CHAPTER 6

Practical Implementation of Discriminative Learning

The basic development of growth transformation (GT)-based optimization for the hidden Markov model (HMM) and of the unified objective function has been presented in earlier chapters, where several practical considerations and implementation issues are left to this chapter.

6.1 COMPUTING $\Delta\gamma(i,r,t)$ IN GROWTH-TRANSFORM FORMULAS

In (5.20) computing $\Delta\gamma(i,r,t)$ involves summation over all possible superstring label sequences $s = s_1, \ldots, s_R$. The number of training tokens (sentence strings), R, is usually very large. Hence, the summation over s needs to be decomposed and simplified. To proceed, we use the notations of $s' = s_1, \ldots, s_{r-1}$, $s'' = s_{r+1}, \ldots, s_R$, $X' = X_1, \ldots, X_{r-1}$, and $X'' = X_{r+1}, \ldots, X_R$. Then, from (5.20), we have,

$$\begin{aligned}\Delta\gamma(i,r,t) &= \sum_{s'}\sum_{s_r}\sum_{s''} p\left(s',s_r,s''|X',X_r,X'';\Lambda'\right)\left(C(s',s_r,s'') - O(\Lambda')\right)\gamma_{i,r,s_r}(t) \\ &= \sum_{s_r} p\left(s_r|X_r,\Lambda'\right)\underbrace{\left[\sum_{s'}\sum_{s''} p\left(s',s''|X',X'';\Lambda'\right)\left(C(s',s_r\,s'') - O(\Lambda')\right)\right]}_{\Psi}\gamma_{i,r,s_r}(t)\end{aligned} \tag{6.1}$$

where factor Ψ is the average deviation of the accuracy count for the given string s_r. The remaining steps in simplifying the computation of $\Delta\gamma(i,r,t)$ will be separate for maximum mutual information (MMI) and minimum classification error/minimum phone error/minimum word error (MCE/MPE/MWE) because the parameter-independent accuracy count function $C(s)$ for them takes the product and summation form, respectively (as shown in Table 3.1).

6.1.1 Product Form of $C(s)$ (for MMI)

For MMI, we have $C(s) = C(s_1, \ldots, s_R) = \prod_{r=1}^{R} C(s_r) = \prod_{r=1}^{R} \delta(s_r, S_r)$ in a product form. Using $C(s', s_r, s'') = C(s_r) \cdot C(s', s'')$, we simplify factor Ψ in (6.1) to

$$\begin{aligned}\Psi &= \sum_{s'}\sum_{s''} p\left(s', s'' \middle| X', X''; \Lambda'\right)\left(C(s', s_r, s'') - O(\Lambda')\right) \\ &= C(s_r) \cdot \sum_{s'}\sum_{s''} p\left(s', s'' \middle| X', X''; \Lambda'\right) C(s', s'') - O(\Lambda') \\ &= O(\Lambda')\left(\frac{C(s_r) \cdot \sum_{s'}\sum_{s''} p(s', s'' | X', X''; \Lambda') C(s', s'')}{O(\Lambda')} - 1\right) \end{aligned} \tag{6.2}$$

The idea behind the above steps is to make use of the product form of the $C(s)$ function for canceling out common factors in both $O(\Lambda')$ and $C(s)$ functions. To proceed, we now factorize $O(\Lambda')$ as follows:

$$\begin{aligned} O(\Lambda') &= \frac{\sum_{s'}\sum_{s_r}\sum_{s''}\left[\, p\left(s', s_r, s'', X', X_r, X'' \middle| \Lambda'\right) C(s', s_r, s'')\right]}{\sum_{s'}\sum_{s_r}\sum_{s''} p\left(s', s_r, s'', X', X_r, X'' \middle| \Lambda'\right)} \\ &= \frac{\left[\sum_{s_r} p(s_r, X_r | \Lambda') C(s_r)\right]\left[\sum_{s'}\sum_{s''} p\left(s', s'', X', X'' \middle| \Lambda'\right) C(s', s'')\right]}{\left[p(X_r | \Lambda')\right]\left[p(X', X'' | \Lambda')\right]} \\ &= \sum_{s_r}\left[p\left(s_r \middle| X_r, \Lambda'\right) C(s_r)\right] \cdot \sum_{s'}\sum_{s''} p\left(s', s'' | X', X''; \Lambda'\right) C(s', s'') \\ &= p\left(S_r \middle| X_r, \Lambda'\right) \cdot \sum_{s'}\sum_{s''} p\left(s', s'' | X', X''; \Lambda'\right) C(s', s'') \end{aligned}$$

where the last step uses $C(s_r) = \delta(s_r, S_r)$. Substituting this to (6.2) then gives the simplification of

$$\begin{aligned}\Psi &= O(\Lambda')\left(\frac{C(s_r) \cdot \sum_{s'}\sum_{s''} p\left(s', s'' | X', X''; \Lambda'\right) C(s', s'')}{p(S_r | X_r, \Lambda') \cdot \sum_{s'}\sum_{s''} p\left(s', s'' | X', X'', \Lambda'\right) C(s', s'')} - 1\right) \\ &= O(\Lambda')\left(\frac{C(s_r)}{p\left(S_r \middle| X_r, \Lambda'\right)} - 1\right) \end{aligned} \tag{6.3}$$

Substituting (6.3) to (6.1) and using $C(s_r) = \delta(s_r, S_r)$ again for MMI, we obtain

$$\begin{aligned}
\Delta\gamma(i,r,t) &= O(\Lambda')\sum_{s_r} p\left(s_r\middle|X_r,\Lambda'\right)\left(\frac{C(s_r)}{p\left(S_r\middle|X_r,\Lambda'\right)}-1\right)\gamma_{i,r,s_r}(t) \\
&= O(\Lambda')\sum_{s_r,\,s_r\neq S_r} p\left(s_r\middle|X_r,\Lambda'\right)\left(\frac{C(s_r)}{p\left(S_r\middle|X_r,\Lambda'\right)}-1\right)\gamma_{i,r,s_r}(t) \\
&\quad + O(\Lambda')p\left(S_r\middle|X_r,\Lambda'\right)\left(\frac{C(S_r)}{p\left(S_r\middle|X_r,\Lambda'\right)}-1\right)\gamma_{i,r,S_r}(t) \\
&= -O(\Lambda')\sum_{s_r,\,s_r\neq S_r} p\left(s_r\middle|X_r,\Lambda'\right)\gamma_{i,r,s_r}(t) + O(\Lambda')\left(1-p(S_r\middle|X_r,\Lambda')\right)\gamma_{i,r,S_r}(t) \\
&= O(\Lambda')\left[\gamma_{i,r,S_r}(t)-\sum_{s_r} p\left(s_r\middle|X_r,\Lambda'\right)\gamma_{i,r,s_r}(t)\right] \qquad (6.4)
\end{aligned}$$

In the reestimation formulas (5.35) and (5.36), if we divide both the nominator and denominator by $O(\Lambda')$, $\Delta\gamma(i,r,t)$ in (6.4) can take a simplified form of

$$\Delta\tilde{\gamma}(i,r,t) = \left[\gamma_{i,r,S_r}(t)-\sum_{s_r} p\left(s_r\middle|X_r,\Lambda'\right)\gamma_{i,r,s_r}(t)\right] = \gamma_{i,r}^{\text{num}}(t)-\gamma_{i,r}^{\text{den}}(t) \qquad (6.5)$$

The corresponding constant D_i in the reestimation formulas (5.35) and (5.36) then becomes

$$\tilde{D}_i = D_i/O(\Lambda') \qquad (6.6)$$

Substituting this into (5.35) and (5.36), we have the GT formulas for MMI

$$\mu_i = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\left[\gamma_{i,r}^{\text{num}}(t)-\gamma_{i,r}^{\text{den}}(t)\right]x_t + \tilde{D}_i\mu_i'}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\left[\gamma_{i,r}^{\text{num}}(t)-\gamma_{i,r}^{\text{den}}(t)\right] + \tilde{D}_i} \qquad (6.7)$$

$$\Sigma_i = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\left[\gamma_{i,r}^{\text{num}}(t)-\gamma_{i,r}^{\text{den}}(t)\right](x_t-\mu_i)(x_t-\mu_i)^{\mathrm{T}} + \tilde{D}_i\Sigma_i' + \tilde{D}_i(\mu_i-\mu_i')(\mu_i-\mu_i')^{\mathrm{T}}}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\left[\gamma_{i,r}^{\text{num}}(t)-\gamma_{i,r}^{\text{den}}(t)\right] + \tilde{D}_i} \qquad (6.8)$$

This gives the classical GT/EBW-based MMI reestimation formulas described in [34, 52].

Equation (6.4) or (6.5) gives an N-best-string-based solution to computing $\Delta\gamma(i,r,t)$. This is illustrated by the string-level summation over s_r (i.e., the label sequence for token r, including both correct and incorrect strings). For N-best string-level discriminative training, the summation over s_r in (6.4) or (6.5) amounts to going through all N-best string hypotheses and is

computationally inexpensive when N is relatively small (e.g., N in the order of thousands as typical for most N-best experiments).

When a lattice instead of an explicit N-best list is provided for competing hypotheses in discriminative training, in theory, (6.4) or (6.5) can be applied just as for the N-best string based solution already discussed. This is because a lattice is nothing more than a compact representation of N-best strings. However, because N in this equivalent "N-best list" would be huge (in the order of billions or higher [57]), more efficient techniques for dealing with the summation over s_r in computing (6.4) or (6.5) will be needed. Readers are referred to Section 6.2 for details of such computation.

6.1.2. Summation Form of $C(s)$ (MCE and MPE/MWE)

Different from MMI, for MCE and MPE/MWE, we have $C(s) = C(s_1, \ldots, s_R) = \sum_{r=1}^{R} C(s_r)$, or $C(s', s_r, s'') = C(s_r) + C(s', s'')$. That is, the C function is in a summation instead of a product form. This changes the simplification steps for factor Ψ of (6.1) as follows:

$$\begin{aligned}\Psi &= \sum_{s'}\sum_{s''} p\left(s', s''|X', X''; \Lambda'\right)\left(C(s', s_r, s'') - O(\Lambda')\right) \\ &= \sum_{s'}\sum_{s''} p_{\Lambda'}\left(s', s''|X', X''\right) C(s_r) + \sum_{s'}\sum_{s''} p\left(s', s''|X', X''; \Lambda'\right) C(s', s'') \\ &\quad - \sum_{s'}\sum_{s''} p\left(s', s''|X', X''; \Lambda'\right) O(\Lambda') = C(s_r) + \sum_{s'}\sum_{s''} p\left(s', s''|X', X''; \Lambda'\right) C(s', s'') - O(\Lambda')\end{aligned} \tag{6.9}$$

The idea behind the above steps is to make use of the summation form of the $C(s)$ function for subtracting out the common terms in the $O(\Lambda')$ function. To achieve this, we decompose $O(\Lambda')$, based on its original nonrational form (3.20) or (3.22), (3.23) as follows:

$$\begin{aligned}O(\Lambda') &= \sum_{i=1}^{R} \frac{\sum_{s_i} p\left(X_i, s_i|\Lambda'\right) C(s_i)}{\sum_{s_i} p\left(X_i, s_i|\Lambda'\right)} \\ &= \frac{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right) C(s_r)}{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right)} + \sum_{i=1, i\neq r}^{R} \frac{\sum_{s_i} p\left(X_i, s_i|\Lambda'\right) C(s_i)}{\sum_{s_i} p\left(X_i, s_i|\Lambda'\right)} \\ &= \frac{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right) C(s_r)}{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right)} + \frac{\sum_{s', s''} p\left(s', s'', X', X''|\Lambda'\right) C(s', s'')}{\sum_{s', s''} p\left(s', s'', X', X''|\Lambda'\right)} \\ &= \frac{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right) C(s_r)}{\sum_{s_r} p\left(X_r, s_r|\Lambda'\right)} + \sum_{s', s''} p\left(s', s''|X', X''|\Lambda'\right) C(s', s'')\end{aligned}$$

The second term above cancels out the same term in (6.9), leading to the simplification of

$$\Psi = C(s_r) - \frac{\sum_{s_r} p\left(s_r, X_r \middle| \Lambda'\right) C(s_r)}{\sum_{s_r} p\left(s_r, X_r \middle| \Lambda'\right)} \tag{6.10}$$

Now, substituting (6.10) back to (6.1), we obtain

$$\Delta\gamma(i,r,t) = \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right) \left(C(s_r) - \frac{\sum_{s_r} p\left(X_r, s_r \middle| \Lambda'\right) C(s_r)}{\sum_{s_r} p\left(X_r, s_r \middle| \Lambda'\right)} \right) \gamma_{i,r,s_r}(t) \tag{6.11}$$

For MCE that has $C(s_r) = \delta(s_r, S_r)$, the above equation can be further simplified to

$$\Delta\gamma(i,r,t) = p\left(S_r \middle| X_r, \Lambda'\right) \left[\gamma_{i,r,S_r}(t) - \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right) \gamma_{i,r,s_r}(t) \right] \tag{6.12}$$

Again, if a lattice instead of an N-best list is provided for discriminative learning, a huge number of terms in the summation over s_r in (6.11) would be encountered. To keep the computation manageable, one needs to approximate the computation in (6.11), which we describe below.

6.2 COMPUTING $\Delta\gamma(i,r,t)$ USING LATTICES

A lattice, as illustrated in Figure 6.1, is a compact representation of a large list of strings. It is an acyclic directed graph consisting of a number of nodes (nine in Figure 6.1 as a highly simplified example) and a set of directed arcs each connecting two nodes. In Figure 6.1, each node corresponds to a time stamp and each arc corresponds to a substring unit (e.g., a word of a phone in a sentence). A string in the lattice contains multiple arcs. A typical arc is shown as q in Figure 6.1. Two time stamps, b_q and e_q, are associated with each arc, providing an estimate of the segment boundaries for the substring. For a time slice t within the arc segment q, we have $b_q \le t \le e_q$.

We will show below that (6.5) and (6.11) can both be computed efficiently by a forward–backward algorithm. First, given the lattice in Figure 6.1 and s_r as an arbitrary path in that lattice, we will show the occupancy given the entire string s_r can be computed as the occupancy given the local arc q, where arc q belongs to s_r. that is,

$$\gamma_{i,r,q}(t) = \gamma_{i,r,s_r}(t) \text{ when } b_q \le t \le e_q \tag{6.13}$$

To see this, let s_r be composed of three substrings: s'_r, q, s'', and correspondingly the observation sequence X_r is composed of three subsequences: X''_r, X_q, X''. Then the right-hand side of (6.13) can be analyzed as

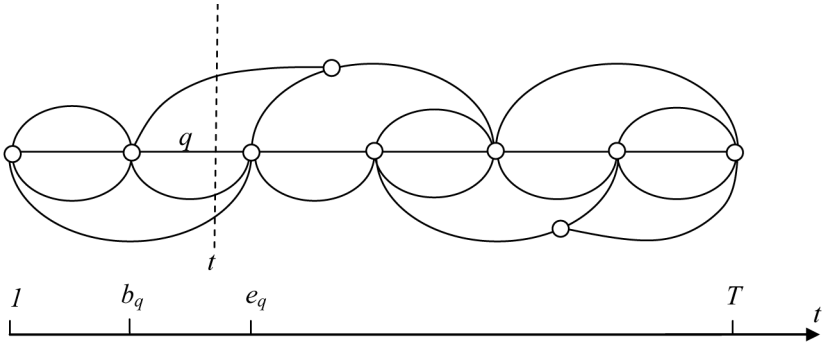

FIGURE 6.1: A graphical illustration of a lattice, where q represents an arc in the lattice and t represents a time slice. The time span of arc q is $b_q \le t \le e_q$ and that for the entire lattice is $1 \le t \le T$. In this simple example, the total number of arcs (q) is 21, which is substantially lower than the total number of paths (s_r) of (4.20). The essence of the decomposition of occupation probability introduced in the text (Eq. (6.25)) is to enable fast computation by reducing the number of terms in summation over s_r to that over q.

$$\begin{aligned}\gamma_{i,r,s_r}(t{:}\, b_q \;\le\; t \;\le\; e_q) &= p\left(q_{r,t:b_q \le t \le e_q} = i \middle| X_r, s_r, \Lambda'\right) \\ &= p\left(q_{r,t:b_q \le t \le e_q} = i \middle| X'_r, X_q, X''_r, s'_r, q, s''_r, \Lambda'\right) \\ &= p\left(q_{r,t:b_q \le t \le e_q} = i \middle| X_q, q, \Lambda'\right) \\ &= \gamma_{i,r,q}(t{:}\, b_q \le t \le e_q)\end{aligned}$$

which is the left-hand side of (6.13). The third step holds because the HMM of s_r is formed by concatenating phone-specific HMMs, so that the states in different arcs belong to different HMMs, and are independent of each other, that is, given arc q, its first HMM state $q_{r,b}$ is independent of its preceding state q_{r,b_q-1}.

The essence of (6.13) is to decouple the dependency on the local arc q from the entire string s_r. This enables drastic simplification of the computation in (6.5) and (6.11), which we discuss below for three separate cases.

6.2.1 Computing $\Delta\gamma(i,r,t)$ for MMI Involving Lattices

The principal computation burden in (6.5) is the huge number (N) of summation terms for s_r for the equivalent N-best list of a lattice in the following quantity in (6.5):

$$\Upsilon = \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right) \gamma_{i,r,s_r}(t) \tag{6.14}$$

Using (6.13), we can significantly reduce the computation by the following simplification:

$$\begin{aligned}\Upsilon &= \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right) \gamma_{i,r,q}(t) = \sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot \sum_{s_r : q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right) \\ &= \sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot p\left(q \middle| X_r, \Lambda'\right) = \sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot \frac{p\left(q, X_r \middle| \Lambda'\right)}{p\left(X_r \middle| \Lambda'\right)}\end{aligned} \tag{6.15}$$

Note that the number of summation terms for q in (6.15) after the approximation is substantially smaller than that for s_r before the approximation. The key quantities in (6.15) can be efficiently computed as follows (proof omitted):

$$p(q, X_r \middle| \Lambda') = \alpha(q)\beta(q) \tag{6.16}$$

$$p\left(X_r \middle| \Lambda'\right) = \sum_{q:q\in\{\text{ending arcs}\}} p\left(q, X_r \middle| \Lambda'\right) = \sum_{q:q\in\{\text{ending arcs}\}} \alpha(q) \tag{6.17}$$

where the "forward" and "backward" probabilities are defined by

$$\alpha(q) \triangleq p\left(q, X'_r(q), X_r(q) \middle| \Lambda'\right) \tag{6.18}$$

$$\beta(q) \triangleq p\left(X''_r(q) \middle| q, \Lambda'\right) \tag{6.19}$$

In (6.18), $X'_r(q)$ denotes the rth training token's partial observation sequence preceding arc q, that is, during $1 \le t < b_q$. $X_r(q)$ is the observation sequence bounded by arc q with $b_q \le t \le e_q$. $X''_r(q)$ in (6.19) denotes the partial observation sequence succeeding arc q, or during $e_q < t \le T_r$. $\alpha(q)$ is the probability that lattice is at arc q during time $b_q \le t \le e_q$, and having generated partial observation $X'_r(q)$ plus $X_r(q)$, that is, $x_{r,1}, \ldots, x_{r,\text{eq}}$. $\beta(q)$ is the probability of generating partial observation $X''_r(q)$ given that the lattice is at arc q at time $t = e_q$.

For each arc q in the lattice, $\alpha(q)$ and $\beta(q)$ can be computed by the following efficient forward and backward recursions, respectively (proofs omitted):

$$\alpha(q) = \sum_{\{p:\, p \text{ precedes } q\}} P\left(q \middle| p, \Lambda'\right) p\left(X_r(q) \middle| p, \Lambda'\right) \alpha(p) \tag{6.20}$$

and

$$\beta(q) = \sum_{\{v:\, v \text{ succeeds } q\}} P\left(v \middle| q, \Lambda'\right) p\left(X_r(v) \middle| v, \Lambda'\right) \beta(v) \tag{6.21}$$

where in (6.20), $\{p: p \text{ precedes } q\}$ is the collection of all arcs p that directly connects to q in the lattice. Similarly, $\{v: v \text{ succeeds } q\}$ in (6.21) is the collection of all arcs v that directly connect to q in the lattice. $\alpha(q)$ is initialized at the starting arc q_0 by $\alpha(q_0) = \pi(q_0)p(X_r(q_0)|q_0, \Lambda')$, and $\beta(q)$ initialized at the ending arc q_E by $\beta(q_E) = 1$.

The recursive computation of $\alpha(q)$ and $\beta(q)$ is illustrated in Figure 6.2. There is a direct analogy between this forward and backward probability computation over the sublattice illustrated here and that for the standard HMM over time [10, 43]. In Figure 6.2, the arc q under consideration is analogous to the HMM state occupied at current time frame t in describing the HMM's forward–backward algorithm, the set of arcs $\{p: p \text{ precedes } q\}$ is analogous to all states in HMM at frame $t-1$, the set $\{v: v \text{ succeeds } q\}$ is analogous to all states in HMM at frame $t+1$. $X_r'(q)$ plays the role of the sequence of observation vectors from 1 to $t-1$, and $X_r''(q)$ plays the role of the sequence of observation vectors from $t+1$ to the end. $P(q|p,\Lambda')$ is analogous to the HMM's transition probability (and its value is available from the lattice as the phone or word's "bigram language model" score). $p(X_r(q)|q,\Lambda')$ is analogous to the HMM's emission probability (and its value is available from the lattice as the "acoustic model" score for arc q). Given these analogies, the forward and backward probability computation for (6.20) and (6.21) as illustrated in Figure 6.2 becomes identical to that for the standard HMM (as illustrated in Figures 6.5 and 6.6 of Ref. [43]).

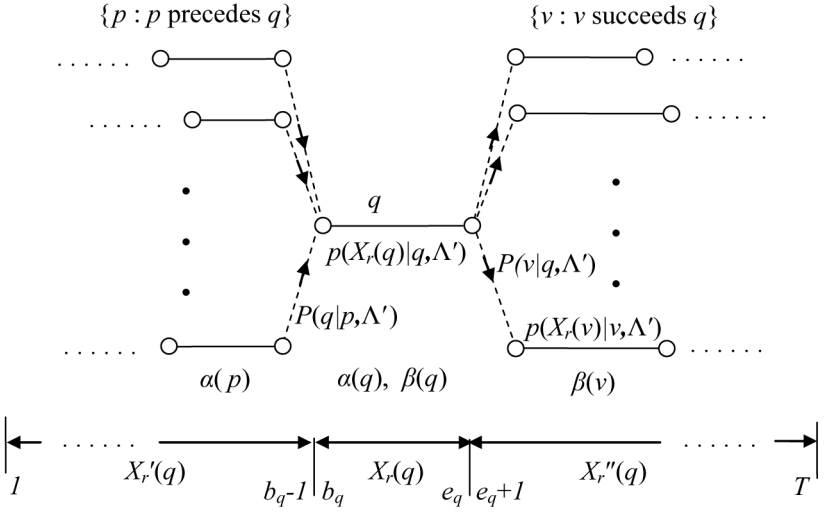

FIGURE 6.2: Illustrations of the sublattice that contains arc q and of the computation of the forward and backward $\alpha(q)$ and $\beta(q)$ based on the sublattice. Each solid line represents an arc in the lattice, and each dashed line represents the direct connection between two arcs (i.e., $b_q - 1 = e_p$).

6.2.2 Computing $\Delta\gamma(i,r,t)$ for MPE/MWE Involving Lattices

We now describe how the computation burden in (6.11) due to the huge number of summation terms over string s_r can be drastically reduced for the MPE/MWE case. It should be pointed out that (6.11) is a unified form for both MCE and MPE/MWE. However, due to the different properties of $C(s_r)$ (i.e., MCE has each term as the Kronecker delta function, but not so for MPE/MWE), the lattice-based computation of (6.11) for MCE and MPE/MWE becomes different.

Consider a particular string token s_r that consists of a sequence of subtokens or substrings. For MCE, $C(s_r) = \delta(s_r, S_r)$, and hence if any of the subtokens is incorrect, the entire token is incorrect also. On the other hand, for MPE, $C(s_r) = A(s_r, S_r)$, which is the raw phone (substring) accuracy count in the sentence string s_r. Therefore, we have a sum of raw phone (substring) accuracy counts of all subtokens; that is, for $s_r = s_{r,1}, \ldots, s_{r,N_r}$, we have $C(s_r) = \sum_{i=1}^{N} C(s_{r,i})$, where $C(s_{r,i})$ is the raw accuracy count of the subtoken $s_{r,i}$. Readers are referred to [40] for the computation of $C(s_{r,i})$ for subtoken $s_{r,i}$ in the lattice.

In this section, we discuss the lattice-based computation of (6.5) for MPE/MWE. (The lattice-based MCE will be discussed in the next section.)

To proceed, we define

$$\bar{C}_r \triangleq \frac{\sum_{s_r} p\left(X_r, s_r \middle| \Lambda'\right) C(s_r)}{\sum_{s_r} p\left(X_r, s_r \middle| \Lambda'\right)} \tag{6.22}$$

which is the average accuracy count of utterance r, given the observation sequence (X_r) and the lattice that represents all possible strings s_r.

Then, we make use of (6.13) to simplify (6.11) as follows:

$$\begin{aligned}
\Delta\gamma(i,r,t) &= \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right)\left(C(s_r) - \bar{C}_r\right)\gamma_{i,r,s_r}(t) \\
&= \sum_{q:\, t\in[b_q,e_q]} \sum_{s_r:\, q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right)\left(C(s_r) - \bar{C}_r\right)\gamma_{i,r,q}(t) \\
&= \sum_{q:\, t\in[b_q,e_q]} \gamma_{i,r,q}(t)\cdot\left[\sum_{s_r:\, q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right) C(s_r) - \bar{C}_r \cdot \sum_{s_r:\, q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right)\right] \\
&= \sum_{q:\, t\in[b_q,e_q]} \gamma_{i,r,q}(t)\cdot\left[p\left(q \middle| X_r, \Lambda'\right)\cdot\frac{\sum_{s_r:\, q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right) C(s_r)}{p\left(q \middle| X_r, \Lambda'\right)} - \bar{C}_r \cdot p\left(q \middle| X_r, \Lambda'\right)\right] \\
&= \sum_{q:\, t\in[b_q,e_q]} \gamma_{i,r,q}(t)\cdot p\left(q \middle| X_r, \Lambda'\right)\cdot\left[\bar{C}_r(q) - \bar{C}_r\right]
\end{aligned} \tag{6.23}$$

where $p(q|X_r, \Lambda') = \sum_{s_r: q \in s_r} p(s_r | X_r, \Lambda') = \dfrac{p(q|X_r, \Lambda')}{p(X_r, \Lambda')}$ is computed in the same way as for (6.16) and (6.17). In (6.23), we define

$$\bar{C}_r(q) = \frac{\sum_{s_r: q \in s_r} p\left(s_r | X_r, \Lambda'\right) C(s_r)}{p\left(q | X_r, \Lambda'\right)} = \frac{\sum_{s_r: q \in s_r} p\left(s_r, X_r | \Lambda'\right) C(s_r)}{\sum_{v_r: q \in v_r} p\left(v_r, X_r | \Lambda'\right)} \tag{6.24}$$

which is the average accuracy count of the utterance r, given observation sequence X_r and the sublattice that represents all strings s_r containing arc q.

The difficulty of computing $\bar{C}_r(q)$ and $\bar{C}_r$ in (6.23) lies in the very large number of terms in the summation over s_r: $q \in s_r$ and over s_r, respectively. To efficiently compute $\bar{C}_r(q)$ and $\bar{C}_r$, we now further define the following two additional "forward" and "backward" variables for each arc q (following [40]):

$$\varphi(q) \triangleq \frac{\sum_{\{s': s' \text{ precedes } q\}} p\left(s', q, X'_r(q), X_r(q) | \Lambda'\right) C(s', q)}{\sum_{\{s': s' \text{ precedes } q\}} p\left(s', q, X'_r(q), X_r(q) | \Lambda'\right)} \tag{6.25}$$

and

$$\psi(q) \triangleq \frac{\sum_{\{s'': s'' \text{ succeeds } q\}} p\left(s'', X''_r(q) | q, \Lambda'\right) C(s'')}{\sum_{\{s'': s'' \text{ succeeds } q\}} p\left(s'', X''_r(q) | q, \Lambda'\right)} \tag{6.26}$$

In (6.25), $\varphi(q)$ is the weighted average accuracy count of the sublattice that represents all partial paths (s', q) ending inclusively in q, with the partial observation sequence $X'_r(q) \cup X_r(q)$ (i.e., $x_{r,1}$, …, $x_{r,\text{eq}}$). In (6.26), $\psi(q)$ is the weighted average accuracy count of the sublattice that represents all partial paths s'' that succeeds q, with the partial observation sequence $X'_r(q)$. Figure 6.3 illustrates the sublattice that represents all s_r that contains arc q, together with all the relevant quantities for defining $\varphi(q)$ and $\psi(q)$ based on the sublattice. To show these quantities in defining $\varphi(q)$, we denote the accuracy count as $C(s', q)$ for a given partial path (s', q) encircled by the dotted line to the left of Figure 6.3. We denote the weight associated with this partial path as $p(s', q, X'_r(q), X_r(q)|\Lambda')$. The relevant quantities defining $\psi(q)$ are illustrated to the right of Figure 6.3, including the partial path (s'') that is to the future of arc q, the accuracy count $C(s'')$ associated with this path, and the associated weight of $p(s'', X''_r(q)|\Lambda')$.

We now describe the computation of $\varphi(q)$ as defined in (6.25) and $\psi(q)$ defined in (6.26) efficiently for each arc q in the lattice. For $\varphi(q)$, we use the following efficient "forward" recursion (proof omitted):

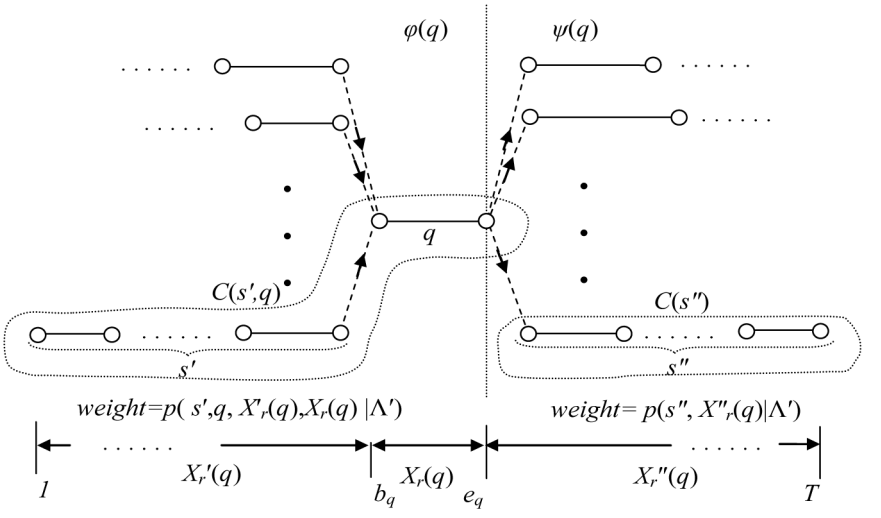

FIGURE 6.3: Illustrations of the sublattice that contains arc q, and of the probability weights that define $\varphi(q)$ of (6.25) and $\psi(q)$ of (6.26) based on the sublattice.

$$\varphi(q) = \frac{\sum_{\{p:p \text{ precedes } q\}} P(q|p,\Lambda')\, p(X_r(q)|p,\Lambda')\, \alpha(p)[\varphi(p) + C(q)]}{\sum_{\{p:p \text{ precedes } q\}} P(q|p,\Lambda')\, p(X_r(q)|p,\Lambda')\, \alpha(p)}$$
$$= \frac{\sum_{\{p:p \text{ precedes } q\}} P(q|p,\Lambda')\, \alpha(p)\varphi(p)}{\sum_{\{p:p \text{ precedes } q\}} P(q|p,\Lambda')\, \alpha(p)} + C(q) \qquad (6.27)$$

where $\varphi(q)$ is initialized for each starting arc q_0 by $\varphi(q_0) = C(q_0)$, which is the raw phone or word accuracy for q_0. For $\psi(q)$, we use the following efficient "backward" recursion (proof omitted):

$$\psi(q) = \frac{\sum_{\{v:v \text{ succeeds } q\}} p(X_r(v)|v,\Lambda')\, P(v|q,\Lambda')\, \beta(v)\,[C(v) + \psi(v)]}{\sum_{\{v:v \text{ succeeds } q\}} p(X_r(v)|v,\Lambda')\, P(v|q,\Lambda')\, \beta(v)} \qquad (6.28)$$

where $\psi(q)$ is initialized for each ending arc q_E by $\varphi(q_E) = 0$.

The recursive computation of $\varphi(q)$ in (6.27) is illustrated in Figure 6.4. Given the partial observation sequence $x_{r,1}, \ldots, x_{r,\text{eq}}$, $[\varphi(p) + C(q)]$ is the mean accuracy count of the sublattice that represents all partial paths that pass p and end with q. These paths are marked by the dotted line

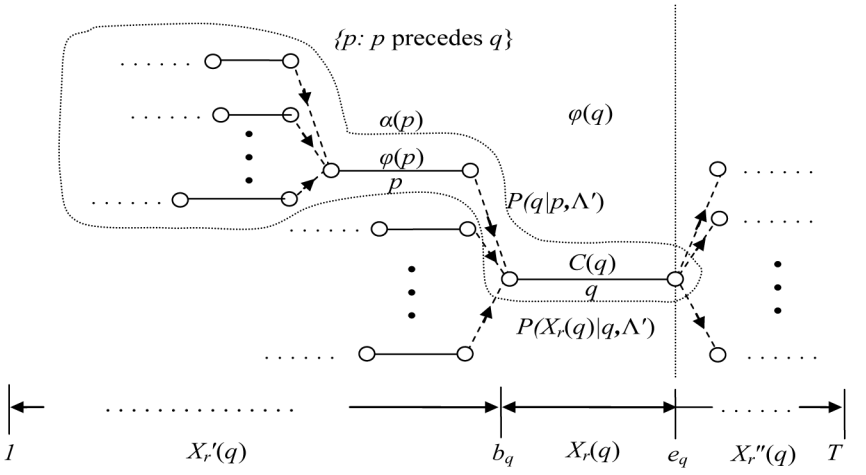

FIGURE 6.4: Illustrations of the sublattice containing arc q and of the recursive $\varphi(q)$ computation based on the sublattice. Each solid line represents an arc in the sublattice, and each dashed line represents the transition between two arcs. The dotted line encircles all partial paths that pass p and end with q.

in Figure 6.4. $\varphi(q)$ is a weighted sum and the weighted associated with each path passing arc p is $\alpha(p)P(q\,|\,p, \Lambda')p(X_r(q)|\,p, \Lambda')$, where each of the three factors is associated with each corresponding portion that makes up the path. The three factors are placed in the corresponding portions on the path in Figure 6.4. The weighted average of $[\varphi(p) + C(q)]$ over all arcs p (directly preceding q) using the three-factor weight above gives the recursive form of $\varphi(q)$ shown in the first line of (6.27). The second line of (6.27) removes some redundant computation and has been implemented in practice.

The recursive computation of $\psi(q)$ in (6.28) can be similarly interpreted as the weighted average of the accuracy count $C(v) + \psi(v)$ for all arcs v directly following q.

Now given that both $\varphi(q)$ and $\psi(q)$ are computed, and assuming that arc q depends only on the arcs directly preceding it and succeeding it, we can use (6.25) and (6.26) to directly prove that

$$\bar{C}_r(q) = \varphi(q) + \psi(q) \tag{6.29}$$

as one of the two quantities required to compute $\Delta\gamma(i, r, t)$ in (6.23). The interpretation of (6.29) is offered by using Figure 6.3. By definition, $\bar{C}_r(q)$ is the average accuracy count for utterance r over the sublattice shown in Figure 6.3 that contains arc q. This count can be decomposed into two parts. The first part is the "forward" average accuracy count of the left part of the sublattice in Figure 6.3 for the utterance from $t = 1$ to e_q, which is $\varphi(q)$. The second part is the "backward" average accuracy count of the right part of the sublattice for the utterance from $t = e_q + 1$ to T_r, which is $\psi(q)$.

The second quantity, $\bar{C}_r$, required to compute $\Delta\gamma(i, r, t)$ in (6.23) can be proved to be

$$\bar{C}_r = \frac{\sum_{q:q\in\{\text{ending arcs}\}} \varphi(q)\alpha(q)}{\sum_{q:q\in\{\text{ending arcs}\}} \alpha(q)} \tag{6.30}$$

The interpretation of (6.30) is as follows. Let arc q be an ending arc in the lattice. And recall that $\varphi(q)$ is the average accuracy count of utterance r given the sublattice that represents all s_r containing (sublattice-ending) arc q, and $\varphi(q)$ is the weight of this sublattice. Therefore, $\bar{C}_r$, which is defined in (6.22) as the average accuracy count for the entire lattice, becomes a weighted sum of the average accuracy counts of all sublattices as shown in (6.30).

This completes the description of the computation of $\Delta\gamma(i, r, t)$ in (6.23).

6.2.3 Computing $\Delta\gamma(i, r, t)$ for MCE Involving Lattices

Finally, we discuss using lattice approximation (6.13) to compute $\Delta\gamma(i, r, t)$ of (6.11) for MCE. As we mentioned earlier, whereas (6.13) is unified between MPE and MCE, the specific form of $C(s_r) = \delta(s_r, S_r)$ in MCE permits special simplification of $\Delta\gamma(i, r, t)$ of (6.11) for MCE. The simplification steps, followed by the use of (6.13), lead to

$$\begin{aligned}
\Delta\gamma(i,r,t) &= \sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right)\left[C(s_r) - p\left(S_r \middle| X_r, \Lambda'\right)\right]\gamma_{i,r,s_r}(t) \\
&= p\left(S_r \middle| X_r, \Lambda'\right)\gamma_{i,r,S_r}(t) - p\left(S_r \middle| X_r, \Lambda'\right)\left[\sum_{s_r} p\left(s_r \middle| X_r, \Lambda'\right)\gamma_{i,r,s_r}(t)\right] \\
&= p\left(S_r \middle| X_r, \Lambda'\right)\gamma_{i,r,S_r}(t) - p\left(S_r \middle| X_r, \Lambda'\right)\left[\sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot \sum_{s_r:q\in s_r} p\left(s_r \middle| X_r, \Lambda'\right)\right] \\
&= p\left(S_r \middle| X_r, \Lambda'\right)\gamma_{i,r,S_r}(t) - p\left(S_r \middle| X_r, \Lambda'\right)\left[\sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot p\left(q \middle| X_r, \Lambda'\right)\right] \\
&= p\left(S_r \middle| X_r, \Lambda'\right)\left[\underbrace{\gamma_{i,r,S_r}(t)}_{\gamma_{i,r}^{\text{num}}(t)} - \underbrace{\sum_{q:t\in[b_q,e_q]} \gamma_{i,r,q}(t) \cdot p\left(q \middle| X_r, \Lambda'\right)}_{\gamma_{i,r}^{\text{den}}(t)}\right]
\end{aligned} \tag{6.31}$$

The last line shows striking similarity between lattice-based MCE and MMI. In (6.31), $p(q\,|X_r, \Lambda') = \dfrac{p(q|X_r, \Lambda')}{p(X_r, \Lambda')}$ is computed by (6.16) and (6.17) for the numerator and denominator, respectively. Also in (6.31), we have $p(S_r\,|X_r, \Lambda') = \dfrac{p(X_r|X_r, \Lambda')p(S_r|\Lambda')}{p(X_r, \Lambda')}$, where correct string S_r is known. Hence, $\gamma_{i,r,S_r}(t)$ and $p(X_r|S_r, \Lambda')$ in (6.31) can be efficiently computed by the standard forward–backward algorithm for the HMM [43]. Finally, for the computation of $p(S_r|\,\Lambda')$ and $p(X_r\,|\,\Lambda')$, we use the language model and $\sum_{q:q\in\{\text{ending arcs}\}}\alpha(q)$, respectively.

Note that the computation for the lattice-based MCE we provided in (6.31) does not require removing the correct word string S_r from the lattice.

6.3 ARBITRARY EXPONENT SCALING IN MCE IMPLEMENTATION

In this section, we discuss one of the two empirical issues in MCE implementation that were raised in Chapter 3. In (3.15), if we use the exponent scaling factor $\eta \neq 1$, we can obtain the following result corresponding to (3.17):

$$l_r\left(d_r(X_r,\Lambda)\right) = \frac{\sum\limits_{s_r, s_r\neq S_r} p^{\eta}\left(X_r, s_r|\Lambda\right)}{\sum\limits_{s_r} p^{\eta}\left(X_r, s_r|\Lambda\right)}$$

The corresponding result to (3.19) then becomes

$$O_{\text{MCE}}(\Lambda) = \sum_{r=1}^{R} \frac{p^{\eta}\left(X_r, S_r|\Lambda\right)}{\sum\limits_{s_r} p^{\eta}\left(X_r, s_r|\Lambda\right)}$$

which can be reformulated into a rational function using the same steps as in Section 3.4.2:

$$O_{\text{MCE}}(\Lambda) = \frac{\sum_{s_1,\ldots,s_R} p^{\eta}\left(X_1,\ldots,X_R, s_1,\ldots,s_R|\Lambda\right) C_{\text{MCE}}(s_1,\ldots,s_R)}{\sum\limits_{s_1,\ldots,s_R} p^{\eta}\left(X_1,\ldots,X_R, s_1,\ldots,s_R|\Lambda\right)} \qquad (6.32)$$

The remaining derivations in Chapters 4 and 5 will no longer follow strictly for the more general and practical case of (6.32). In the MCE implementation that we have done, however, we modify (6.11) for computing $\Delta\gamma(i, r, t)$ in the following manner in order to include the effects of the exponent scaling factor:

$$\Delta\gamma(i,r,t) = \sum_{s_r}\tilde{p}\left(s_r\left|X_r,\Lambda'\right.\right)\left(C(s_r) - \sum_{s_r}\tilde{p}\left(X_r\left|s_r,\Lambda'\right.\right)C(s_r)\right)\gamma_{i,r,s_r}(t) \qquad (6.33)$$

where $\tilde{p}(s_r|X_r, \Lambda')$ is the generalized posterior probability of s_r, which can be computed as

$$\tilde{p}\left(s_r|X_r,\Lambda'\right) = \frac{p^{\eta}\left(X_r,s_r|\Lambda'\right)}{\sum_{s_r} p^{\eta}\left(X_r,s_r|\Lambda'\right)} \tag{6.34}$$

After this modification, all derivations in Chapters 4 and 5 are unchanged.

6.4 ARBITRARY SLOPE IN DEFINING MCE COST FUNCTION

The second empirical MCE implementation issue raised in Chapter 3 concerns the use of $\alpha \neq 1$ in (3.16). For 1-best MCE, α acts as η; that is, we can equivalently set $\eta = \alpha$, and $\alpha = 1$. Then, we can compute $\Delta\gamma(i, r, t)$ according to (6.33). For N-best MCE ($N > 1$), given the discriminant function defined in (3.15) and sigmoid function defined in (3.16), we have the following result corresponding to (3.17):

$$l_r\left(d_r(X_r,\Lambda)\right) = \frac{\left(\sum_{s_r,\, s_r \neq S_r} p^{\eta}\left(X_r,s_r|\Lambda\right)\right)^{\alpha}}{p^{\eta\cdot\alpha}\left(X_r,S_r|\Lambda\right) + \left(\sum_{s_r,\, s_r \neq S_r} p^{\eta}\left(X_r,s_r|\Lambda\right)\right)^{\alpha}} \tag{6.35}$$

Now, α is applied outside of the summation of scaled joint probabilities over all competing strings, making rigorous computation intractable. In our practical MCE implementation, we instead use $\sum_{s_r,\, s_r \neq S_r} p^{\alpha\cdot\eta}(X_r, s_r|\Lambda)$ to approximate $\left(\sum_{s_r,\, s_r \neq S_r} p^{\eta}(X_r, s_r|\Lambda)\right)^{\alpha}$. This approximation (which is exact when η approaches infinity) makes it equivalent to setting the new "η" as $\alpha \cdot \eta$, and setting the new $\alpha = 1$. Then, again, we can compute $\Delta\gamma(i, r, t)$ according to (6.33). It should be noted that, with this approximation, the computation for the lattice-based MCE we provided in (6.31) does not require removing the correct word string S_r from the lattice, as shown in (6.31) (Section 6.2.3). This contrasts the solution in [31, 46] where the removal was necessary without using the approximation, making it more difficult to implement in practice.

The two empirical solutions cited above have been successfully implemented in our speech recognition system, yielding strong practical results (published in [20, 58]) that validate the solutions.

• • • •

Author Query Form

(Queries are to be answered by the Author)

He – Chapter 6

The following queries have arisen during the typesetting of your manuscript. Please answer these queries.

Query Marker	Query	Reply
Q1	"420" was tagged as (4.20). Please check if this correct.	

Thank you very much.