paths ending with the same word $w_i = v$ (which is suboptimal for a trigram language model), these decisions can be based on probabilities $P(w_i = v|w_{i-2} = v', w_i = v'')$ for all live combinations (v', v''), and indeed on probabilities $P(w_i = v|\mathbf{h}_i = \mathbf{v})$ for all past live paths $\mathbf{v}$.

5.6 *N*-Best Search

In many applications it is desirable to find the N most likely word sequences given the observed acoustic data $\mathbf{A}$. For instance, we may wish to reprocess the data using more refined models whose complexity is such that the basic recognizer could not have used them directly. These models

8. We do not claim that this would be optimal, only that it might be expedient.

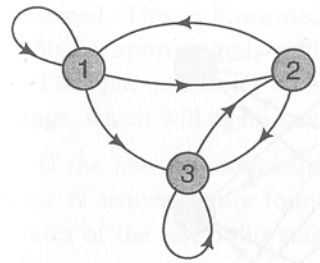

Figure 5.4
Simple HMM used to illustrate two-best search

may be acoustic, providing a sharper estimate of $P(\mathbf{A}|\mathbf{W})$, or linguistic, resulting in a better (or simply an additional) estimate of $P(\mathbf{W})$. As an instance of the latter case, the various alternate hypotheses $\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N$ could be *parsed*[9] or processed to yield their meaning. This postprocessing would then resolve the remaining N-fold ambiguity.

The required N-best search would differ from the Viterbi search in one aspect only: Instead of retaining only one path leading into each trellis state, each trellis state is split into N states, one for each of the N most likely paths leading into the unsplit state from the (split) states of the trellis's previous stage. This is illustrated for $N = 2$ in figure 5.5 based on the simple HMM of figure 5.4:

- Figure 5.5a is a diagram of a stage of the usual trellis corresponding to the HMM of figure 5.4.
- In figure 5.5b the "entry" states of the trellis stage are split in two, representing the two best paths into each entry state. To each of the split states would normally be attached the probability of the single path leading into it. The transitions out of the split states into the next stage are identical within each pair.
- Into the "exit" states of the trellis of figure 5.5b there now lead multiple paths, of which only the two most likely (for each state) should be

9. By parsing we mean supplying a phrase structure analysis of the sentence. This is usually done by means of a parse tree whose nodes are annotated by phrase markers (such as NOUN PHRASE, VERB PHASE, PREPOSITIONAL PHRASE, etc.) identifying the type of the word sequence found at the leaves of the subtree emanating from the annotated node.

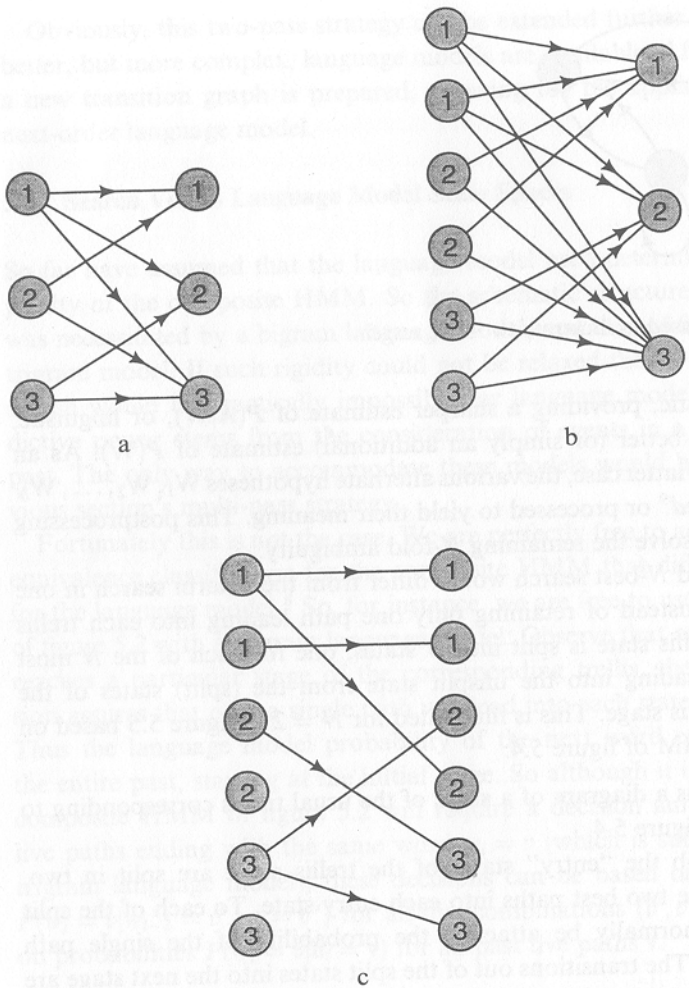

Figure 5.5
Illustration of two-best search based on the HMM of figure 5.4
Figure 5.5a
Trellis stages corresponding to the HMM of figure 5.4
Figure 5.5b
Entry states of the trellis stage of figure 5.5a split so that the identically marked entry states correspond to the best and second-best paths into the corresponding state of the stage of figure 5.5a
Figure 5.5c
Exit states of the stage of figure 5.5b split so that the identically marked exit states correspond to the best and second-best paths into the corresponding state of the stage of figure 5.5b

retained. This is illustrated in figure 5.5c where the exit states of figure 5.5b are appropriately split.

- The split exit states of figure 5.5c become the entry states of the next stage, which will again have the appearance of Figure 5.5b.

If the above modification of the Viterbi algorithm is carried out, the best N sequences are found by determining the N most probable (split) states of the last trellis stage and by retracing the trellis paths leading to them.

5.7 A Maximum Probability Lattice

In section 5.4, describing a search based on successive refinement of the language model, we have encountered a structure commonly referred to as a *lattice* (we didn't use that term). The lattice indicated which words were likely to have been spoken at which time intervals. We can create a lattice in yet another way based on a maximum a posteriori probability approach.

Let $\mathcal{S}(v)$ denote the set of states of the composite HMM (e.g., that of figure 5.2) which are internal to the HMM for word v. We can then define the probability

$$P^*\{s^i \in \mathcal{S}(v)\} \doteq \text{the probability that } \mathbf{A} = a_1, a_2, \ldots, a_m \text{ was produced and at the } i^{th} \text{ trellis stage the HMM was in one of the states of } \mathcal{S}(v).$$

It follows directly from the discussion of section 2.7 that

$$P^*\{s^i \in \mathcal{S}(v)\} = \sum_{s \in \mathcal{S}(v)} \alpha_i(s)\beta_i(s)$$

We can therefore determine, for each (time) stage i of the trellis, the set of the l most probable words $\mathcal{V}_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,l}\}$ in the process of being uttered at that time i. The number l can either be a fixed constant and

$$P^*\{s^i \in \mathcal{S}(v_{i,j})\} \geq \max_{v \notin \mathcal{V}_i} P^*\{s^i \in \mathcal{S}(v)\}, j = 1, 2, \ldots, l$$

or it can be determined by use of a suitably chosen threshold τ:

$$v_{i,1} = \arg\max_v P^*\{s^i \in \mathcal{S}(v)\}$$

$$\mathcal{V}_i = \{v : P^*\{s^i \in \mathcal{S}(v)\} \geq P^*\{s^i \in \mathcal{S}(v_{i,1})\} \times \tau\}$$

The sets $\{v_{i,1}, v_{i,2}, \ldots, v_{i,l_i}\}, i = 1, 2, \ldots, m$ then constitute a time-aligned map of words whose utterance was a likely cause of the observed sequence **A**. This is the basis of the maximum probability lattice we are after.

Obviously, to produce such a lattice we will have to cope with the computational problems presented by the enormity of the composite HMM that corresponds to the recognition task. We will thus need to use thresholding methods analogous to those that defined the beam search of section 5.3.

5.8 Additional Reading

Another multiple-pass search (section 5.4) is presented in reference [4]. Ney and Aubert consider the general problem of dynamic programming searches [5].[10] Finally Kogan considers the problem of early decision in Viterbi decoding [6].[11]

References

[1] A.J. Viterbi, "Error bounds for convolutional codes and an asymmetrically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–67, 1967.

[2] B.T. Lowerre, *The Harpy Speech Recognition System*, Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[3] H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub, "Large-vocabulary dictation using SRI's Decipher Speech Recognition System: Progressive Search Techniques," *Proceedings of the Spoken Language Systems Technology Workshop*, Massachusetts Institute of Technology, Cambridge, MA, January 1993.

[4] R. Schwartz, L. Nguyen, and J. Makhoul, "Multiple-pass search strategies," in *Automatic Speech and Speaker Recognition*, C-H. Lee, F.K. Soong, and K.K. Paliwal, eds., pp. 429–56, Klewer Academic Publishers, Norwell, MA, 1996.

10. Viterbi decoding can be shown to be a variant of dynamic programming search.

11. The way we described it in chapter 2, the trace-back that determines the decoding decision is carried out only when the entire utterance was processed by the algorithm. The question is whether the speech recognizer could make its initial decisions earlier, and if so, what conditions must be met.

[5] H. Ney and X. Aubert, "Dynamic programming search: from digit strings to large vocabulary word graphs," in *Automatic Speech and Speaker Recognition*, C-H, Lee, F.K. Soong, and K.K. Paliwal, eds., pp. 385–413, Klewer Academic Publishers, Norwell, MA, 1996.

[6] J.A. Kogan, "Exact Viterbi recognition of hidden Markovian sequences through the most informative stopping times," in *Image Models (and Their Speech Model Cousins)*, S.E. Levinson and L. Shepp, eds., pp. 115–30, Springer-Verlag, New York, 1996.

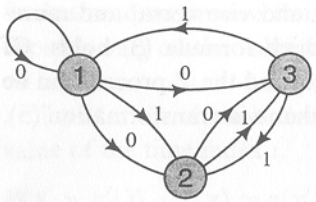

Figure 2.3
Hidden Markov model representation attaching outputs to transitions

the memory of observables is unlimited (except in degenerate cases). That is, in general, for all $k \geq 2$,

$$P(y_{k+1}|y_1, y_2, \ldots, y_k) \neq P(y_{k+1}|y_j, \ldots, y_k) \qquad k \geq j \geq 2$$

We will frequently find it convenient to regard the HMM as having multiple transitions between pairs of states, each associated with a different output symbol generated, with probability 1, when that transition is taken. This is directly equivalent to the formulation of the present section. Figure 2.3 gives an example that can generate the same random process as figure 2.2.[2]

This view has the advantage of allowing us to provide each transition of the entire HMM with a different identifier t and to define an output function $Y(t)$ that assigns to t a unique output symbol taken from the alphabet $\mathcal{Y}$.

We then denote by $L(t)$ and $R(t)$ the source and target states of the transition t, respectively. We let $p(t)$ denote the probability that the state $L(t)$ is exited via the transition t, so that for all $s \in \mathcal{S}$,

$$\sum_{t:L(t)=s} p(t) = 1$$

The correspondence between the two ways of viewing an HMM is given by the relationship

$$p(t) = q(Y(t)|L(t), R(t))p(R(t)|L(t)) \tag{7}$$

When transitions determine outputs, the probability $P(y_1, y_2, \ldots, y_k)$ becomes equal to the sum of the products $\prod_{i=1}^{k} p(t_i)$ over all transition sequences $t_1, \ldots, t_k$ such that $L(t_1) = s_0$, $Y(t_i) = y_i$, and $R(t_i) = L(t_{i+1})$ for $i = 1, \ldots, k$, or, formally,

2. Assuming that in figure 2.2 $q(1|1,1) = q(1|1,3) = q(0|3,1) = q(0|3,2) = 0$

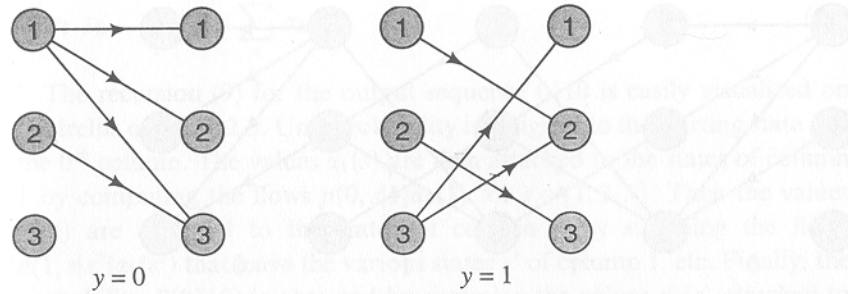

Figure 2.4
Two different trellis stages corresponding to the binary HMM of figure 2.3

$$P(y_1, y_2, \ldots, y_k) = \sum_{\mathcal{S}(y_1, y_2, \ldots, y_k)} \prod_{i=1}^{k} p(t_i)$$

where

$$\mathcal{S}(y_1, y_2, \ldots, y_k) = \{t_1, \ldots, t_k : L(t_1) = s_0, Y(t_i) = y_i, R(t_i) = L(t_{i+1}) \text{ for } i = 1, \ldots, k\}$$

In the following sections we will take whichever point of view (multiple transitions between states s and s', or multiple possible outputs generated by the single transition $s \rightarrow s'$) will be more convenient for the problem at hand.

2.3 The Trellis

There is an easy way to calculate the probability $P(y_1, y_2, \ldots, y_k)$ with the help of a *trellis*. The trellis places in evidence the time evolution of the process that generated the sequence $y_1, y_2, \ldots, y_k$. It consists of the concatenation of elementary *stages* determined by the particular outputs y_i. There are as many different elementary stages as there are different output symbols. Figure 2.4 shows the stages for $y_i = 0$ and $y_i = 1$ corresponding to the HMM of figure 2.3. Note that a stage for y_i consists of left and right columns of c states connected by those transitions t whose output $Y(t) = y_i$. Figure 2.5 shows the trellis corresponding to the output sequence 0110. The required probability $P(0110)$ is equal to the sum of the probabilities of all complete paths through the trellis (those ending in the last column) that start in the obligatory starting state. Figure 2.6 shows these paths for $s_0 = 1$.

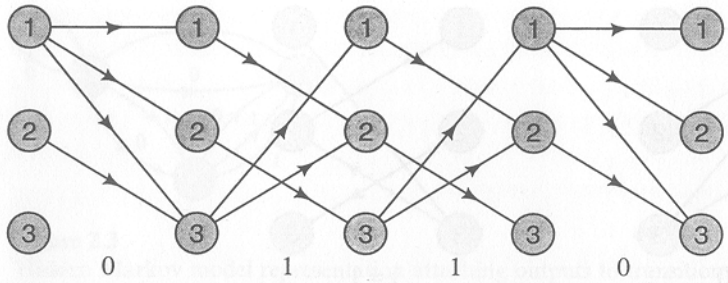

Figure 2.5
Trellis for output sequence 0110 generated by the HMM of figure 2.3

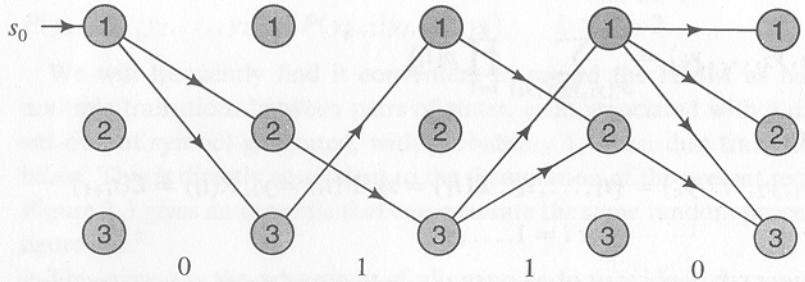

Figure 2.6
Trellis of figure 2.5 purged of all paths that could not have generated the output sequence 0110

The probability $P(y_1, y_2, \ldots, y_n)$ can be obtained recursively. Define the probabilities

$$\alpha_i(s) \doteq P(y_1, y_2, \ldots, y_i, s_i = s) \tag{8}$$

Then with boundary conditions

$\alpha_0(s) = 1$ for $s = s_0$ and $\alpha_0(s) = 0$ otherwise

we get the recursion

$$\alpha_i(s) = \sum_{s'} p(y_1, s|s')\alpha_{i-1}(s') \tag{9}$$

where we use the simplified notation

$$p(y_i, s|s') = q(y_i|s', s)p(s|s') \tag{10}$$

By definition (8) the desired probability then is

$$P(y_1, y_2, \ldots, y_k) = \sum_s \alpha_k(s) \tag{11}$$

The recursion (9) for the output sequence 0110 is easily visualized on the trellis of figure 2.5. Unit probability is assigned to the starting state 1 in the 0^{th} column. The values $\alpha_1(s)$ are then attached to the states of column 1 by computing the flows $p(0, s|1)\alpha_0(1)$ for $s \in \{1, 2, 3\}$. Then the values $\alpha_2(s)$ are attached to the states of column 2 by summing the flows $p(1, s|s')\alpha_1(s')$ that leave the various states s' of column 1, etc. Finally, the probability $P(0110)$ is obtained by summing the values $\alpha_4(s)$ attached to the states of the fourth column.

2.4 Search for the Likeliest State Transition Sequence

Given an observed output sequence $y_1, y_2, \ldots, y_k$, which state sequence $s_1, s_2, \ldots, s_k$ is most likely to have caused it? Noting that

$$P(s_1, s_2, \ldots, s_k|y_1, y_2, \ldots, y_k, s_0) = \frac{P(s_1, s_2, \ldots, s_k, y_1, y_2, \ldots, y_k|s_0)}{P(y_1, y_2, \ldots, y_k|s_0)}$$

we see that we want to find the sequence $s_1, s_2, \ldots, s_k$ maximizing the numerator of the right-hand side of the above equation.

Observe that because the underlying state process is Markov, then for all i,

$$P(s_1, \ldots, s_i, s_{i+1}, \ldots, s_k, y_1, y_2, \ldots, y_i, y_{i+1}, \ldots, y_k|s_0)$$
$$= P(s_1, s_2, \ldots s_i, y_1, y_2, \ldots, y_i|s_0)P(s_{i+1}, \ldots, s_k, y_{i+1}, \ldots, y_k|s_i)$$

so that

$$\max_{s_1, s_2, \ldots, s_k} P(s_1, \ldots, s_i, s_{i+1}, \ldots, s_k, y_1, y_2, \ldots, y_i, y_{i+1}, \ldots, y_k|s_0)$$
$$= \max_{s_i, s_{i+1}, \ldots, s_k} \Big[P(s_{i+1}, \ldots, s_k, y_{i+1}, \ldots, y_k|s_i) \tag{12}$$
$$\times \max_{s_1, s_2, \ldots, s_{i-1}} P(s_1, s_2, \ldots, s_{i-1}, s_i, y_1, y_2, \ldots, y_i|s_0) \Big]$$

Defining

$$\gamma_i(s_i) \doteq \max_{s_1, s_2, \ldots, s_{i-1}} P(s_1, s_2, \ldots s_i, y_1, y_2, \ldots, y_i|s_0) \tag{13}$$

and substituting into (12), we get

$$\max_{s_1,s_2,\ldots,s_k} P(s_1,\ldots,s_i,s_{i+1},\ldots,s_k,y_1,y_2,\ldots,y_i,y_{i+1},\ldots,y_k|s_0)$$
$$= \max_s \left\{ \max_{s_{i+1},\ldots,s_k} [P(s_{i+1},\ldots,s_k,y_{i+1},\ldots,y_k|s)\gamma_i(s)] \right\} \quad (14)$$

Therefore, we can find the maximally likely sequence $s_1,\ldots,s_{i-1},s_i,s_{i+1},\ldots,s_k$ by finding first for each state s on trellis level i the most likely state sequence $s_1(s),\ldots,s_{i-1}(s)$ leading into s, then finding the most likely sequence $s_{i+1}(s),\ldots,s_k(s)$ leading out of s, and finally finding the state s on level i for which the complete sequence $s_1(s),\ldots,s_{i-1}(s)$, $s_i = s$, $s_{i+1}(s)$, $\ldots,s_k(s)$ has the highest probability.

This can be done recursively. In fact, it follows directly from definition (13) and the HMM defining rule (10) that

$$\begin{aligned}\gamma_i(s_i) &= \max_{s_1,\ldots,s_{i-1}} P(s_1,s_2,\ldots,s_i,y_1,y_2,\ldots,y_i|s_0)\\ &= \max_{s_{i-1}} p(y_i,s_i|s_{i-1}) \max_{s_1,\ldots,s_{i-2}} P(s_1,s_2,\ldots,s_{i-1},y_1,y_2,\ldots,y_i|s_0) \quad (15)\\ &= \max_{s_{i-1}} p(y_i s_i|s_{i-1})\gamma_{i-1}(s_{i-1})\end{aligned}$$

Noticing from (13) that

$$\max_{s_1,\ldots,s_k} P(s_1,s_2,\ldots,s_k,y_1,y_2,\ldots,y_k|s_0) = \max_s \gamma_k(s)$$

equation (15) leads directly to the *Viterbi algorithm* [4] that finds a maximizing state sequence (there may be more than one) by finding the maximizing sequence $s_1(s),\ldots,s_{i-1}(s)$, $s_i = s$ (whose probability is $\gamma_i(s)$) for each state s on successive levels i, and finally deciding at level k from among the competing sequences $s_1(s),\ldots,s_{k-1}(s)$, $s_k = s$, $s \in \mathcal{S}$.

Here then is a complete statement of the Viterbi algorithm.

The Viterbi Algorithm

1. Set $\gamma_0(s_0) = 1$.
2. Use (15) to compute $\gamma_1(s)$ for all states s of the trellis's first column, that is,

$$\gamma_1(s) = \max_{s'} p(y_1,s|s')\gamma_0(s') = p(y_1,s|s_0)$$

since $\gamma_0(s') = 0$ for all $s' \neq s_0$.

3. Compute $\gamma_2(s)$ for all states s of the trellis's second column, that is,

$$\gamma_2(s) = \max_{s'} p(y_2,s|s')\gamma_1(s')$$

Purge all transitions from states s' in the first column to states s in the second column for which $\gamma_2(s) > p(y_2,s|s')\gamma_1(s')$. If more than one transition into state s remains, select (arbitrarily) one to keep and purge the rest.

4. In general, compute $\gamma_i(s)$ for all states s of the trellis's i^{th} column. Purge all transitions from states s' in the $i-1^{th}$ column to states s in the i^{th} column for which $\gamma_i(s) > p(y_i,s|s')\gamma_{i-1}(s')$. Then purge all but one of the remaining transitions into s.
5. Find the state s in the trellis's k^{th} column for which $\gamma_k(s)$ is maximal. In the purged trellis, trace back from this state s to the initial state s_0 in the 0^{th} column along the remaining transitions. The states $s_1,s_2,\ldots,s_k = s$ encountered along this path constitute a most likely state sequence.

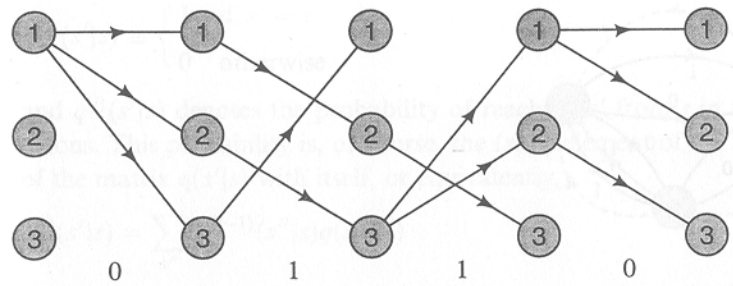

Figure 2.7
Possible result of applying the Viterbi algorithm to the trellis of figure 2.5

Figure 2.7 shows an example of a possible effect of the Viterbi algorithm on the trellis of figure 2.5 that corresponds to the observed sequence 0110. Assuming that the largest $\gamma_4(s)$ is achieved for $s = 3$, the traceback gives as a solution the state sequence 12323.

2.5 Presence of Null Transitions

In many applications, and in speech recognition in particular, it is useful to introduce the possibility of *null* transitions that change state but result in no output. In figure 2.8 we have added null transitions (interrupted lines) to the HMM of figure 2.3.

Since a trellis stage corresponds to a single output, null transitions take place between the states in the initial column of a stage. Figure 2.9 is then the trellis corresponding to the output 0110.

We must now make the appropriate changes in our formulas to accommodate the presence of null transitions. The changes made in this section are completely general and result in defining an equivalent HMM that