Chapter 8

More on HMMs

In this chapter, we treat the two central algorithms for HMMs: *Viterbi* and *Baum-Welch*. The Viterbi algorithm is used to calculate the most likely transition sequence for some string, while the Baum-Welch—or *forward-backward* algorithm—is used to train an HMM.

8.1 Viterbi algorithm

Given some string, the Viterbi algorithm is an efficient means for finding the most likely path through some HMM. More formally, it calculates the following:

$$\arg\max_{s_{1,t}} \mathrm{P}(s_{1,t}|w_{1,t-1}) \tag{8.1}$$

The basic idea is straightforward. Starting from the null string, one calculates the most likely sequence of states that could produce some string assuming one could end in any possible states. For each point, one need only consider how one gets to the state in question from the previous most likely state. Consider the HMM in figure 7.3 on page 59, repeated here as 8.1.

Let's consider how we might use the Viterbi algorithm to compute the most likely path for the string *a a a*. First, we determine the possible ways of producing the empty string and end in each of the states of the HMM: s_1 and s_2. This amounts to determining what the probability of starting in each state is:

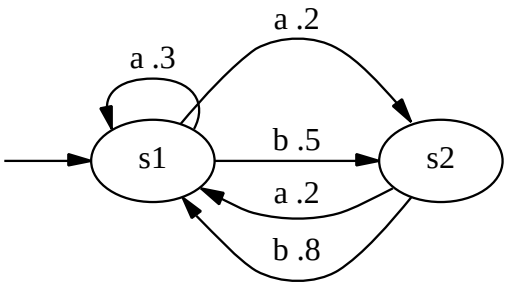

Figure 8.1: Multiple arcs

(8.2)

	[]
s_1	$\emptyset, 1$
s_2	$\emptyset, 0$

We now extend this by considering what is the most likely path to each of the possible states given the first symbol in the string: a. For s_1, the most likely path is either $s_1 \rightarrow s_1$ with a probability of $1 \times .3 = .3$ or $s_2 \rightarrow s_1$ with a probability of $0 \times .2 = 0$. Hence the first path fills the first cell of the relevant column. For s_2, the most likely path is either $s_1 \rightarrow s_2$ with a probability of $1 \times .2 = .2$ or $s_2 \rightarrow s_2$ with a probability of $0 \times 0 = 0$. Hence the first path here also fills the second cell.

(8.3)

	[]	$[a]$
s_1	$\emptyset, 1$	$s_1, .3$
s_2	$\emptyset, 0$	$s_1, .2$

We continue in like fashion. To determine the most likely path for $a\ a$, we consider how to extend each of the paths already determined up to each possible state. For s_1, we consider whether $s_1 \rightarrow s_1 \rightarrow s_1$ $(.3 \times .3 = .09)$ has a greater or lower overall probability than $s_1 \rightarrow s_2 \rightarrow s_1$ $(.2 \times .2 = .04)$. Analogously, for s_2, we compare $s_1 \rightarrow s_1 \rightarrow s_2$ $(.3 \times .2 = .01)$ to $s_1 \rightarrow s_2 \rightarrow s_2$ $(.2 \times 0 = 0)$. These comparisons result in the following extension of the chart.

(8.4)

	[]	[a]	[a a]
s_1	$\emptyset, 1$	$s_1, .3$	$s_1 \rightarrow s_1, .04$
s_2	$\emptyset, 0$	$s_1, .2$	$s_1 \rightarrow s_1, .01$

The third stage proceeds in the same way.

(8.5)

	[]	[a]	[a a]	[a a a]
s_1	$\emptyset, 1$	$s_1, .3$	$s_1 \rightarrow s_1, .04$	$s_1 \rightarrow s_1 \rightarrow s_1, .012$
s_2	$\emptyset, 0$	$s_1, .2$	$s_1 \rightarrow s_1, .01$	$s_1 \rightarrow s_1 \rightarrow s_1, .008$

Finally, we select the most likely path from the final column of the chart by choosing the final state that exhibits the highest probability overall: $s_1 \rightarrow s_1 \rightarrow s_1 \rightarrow s_1$.

We can express this all formally as follows.[1] Let $\sigma_i(t)$ represent the most likely state sequence that generates $w_1 \ldots w_{t-1}$ ending up in state s_i. Formally:

$$\sigma_i(t) \stackrel{\text{def}}{=} \arg\max_{s_{1,t}} \mathrm{P}(w_{1,t-1}, s_{1,t-1}, S_t = s^i) \tag{8.6}$$

We can use this recursively. Where $t = 1$, we have:

$$\sigma_i(1) = s^i \tag{8.7}$$

When $t > 1$, we have:

$$\sigma_i(t+1) = \sigma_j(t) \circ s^i, \text{ where } j = \arg\max_{k=1}^{\sigma} \mathrm{P}(\sigma_k(t))\mathrm{P}(s^k \stackrel{w_t}{\rightarrow} s^i) \tag{8.8}$$

where $\circ$ stands for concatenation. This algorithm is far more efficient than calculating every possible path in toto and selecting the one with the highest probability.[2]

8.2 Baum-Welch algorithm

A really important algorithm for HMMs is the *Baum-Welch* or *forward-backward* algorithm. This allows one to deduce the probabilities in a HMM from data directly.

[1] We follow Charniak (1993) here (p.56).

[2] Note that the Viterbi algorithm is really an instance of chart parsing and dynamic programming generally.

To do this, we must first be able to calculate the probability of producing any string in two different ways: the *forward* algorithm, and the *backward* algorithm.

8.2.1 The 'forward' algorithm

Following Charniak (1993) again, we define $\alpha_i(t)$ as the probability of producing $w_{1,t-1}$ and ending up in s^i.

$$\alpha_i(t) \stackrel{\text{def}}{=} \mathrm{P}(w_{1,t-1}, S_t = s^i), \text{ where } t > 1 \tag{8.9}$$

We can compute this value recursively. There will be a "base" case and a recursive rule. We calculate the base value and then use the recursive rule to fill in a series of values up to the final one. The base case here is determined by the starting probability of node s^i.

$$\alpha_i(1) = \pi_i \tag{8.10}$$

From the base case, we can calculate the other cases recursively:

$$\alpha_j(t+1) = \sum_{i=1}^{\sigma} \alpha_i(t)\mathrm{P}(s^i \stackrel{w_t}{\rightarrow} s^j) \tag{8.11}$$

Let's consider an example. What is the forward probability of $[a\ b\ a]$ given the HMM in figure 8.1? We first determine the forward probability for the null string at time 1, i.e. the starting probability for each state.

	[]
s_1	1
s_2	0

(8.12)

We next calculate forward probabilities for $[a]$, given the base cases calculated so far. For $\alpha_1(2)$, we have:

$$\begin{aligned} \alpha_1(2) &= \alpha_1(1)\mathrm{P}(s^1 \stackrel{a}{\rightarrow} s^1) + \alpha_2(1)\mathrm{P}(s^2 \stackrel{a}{\rightarrow} s^1) \\ &= (1 \times .3) + (0 \times .2) \\ &= .3 \end{aligned} \tag{8.13}$$

For $\alpha_2(2)$, we have:

$$\begin{aligned} \alpha_2(2) &= \alpha_1(1)\mathrm{P}(s^1 \xrightarrow{a} s^2) + \alpha_2(1)\mathrm{P}(s^2 \xrightarrow{a} s^2) \\ &= (1 \times .2) + (0 \times 0) \\ &= .2 \end{aligned} \tag{8.14}$$

Hence:

	[]	$[a]$
s_1	1	.3
s_2	0	.2

(8.15)

We continue in similar fashion. To calculate $\alpha_1(3)$ for $[a\ b]$:

$$\begin{aligned} \alpha_1(3) &= \alpha_1(2)\mathrm{P}(s^1 \xrightarrow{b} s^1) + \alpha_2(2)\mathrm{P}(s^2 \xrightarrow{b} s^1) \\ &= (.3 \times 0) + (.2 \times .8) \\ &= .16 \end{aligned} \tag{8.16}$$

To calculate $\alpha_2(3)$ for $[a\ b]$:

$$\begin{aligned} \alpha_2(3) &= \alpha_1(2)\mathrm{P}(s^1 \xrightarrow{b} s^2) + \alpha_2(2)\mathrm{P}(s^2 \xrightarrow{b} s^2) \\ &= (.3 \times .5) + (.2 \times 0) \\ &= .15 \end{aligned} \tag{8.17}$$

These extend the table as follows:

	[]	$[a]$	$[a\ b]$
s_1	1	.3	.16
s_2	0	.2	.15

(8.18)

Finally, we calculate $\alpha_1(4)$ and $\alpha_2(4)$ for $[a\ b\ a]$:

$$\begin{aligned} \alpha_1(4) &= \alpha_1(3)\mathrm{P}(s^1 \xrightarrow{a} s^1) + \alpha_2(3)\mathrm{P}(s^2 \xrightarrow{a} s^1) \\ &= (.16 \times .3) + (.15 \times .2) \\ &= .078 \end{aligned} \tag{8.19}$$

$$\begin{aligned} \alpha_2(4) &= \alpha_1(3)\mathrm{P}(s^1 \xrightarrow{a} s^2) + \alpha_2(3)\mathrm{P}(s^2 \xrightarrow{a} s^2) \\ &= (.16 \times .2) + (.15 \times 0) \\ &= .032 \end{aligned} \tag{8.20}$$

These complete the table as follows:

	[]	$[a]$	$[a\ b]$	$[a\ b\ a]$
s_1	1	.3	.16	.078
s_2	0	.2	.15	.032

(8.21)

The overall probability of the string is simply the sum of the rightmost column of the table: .11.

8.2.2 The 'backward' algorithm

We also need to be able to calculate the *backward* probability of a string. The difference is that forward probability refers to the final state of the HMM, while backward probability refers to the *initial* state of the HMM. Thus $\beta_i(t)$ refers to the overall probability of producing $w_{t,n}$ where the HMM is in state s^i at time t.

The base case is straightforward. The probability of producing ϵ (nothing) is 1, regardless of what state you are in.

$$\beta_i(n+1) = \mathrm{P}(\epsilon | S_{n+1} = s^i) = 1 \tag{8.22}$$

The recursive case is as follows:

$$\beta_i(t-1) = \sum_{j=1}^{\sigma} \mathrm{P}(s^j \overset{w_{t-1}}{\rightarrow} s^j)\beta_j(t) \tag{8.23}$$

Let's consider an example. How do we calculate the *backward* probability of the same string [*a b a*] with the same HMM? First, we calculate $\beta_1(4)$ and $\beta_2(4)$. As defined in equation 8.22, these are both 1.

	[]
s_1	1
s_2	1

(8.24)

We then move right to left working through the string. We now calculate $\beta_1(3)$ and $\beta_2(3)$ for [*a*]. For the former, we have:

$$\begin{aligned} \beta_1(3) &= \mathrm{P}(s^1 \overset{a}{\rightarrow} s^1)\beta_1(4) + \mathrm{P}(s^1 \overset{a}{\rightarrow} s^2)\beta_2(4) \\ &= (.3 \times 1) + (.2 \times 1) \\ &= .5 \end{aligned} \tag{8.25}$$

For the latter, we have:

$$\begin{aligned} \beta_2(3) &= \mathrm{P}(s^2 \overset{a}{\rightarrow} s^1)\beta_1(4) + \mathrm{P}(s^2 \overset{a}{\rightarrow} s^2)\beta_2(4) \\ &= (.2 \times 1) + (0 \times 1) \\ &= .2 \end{aligned} \tag{8.26}$$

This fills out the chart as follows.

(8.27)

	[]	[a]
s_1	1	.5
s_2	1	.2

We now calculate $\beta_1(2)$ and $\beta_2(2)$ for $[b\ a]$. For the former, we have:

$$\begin{aligned} \beta_1(2) &= \mathrm{P}(s^1 \xrightarrow{b} s^1)\beta_1(3) + \mathrm{P}(s^1 \xrightarrow{b} s^2)\beta_2(3) \\ &= (0 \times .5) + (.5 \times .2) \\ &= .1 \end{aligned} \tag{8.28}$$

For the latter:

$$\begin{aligned} \beta_2(2) &= \mathrm{P}(s^2 \xrightarrow{b} s^1)\beta_1(3) + \mathrm{P}(s^2 \xrightarrow{b} s^2)\beta_2(3) \\ &= (.8 \times .5) + (0 \times .2) \\ &= .4 \end{aligned} \tag{8.29}$$

This augments the chart as we would expect:

(8.30)

	[]	[a]	[b a]
s_1	1	.5	.1
s_2	1	.2	.4

Finally, we calculate $\beta_1(1)$ and $\beta_2(1)$ for $[a\ b\ a]$. At this stage we must factor in the start probabilities: $\pi_1 = 1$ and $\pi_2 = 0$.

$$\begin{aligned} \beta_1(1) &= \pi_1\mathrm{P}(s^1 \xrightarrow{a} s^1)\beta_1(2) + \pi_1\mathrm{P}(s^1 \xrightarrow{a} s^2)\beta_2(2) \\ &= (1 \times .3 \times .1) + (1 \times .2 \times .4) \\ &= .11 \end{aligned} \tag{8.31}$$

$$\begin{aligned} \beta_2(1) &= \pi_2\mathrm{P}(s^2 \xrightarrow{a} s^1)\beta_1(2) + \pi_2\mathrm{P}(s^2 \xrightarrow{a} s^2)\beta_2(2) \\ &= (0 \times .2 \times .1) + (0 \times 0 \times .4) \\ &= 0 \end{aligned} \tag{8.32}$$

This completes the chart as follows:

(8.33)

	[]	[a]	[b a]	[a b a]
s_1	1	.5	.1	.11
s_2	1	.2	.4	0

As with forward probabilities, the overall probability of the string is obtained by summing the complete backward probabilities (the rightmost column): .11. The value here is the same as that obtained from the forward probabilities.

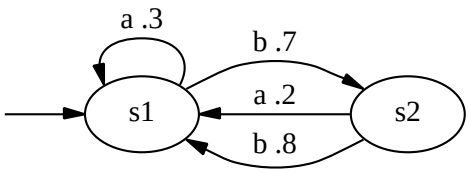

Figure 8.2: A simple Markov chain

8.2.3 Training a simple Markov chain

We can use forward and backward probabilities to train a HMM. Before showing how to do this, however, let's consider the basic training idea with respect to the simpler Markov chain. For concreteness, we will use the chain in figure 7.1 on page 56, repeated in figure 8.2. What we want to be able to do is *derive* appropriate probability values from data. The basic idea is to take some Markov chain, expose it to some specific data and then calculate the probability values from that exposure.

The math here is actually very straightforward. You calculate how many times each arc is traversed for the training data. The probability of any arc can be derived from the fact that the probabilities of the arcs leaving any particular state form a probability distribution. Hence the probability of any arc can be estimated by dividing the number of times that arc is traversed by the number of times all arcs leaving that state are traversed.

$$\mathrm{P}_e(s^i \xrightarrow{w^k} s^j) = \frac{|s^i \xrightarrow{w^k} s^j|}{\sum_{l=1,m=1}^{\sigma,\omega} |s^i \xrightarrow{w^m} s^l|} \tag{8.34}$$

Taking the Markov chain in figure 8.2 and the training text *abababb*, we get the following unique path through the chain.

$$s^1 \xrightarrow{a} s^1 \xrightarrow{b} s^2 \xrightarrow{a} s^1 \xrightarrow{b} s^2 \xrightarrow{a} s^1 \xrightarrow{b} s^2 \xrightarrow{b} s^1 \tag{8.35}$$

We get the following counts. The probability of each arc is then estimated straightforwardly.

(8.36)

arc	count	probability
$s^1 \xrightarrow{a} s^1$	1	.25
$s^1 \xrightarrow{b} s^2$	3	.75
$s^2 \xrightarrow{a} s^1$	2	.66
$s^2 \xrightarrow{b} s^1$	1	.33

For example, we have two arcs leaving s^1 four times. The arc labeled a accounts for 1 of those; hence it has a probability of .25.

8.2.4 Putting them together awkwardly

The training method for Markov chains works because we know precisely how many times each arc is traversed for any training sequence. This is not the case for a HMM. For any particular string there can be any number of alternative paths through the HMM. This has to be accommodated in determining the count for any particular arc.

The usual method is to compute all possible paths through the HMM and their associated probabilities. We then "prorate" the count for each arc by the probability of the state sequence it occurs in. If, for example, some arc occurs twice in a state sequence that has a probability of .2, then that would add .4 to the count for that arc. This is given as:

(8.37)
$$\begin{aligned} |s^i \xrightarrow{w^k} s^j| &= \textstyle\sum_{s_{1,n+1}} \mathrm{P}(s_{1,n+1}|w_{1,n})\eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n}) \\ &= \tfrac{1}{\mathrm{P}(w_{1,n})} \textstyle\sum_{s_{1,n+1}} \mathrm{P}(s_{1,n+1}, w_{1,n})\eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n}) \end{aligned}$$

Here, $\eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n})$ is defined as the number of times $s^i \xrightarrow{w^k} s^j$ appears in a sequence $s_{1,n}$ given an output $w_{1,n}$.

This idea raises a very awkward question, however. How do we obtain the probabilities of the paths, given that we are trying to determine the probabilities of their component arcs? It turns out that this problem can be solved simply by starting with *any* probability values for the arcs (as long as they constitute a valid probability distribution).

Let's consider again the HMM in figure 8.1 and repeated yet again as figure 8.3. Imagine we are interested in determining the new arc probability values based on a training string *aabb*. This can be produced in any of three ways with each of the total sequence probabilities given.

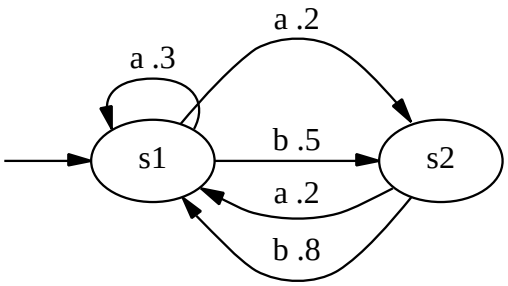

Figure 8.3: Multiple arcs

$s^1 \xrightarrow{a} s^1$	$2 * .036 + 1 * .024 + 0 * .016$	$=$	$.072 + .024 + 0$	$=$	$.096$
$s^1 \xrightarrow{a} s^2$	$0 * .036 + 1 * .024 + 1 * .016$	$=$	$0 + .024 + .016$	$=$	$.04$
$s^1 \xrightarrow{b} s^2$	$1 * .036 + 1 * .024 + 1 * .016$	$=$	$.036 + .024 + .016$	$=$	$.076$
$s^2 \xrightarrow{a} s^1$	$0 * .036 + 0 * .024 + 1 * .016$	$=$	$0 + 0 + .016$	$=$	$.016$
$s^2 \xrightarrow{b} s^1$	$1 * .036 + 1 * .024 + 1 * .016$	$=$	$.036 + .024 + .016$	$=$	$.076$
$\sum s^1 \rightarrow$	$.096 + .04 + .076$	$=$	$.212$		
$\sum s^2 \rightarrow$	$.016 + .076$	$=$	$.092$		

Figure 8.4: Results of simple training

$$
\begin{array}{ll}
s^1 \xrightarrow{a} s^1 \xrightarrow{a} s^1 \xrightarrow{b} s^2 \xrightarrow{b} s^1 & .036 \\
s^1 \xrightarrow{a} s^1 \xrightarrow{a} s^2 \xrightarrow{b} s^1 \xrightarrow{b} s^2 & .024 \\
s^1 \xrightarrow{a} s^2 \xrightarrow{a} s^1 \xrightarrow{b} s^2 \xrightarrow{b} s^1 & .016
\end{array} \tag{8.38}
$$

The estimated new probability values are then calculated per the equation above. For each path, we figure out the number of times any particular arc occurs and multiply that times the overall probability of the path. We sum across all paths. This produces the results in figure 8.4 for the example at hand:

The estimated probability values are estimated by dividing the counts

$$
\begin{array}{l|ll}
s^1 \xrightarrow{a} s^1 & \frac{.096}{.212} & = .45 \\
s^1 \xrightarrow{a} s^2 & \frac{.04}{.212} & = .19 \\
s^1 \xrightarrow{b} s^2 & \frac{.076}{.212} & = .36 \\
s^2 \xrightarrow{a} s^1 & \frac{.016}{.092} & = .17 \\
s^2 \xrightarrow{b} s^1 & \frac{.076}{.092} & = .83
\end{array}
$$

Figure 8.5: Restimated arc probabilities

	$s^2 \xrightarrow{a} s^1$	$s^2 \xrightarrow{b} s^1$	$s^1 \xrightarrow{a} s^1$	$s^1 \xrightarrow{a} s^2$	$s^1 \xrightarrow{b} s^2$	P($aabb$)
0	0.2	0.8	0.3	0.2	0.5	.076
1	0.173	0.826	0.452	0.188	0.358	.0951
2	0.092	0.907	0.528	0.126	0.345	0.1116
3	0.031	0.968	0.589	0.073	0.336	0.1275
4	0.005	0.994	0.627	0.038	0.333	0.1380
5	0.0005	0.999	0.646	0.019	0.333	0.1429
6	$2.488e-005$	0.999	0.656	0.009	0.333	0.1451
7	$5.587e-007$	0.999	0.661	0.004	0.333	0.1462
8	$6.226e-009$	0.999	0.664	0.002	0.333	0.1471
9	$3.457e-011$	0.999	0.665	0.001	0.333	0.1473

Figure 8.6: Repeated presentation of training text

by the totals for each node. This is shown in figure 8.5. Notice how the new values exhibit a probability distribution. Notice too how the overall probability of the training string increases when we compare the probabilities assigned by the original model to those assigned when the new values are substituted. With the original values, we had .076. With the new values, we get .095.

Repeated presentation of the same stimuli can result in further changes, but eventually the values will settle on a reasonable approximation of the optimal ones. The chart in figure 8.6 shows the results of presenting the same string nine times. Notice how the overall probability of *aabb* increases, yet settles toward .147.

The algorithm doesn't succeed in finding optimal values in all cases.

Sometimes it can be mislead and converge on some "local optimum", and not find the very best set of values.

8.2.5 Putting them together efficiently

A major problem with the training algorithm is that it sums over all possible paths through the HMM. Since this number can explode exponentially, we need a more efficient algorithm. We can do this by exploiting forward and backward probabilities.

The following equation does this. It sums over each arc possible at each point in the string (t).

$$|s^i \overset{w^k}{\rightarrow} s^j| = \frac{1}{\mathrm{P}(w_{1,n})} \sum_{t=1}^{n} \alpha_i(t)\mathrm{P}(s^i \overset{w^k}{\rightarrow} s^j)\beta_j(t+1) \tag{8.39}$$

Let's go through the same example; we will train the same HMM (figure 8.3) with the same string *aabb*. We first calculate the values for α and β.

(8.40)

	$\alpha(1)$	$\alpha(2)$	$\alpha(3)$	$\alpha(4)$	$\alpha(5)$
s^1	1	.3	.13	.048	.052
s^2	0	.2	.06	.065	.024

(8.41)

	$\beta(1)$	$\beta(2)$	$\beta(3)$	$\beta(4)$	$\beta(5)$
s^1	.076	.2	.4	.5	1
s^2	0	.08	.4	.8	1

We then compute the counts for each of the arcs according to equation 8.39.

(8.42)

$\|s^i \overset{w^k}{\rightarrow} s^j\|$	$=$	$\frac{1}{\mathrm{P}(w_{1,n})} \sum_{t=1}^{n} \alpha_i(t)\mathrm{P}(s^i \overset{w^k}{\rightarrow} s^j)\beta_j(t+1)$	
$\|s^1 \overset{a}{\rightarrow} s^1\|$	$=$	$\frac{1}{.074}[(1 \times .3 \times .2) + (.3 \times .3 \times .4)]$	$= 1.2972$
$\|s^1 \overset{a}{\rightarrow} s^2\|$	$=$	$\frac{1}{.074}[(1 \times .2 \times .08) + (.3 \times .2 \times .4)]$	$= .5405$
$\|s^1 \overset{b}{\rightarrow} s^2\|$	$=$	$\frac{1}{.074}[(.13 \times .5 \times .8) + (.048 \times .5 \times 1)]$	$= 1.027$
$\|s^2 \overset{a}{\rightarrow} s^1\|$	$=$	$\frac{1}{.074}[(0 \times .2 \times .2) + (.2 \times .2 \times .4)]$	$= .2162$
$\|s^2 \overset{b}{\rightarrow} s^1\|$	$=$	$\frac{1}{.074}[(.06 \times .8 \times .5) + (.065 \times .8 \times 1)]$	$= 1.027$

This produces the exact same results without having to compute all possible paths through the HMM.

(8.43)

$\lvert s^i \xrightarrow{w^k} s^j \rvert$	P_e
$\lvert s^1 \xrightarrow{a} s^1 \rvert$	.45
$\lvert s^1 \xrightarrow{a} s^2 \rvert$	.19
$\lvert s^1 \xrightarrow{b} s^2 \rvert$	.36
$\lvert s^2 \xrightarrow{a} s^1 \rvert$	.17
$\lvert s^2 \xrightarrow{b} s^1 \rvert$	.83