Chapter 3

Hierarchical Training Methods

Training a maximum entropy model is usually a computationally intensive task. This drawback of the maximum entropy approach has prevented many people from using this method. In this chapter, we will describe some efficient training methods for maximum entropy models. Preliminary results have been introduced in Wu & Khudanpur (2000b) and Wu & Khudanpur (2002). We will present more detailed results in this chapter.

3.1 Introduction

Training an ME model using either generalized iterative scaling or its improved version described by Della Pietra, Della Pietra & Lafferty (1997) needs several iterations for converging. The running time is proportional to both the number of iterations and the time for each iteration. The training procedure inside each iteration can be separated into three phases: computing normalization factors z for all the histories seen in training, calculating feature expectations $p[g]$ for each constraint g and updating model parameters α. We study the first two phases of training in this chapter, and the last phase has been discussed in Section 2.5.3. The first two phases are the same for both GIS and IIS. The only difference lies in the updating of model parameters, which takes only a small amount of computation in each iteration. Therefore, the total training time is predominantly the time for computing z and $p[g]$.

The crucial idea of training maximum entropy models efficiently is to take advantage of hierarchical structures among features, i.e., to share the computation corresponding to lower order features among high order features nesting them. The computation of z's and that of $p[g]$'s take the same amount of time, as we have shown in the previous chapter, and any improvement made for the former applies to the latter (and vice versa). We therefore focus on improving the computation of normalization factors first, and then transfer the resulting efficient methods for computing z to the computation of feature expectations.

In order to address the training methods for maximum entropy models, we need to introduce the relations between features and the feature hierarchy of maximum entropy models. It will later be shown that the efficiency of training an ME model mainly depends on the extent of computation sharing permitted by the hierarchical structure of the model.

By taking advantage of the hierarchical structure of N-gram features, training N-gram models (and models that look like N-gram models) can be as fast per iteration as training the corresponding back-off models, which is the time for scanning the training data once. However, the major motivation of using the ME method is not to build N-gram models, but rather models with some kinds of non-local dependencies. Unfortunately, training these models cannot take advantage of nested features directly and thus it is computationally intensive. However, the ideas of sharing computation among features can be ported from the training of N-gram models to these kinds of models. There are some nested features (although not all features are nested) in these models, and the computation related to these nested features can still be simplified. As a result, the training time can be made much shorter than that of the state-of-the-art unigram-caching method introduced in the previous chapter, even for more complex models.

For specific kinds of non-N-gram models such as the topic-dependent model, a divide-and-conquer strategy is designed to train them almost as efficiently as training back-off models.

3.1.1 Lower Bound and Upper Bound on Complexity

Training a maximum entropy model per iteration needs at least the time $O(L)$ of scanning the training data once. Even the empirical estimation needs $O(L)$ time. If a training algorithm reaches this lower bound, then it is already optimal.

On the other hand, if the probability of histories is approximated by their relative frequency in the training data, a maximum entropy model needs no more than $O(L \cdot |V|)$ time where $|V|$ is the vocabulary size. Therefore, the upper bound of training any ME model is $O(L \cdot |V|)$. It should be noted that $O(L \cdot |V|) \neq O(L)$ because $|V|$ is actually a function of L and cannot be simply treated as a constant[1]. Using unigram-caching will reduce the training time to $O(|\hat{X}| \cdot \bar{C})$ where $|\hat{X}|$ is the number of different history classes seen in the training data and $\bar{C}$, the average number of words with some conditional features activated for each history class, is also a function of $|V|$ and L. Usually $|\hat{X}|$ is several times smaller than L and $\bar{C}$ is 10-100 times smaller than $|V|$ in language modeling.

In some applications such as part-of-speech tagging and syntactic parsing, $O(|V|)$ is only of the order of 100 and independent of L. A complexity of $O(|\hat{X}| \cdot \bar{C})$ or $O(L \cdot |V|)$ is acceptable in such applications. In language modeling, however, the vocabulary size is more than tens of thousands. Consequently, a time of $O(|\hat{X}| \cdot \bar{C})$ becomes intractable and we are seeking training methods with $O(|\hat{X}| \cdot c)$ complexity where c is significantly smaller than $\bar{C}$.

The remainder of this chapter is organized as follows. We start with an example of training a trigram model and then derive the hierarchical training method for N-gram models in Section 3.2. In Section 3.3, we define three relations between features: nested, overlapping and non-overlapping and describe the feature hierarchy of an ME model by a digraph, in which nodes are feature patterns and arcs are feature relations. The extension of hierarchical training method for the model with non-nested features is described in Section 3.4. The generalized version of the hierarchical method for ME models and the divide-and-conquer strategy for topic models in particular follow

[1] Even if $|V|$ were to be treated as a constant, it is typically $10^3 - 10^4$ and hence not negligible in practice even if it were to be negligible in the $O(\cdot)$ sense.

in Section 3.5 and Section 3.6, respectively. We only describe the computation of the normalization factor z in these sections. The computation of the expectation of feature functions is briefly discussed in Section 3.7. In Section 3.8, we provide some details of training specific language models studied in this dissertation. The efficiency of training methods is evaluated in Section 3.9 based on two tasks, Switchboard and Broadcast News.

3.2 Hierarchical Training Method for Maximum Entropy Models with only Nested Features

In an N-gram model

$$p(w_i|w_{i-N+1},\cdots,w_{i-1}) = \frac{1}{z}\prod_{n=1}^{N}\alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)}$$

where

$$g(w_{i-n+1},\cdots,w_i) = \begin{cases} 1 & w_{i-n+1},\cdots,w_i \text{ has an n-gram constraint,} \\ 0 & \text{otherwise} \end{cases}$$

are feature functions and $\alpha_{w_{i-n+1},\cdots,w_i}$ are corresponding parameters. If a word w_i has an order n feature $g(w_{i-n+1},\cdots,w_i)$ active in the given history $w_{i-N+1},\cdots,w_{i-1}$, it also usually has an order $n-1$ feature $g(w_{i-n+2},\cdots,w_i)$ activated (for $2 \leq n < N$). Therefore, $g(w_{i-n+1},\cdots,w_i)$ is nested in a feature $g(w_{i-n+2},\cdots,w_i)$ for $2 \leq n \leq N$. Figure 3.1 shows the sets of words Y_1, Y_2, $\cdots$, Y_N with unigram, bigram,$\cdots$, and N-gram features activated respectively for a given history $w_{i-N+1},\cdots,w_i$.

We will show in this section that this kind of model can be trained hierarchically, and training such a model for one iteration takes the same amount of time as the calculation of empirical expectations. We demonstrate the hierarchical training method by a trigram model and then extend this method to any model with only nested features.

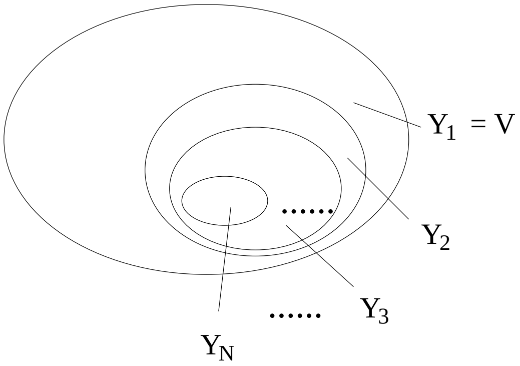


Figure 3.1: Word sets with unigram, bigram, $\cdots$, N-gram features activated.

3.2.1 Computing Normalization Factors in a Trigram Model

Recall the trigram model

$$p(w_i|w_{i-2}, w_{i-1}) = \frac{\alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)}}{z(w_{i-2}, w_{i-1})} \tag{3.1}$$

where

$$z(w_{i-2}, w_{i-1}) = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)}.$$

Subscripts of α's here indicate the indices of corresponding features in the feature set. The order of features g is indicated by the number of coefficients of g and may be omitted without causing any confusion. The normalization factor z can be computed as

$$z(w_{i-2}, w_{i-1}) \;=\; \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} + \sum_{w_i \in Y_x} [\alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} - 1] \cdot \alpha_{w_i}^{g(w_i)} \tag{3.2}$$

as described in Chapter 13, page 226, in Jelinek (1997). According to the analysis in Section 2.5.1, the second term in Equation (3.2)

$$\sum_{w_i \in Y_x} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1) \cdot \alpha_{w_i}^{g(w_i)} \tag{3.3}$$

with complexity of $O(\sum_{x \in \hat{X}} |Y_x|)$ dominates the computation. In the future, the short-hand notation $O(|\hat{X}| \cdot \bar{C})$ will be used for $O(\sum_{x \in \hat{X}} |Y_x|)$ for simplicity, where $\bar{C} = \frac{1}{|\hat{X}|} \sum_{x \in \hat{X}} |Y_x|$.

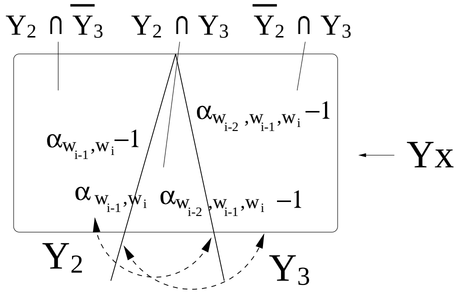
Most of the words in Y_x have only bigram features active. For example, in any given history x, the set of words with bigram features contains on average roughly 300 words and 6000 words respectively in Switchboard and Broadcast News, among which only 2 or 3 (on average) have trigram or higher order constraints. We take advantage of this fact and split the word set Y_x into Y_2 and $Y_x - Y_2$ where

$$Y_2 = \{w_i : g(w_{i-1}, w_i) = 1 \text{ for some bigram feature}\}$$

and also into Y_3 and $Y_x - Y_3$ where

$$Y_3 = \{w_i : g(w_{i-2}, w_{i-1}, w_i) = 1 \text{ for some trigram feature}\}.$$

Note that Y_2 and Y_3 are history dependent even though we omit the subscripts x for the sake of simplicity. Now, since $Y_x = Y_2 \cup Y_3$ and $\overline{Y}_2 \cap Y_3 = \phi$, Y_x is split into three disjoint subsets as shown in Figure 3.2



$Y_2 \cap \overline{Y_3}$
$Y_2 \cap Y_3$
$\overline{Y_2} \cap Y_3$
$\alpha_{w_{i-1},w_i} - 1$
$\alpha_{w_{i-2},w_{i-1},w_i} - 1$
$\alpha_{w_{i-1},w_i} \alpha_{w_{i-2},w_{i-1},w_i} - 1$
Yx
Y_2
Y_3

Figure 3.2: Splitting Y_x according to Y_2 and Y_3. The values of $(\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1)$ in different cases are shown.

Obviously, for any sum over all words in Y_x,

$$\sum_{Y_x} * = \sum_{Y_2 \cap \overline{Y_3}} * + \sum_{\overline{Y_2} \cap Y_3} * + \sum_{Y_2 \cap Y_3} *.$$

Table 3.1 gives the value of $g(w_{i-1}, w_i)$ and $g(w_{i-2}, w_{i-1}, w_i)$ for words w_i in the three subsets above. Applying the values in Table 3.1 to the sum in (3.3), we have

$$\begin{aligned}
&(\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1)\alpha_{w_i}^{g(w_i)} \\
&= \begin{cases} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & w_i \in Y_2 \cap \overline{Y_3} \\ (\alpha_{w_{i-2},w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & w_i \in \overline{Y_2} \cap Y_3 \ , \\ (\alpha_{w_{i-1},w_i} \cdot \alpha_{w_{i-2},w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & w_i \in Y_2 \cap Y_3 \end{cases}
\end{aligned}$$

word set	$g(w_{i-1}, w_i)$	$g(w_{i-2}, w_{i-1}, w_i)$
$Y_2 \cap \overline{Y_3}$	1	0
$\overline{Y_2} \cap Y_3$	0	1
$Y_2 \cap Y_3$	1	1

Table 3.1: Value of $g(w_{i-1}, w_i)$ and $g(w_{i-2}, w_{i-1}, w_i)$

and therefore

$$\begin{aligned}
& \sum_{w_i \in Y_x} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1)\alpha_{w_i}^{g(w_i)} \qquad (3.4) \\
= & \sum_{w_i \in Y_2} (\alpha_{w_{i-1},w_i} - 1)\alpha_{w_i} + \sum_{w_i \in Y_3} (\alpha_{w_{i-2},w_{i-1},w_i} - 1)\alpha_{w_{i-1},w_i} \cdot \alpha_{w_i}.
\end{aligned}$$

We merge (3.2) and (3.4) into

$$\begin{aligned}
z(w_{i-2}, w_{i-1}) = & \sum_{w_i \in V} \alpha_{w_i} + \sum_{w_i \in Y_2} (\alpha_{w_{i-1},w_i} - 1)\alpha_{w_i} \qquad (3.5) \\
+ & \sum_{w_i \in Y_3} (\alpha_{w_{i-2},w_{i-1},w_i} - 1)\alpha_{w_{i-1},w_i} \cdot \alpha_{w_i}.
\end{aligned}$$

Analysis of Complexity

Now we analyze the computational complexity of implementation of the sum in (3.5).

- The first term on the right-hand size of (3.5) takes $O(U)$ time to calculate, where U is the number of unigrams in the training data.

- The sum over Y_2 in the right-hand side of the equation is fixed for all histories sharing a suffix w_{i-1} and need be computed only once for each word w_{i-1} in the vocabulary. If for a given w_{i-1}, Y_2 contains only those words following w_{i-1} in the training data, the size of Y_2 is exact the number of bigrams beginning with w_{i-1}. The number of combinations of w_{i-1} and Y_2 is exactly the number of bigrams in the training data denoted as B. The running time for the second summation over Y_2 is thus $O(B)$ for all histories.

- The sum over Y_3 depends on both w_{i-2} and w_{i-1}. Each Y_3 for a given history class w_{i-2} and w_{i-1} contains only a few words that have trigram constraints, i.e., those words following w_{i-2}, w_{i-1} in the training data. The cumulative size of all Y_3's for different w_{i-2}, w_{i-1} is thus no more than the number of seen trigrams started by w_{i-2}, w_{i-1}, denoted as T. The running time of the third summation for all histories is thus $O(T)$.

Overall, the complexity of the implementation (3.5) is therefore $O(U + B + T)$ for one iteration.

Discussion

In language modeling, training an interpolated trigram model (1.1) or a back-off trigram model (1.2) takes exactly $O(U+B+T)$ time because the empirical frequency for all unigrams, bigrams and trigrams must be collected.

Even though the value of feature functions g is binary in language modeling, the implementation (3.5) does not require g to be binary. The equation still holds when g is a real function and can be rewritten as

$$\begin{aligned} & z(w_{i-2}, w_{i-1}) \\ & \quad = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} + \sum_{w_i \in Y_2} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} - 1) \cdot \alpha_{w_i}^{g(w_i)} \\ & \quad + \sum_{w_i \in Y_3} (\alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1) \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i))} \cdot \alpha_{w_i}^{g(w_i)}. \end{aligned}$$

3.2.2 Extension for N-gram Models

The training method discussed in the previous section can be extended to N-gram models with any arbitrary value of N. Without the loss of generality, we can assume each N-gram seen in the training data corresponds to a feature function[2].

In an N-gram model,

$$p(w_i | w_{i-N+1}, \cdots, w_{i-1}) = \frac{1}{z} \prod_{j=1}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)}, \tag{3.6}$$

[2] If some N-grams are not included in the feature set, we can simply assume that they correspond to features g_k whose parameters equal $\alpha_k = 1$.

where

$$\begin{aligned} z &= z(w_{i-N+1}, \cdots, w_{i-1}) \\ &= \sum_{w_i \in V} \prod_{j=1}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)}. \end{aligned} \tag{3.7}$$

We first apply unigram-caching to (3.7) and rewrite z as

$$z = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} + \sum_{w_i \in Y_x} \left[\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_{i-1},w_i}^{g(w_{i-j+1},\cdots,w_{i-1},w_i)} - 1 \right] \alpha_{w_i}^{g(w_i)}. \tag{3.8}$$

Now we use the computational trick for the trigram model recursively to the N-gram model. We split the word set Y_x into Y_2 and $\overline{Y_2} = Y_x - Y_2$ where

$$Y_2 = \{w_i : g(w_{i-1}, w_i) = 1 \text{ for some bigram feature}\}$$

as we have done in the previous section. However, instead of defining a subset Y_3 as we have done for trigram models, we define

$$Y_3' = \{w_i : g(w_{i-n+1}, \cdots, w_i) = 1 \text{ for some n-gram features}, n \geq 3\}$$

instead. Y_x can be split into Y_3' and $\overline{Y_3'} = Y_x - Y_3'$.

Obviously, $Y_x = Y_2 \cup Y_3'$ and $\overline{Y_2} \cap \overline{Y_3'} = \phi$. Y_x is split into three disjoint subsets: $Y_2 \cap \overline{Y_3'}$, $\overline{Y_2} \cap Y_3'$ and $Y_2 \cap Y_3'$. This splitting shown in Figure 3.3 is similar to the one for training trigram models.

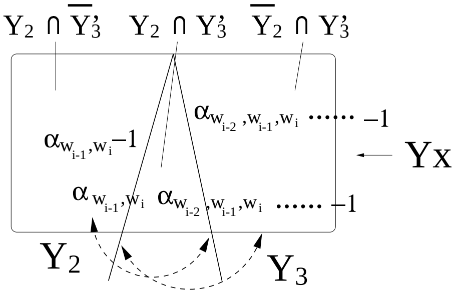

$Y_2 \cap \overline{Y_3'}$
$Y_2 \cap Y_3'$
$\overline{Y_2} \cap Y_3'$
$\alpha_{w_{i-2},w_{i-1},w_i} \cdots\cdots -1$
$\alpha_{w_{i-1},w_i} - 1$
Yx
$\alpha_{w_{i-1},w_i} \alpha_{w_{i-2},w_{i-1},w_i} \cdots\cdots -1$
Y_2
Y_3

Figure 3.3: Splitting Y_x according to Y_2 and Y_3'. The value of $\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_{i-1},w_i}^{g(w_{i-j+1},\cdots,w_{i-1},w_i)} - 1$ is shown.

Words in $Y_2 \cap \overline{Y_3'}$ have only bigram features $g(w_{i-1}, w_i)$ activated, while words in $\overline{Y}_2 \cap Y_3'$ have only order 3 or higher features $g(w_{i-2}, w_{i-1}, w_i)$, $g(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$,

$\cdots$, or $g(w_{i-N+1}, \cdots, w_i)$ activated. Only words in $Y_2 \cap Y_3'$ have both bigram features and some higher order features are activated.

By taking advantage of the fact that all feature functions g's are binary, the value of the term in the second sum in (3.8) can be rewritten as

$$\begin{aligned}
&\left[\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1\right] \cdot \alpha_{w_i}^{g(w_i)} \\
&\quad = \begin{cases} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & w_i \in Y_2 \cap \overline{Y_3'} \\ (\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1) \cdot \alpha_{w_i} & w_i \in \overline{Y_2} \cap Y_3' \\ (\alpha_{w_{i-1},w_i} \prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1) \cdot \alpha_{w_i} & w_i \in Y_2 \cap Y_3' \end{cases}, \qquad (3.9)
\end{aligned}$$

in different cases. Note that $g(w_{i-j+1}, \cdots, w_i) = 0$ is equivalent to setting $\alpha_{w_{i-j+1},\cdots,w_i} = 1$ in the equations above.

The second summation in (3.8) is thus calculated as

$$\begin{aligned}
&\sum_{w_i \in Y_x} \left[\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1\right] \alpha_{w_i}^{g(w_i)} \qquad (3.10) \\
&= \sum_{w_i \in Y_2 \cap \overline{Y_3'}} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} \\
&\quad + \sum_{w_i \in \overline{Y_2} \cap Y_3'} \left[\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1\right] \alpha_{w_i} \\
&\quad + \sum_{w_i \in Y_2 \cap Y_3'} \left[\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1\right] \alpha_{w_i} \\
&= \sum_{w_i \in Y_2 \cap \overline{Y_3'}} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} \\
&\quad + \sum_{w_i \in \overline{Y_2} \cap Y_3'} \left[\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1\right] \cdot \alpha_{w_{i-1},w_i} \cdot \alpha_{w_i}{}^{3} \\
&\quad + \sum_{w_i \in Y_2 \cap Y_3'} \left[\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - \alpha_{w_{i-1},w_i} + \alpha_{w_{i-1},w_i} - 1\right] \cdot \alpha_{w_i}
\end{aligned}$$

[3] Note: $\alpha_{w_{i-1},w_i} = 1$ here.

$$\begin{aligned} = & \sum_{w_i \in Y_2 \cap \overline{Y_3'}} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & (3.11) \\ & + \sum_{w_i \in \overline{Y_2} \cap Y_3'} \left[\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1 \right] \cdot \alpha_{w_{i-1},w_i} \cdot \alpha_{w_i} \\ & + \sum_{w_i \in Y_2 \cap Y_3'} \left[\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1 \right] \cdot \alpha_{w_{i-1},w_i} \cdot \alpha_{w_i} \\ & + \sum_{w_i \in Y_2 \cap Y_3'} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} \\ = & \sum_{w_i \in Y_2} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} & (3.12) \\ & + \sum_{w_i \in Y_3'} \left[\prod_{j=3}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1 \right] \cdot \alpha_{w_{i-1},w_i} \cdot \alpha_{w_i}. & (3.13) \end{aligned}$$

The summation (3.12) for all histories takes $O(B)$ time as we have shown in the previous section. Even if we stop here, the above implementation with the complexity of $O(B + |\hat{X}| \cdot |Y_3'|)$ is already much more efficient than that of unigram-caching $(O(|\hat{X}| \cdot |Y_x|))$ because $|Y_3'| << |Y_x|$ on average. However, we can further optimize the training algorithm. The second summation (3.13) has the same form as (3.10), and can be computed recursively by applying the same strategy.

In the remainder of this section, we formally state the hierarchical training method and prove its correctness. We begin with the following lemma that gives the recursive equation for the second summation above.

Lemma 1

In the N-gram model (3.6), $g(w_{i-n+1}, \cdots, w_i)$ for $n = 1, \cdots, N$, denotes an n-gram feature function and $\alpha_{w_{i-n+1},\cdots,w_i}$ is its corresponding parameter. Let

$$\begin{aligned} Y_n = Y_n(x) &= \{w_i : g(w_{i-n+1}, \cdots, w_i) = 1\}, \text{ and} & (3.14) \\ Y_n' = Y_n'(x) &= \{w_i : g(w_{i-k+1}, \cdots, w_i) = 1, \text{ for some } k \geq n\}, & (3.15) \end{aligned}$$

for $n = 1, 2, \cdots, N$. Then, for any given history class $w_{i-N+1}, \cdots, w_i$,

$$\begin{aligned}
&\sum_{w \in Y'_n} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
&= \sum_{w \in Y_n} (\alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)} - 1) \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
&+ \sum_{w_i \in Y'_{n+1}} \left[\prod_{j=n+1}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \prod_{j=1}^{n} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)}.
\end{aligned} \tag{3.16}$$

Proof: $Y'_n = Y_n \cup Y'_{n+1}$ and $\overline{Y_n} \cap \overline{Y'_{n+1}} = \phi$ according to the definition of Y_n and Y'_n. Y'_n can be decomposed to three disjoint subsets: $Y_n \cap Y'_{n+1}$, $\overline{Y_n} \cap Y'_{n+1}$, and $Y_n \cap \overline{Y'_{n+1}}$. Note that words in $Y_n \cap \overline{Y'_{n+1}}$ have no order $n+1$ or higher features activated, those in $\overline{Y_n} \cap Y'_{n+1}$ do not have n^{th} order features activated and only words in $Y_n \cap Y'_{n+1}$ have both activated. Therefore,

$$\begin{aligned}
&\sum_{w_i \in Y'_n} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
&= \sum_{w_i \in Y_n \cap \overline{Y'_{n+1}}} (\alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)} - 1) \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
&+ \sum_{w_i \in \overline{Y_n} \cap Y'_{n+1}} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
&+ \sum_{w_i \in Y_n \cap Y'_{n+1}} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)}
\end{aligned} \tag{3.17}$$

$$
\begin{aligned}
= & \sum_{w_i \in Y_n \cap \overline{Y'_{n+1}}} (\alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1) \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
+ & \sum_{w_i \in \overline{Y_n} \cap Y'_{n+1}} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \cdot \prod_{n=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
+ & \sum_{w_i \in Y_n \cap Y'_{n+1}} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - \alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)} + \alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)} - 1 \right] \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \\
= & \sum_{w_i \in Y_n} (\alpha_{w_{i-n+1},\cdots,w_i}^{g(w_{i-n+1},\cdots,w_i)} - 1) \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} \qquad (3.18) \\
& + \sum_{w_i \in Y'_{n+1}} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1 \right] \cdot \prod_{j=1}^{n} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} . \qquad (3.19)
\end{aligned}
$$

▮

This proves the assertion of the lemma.

Note the first summation (3.18) takes $O(\#[\text{n-gram}])$ time, and the second one (3.19) has the same form as (3.17) does and can be computed recursively.

We summarize the above ideas in the following theorem.

Theorem

In the N-gram model of (3.6), the normalization factor $z(x)$ may be computed as

$$
z(x) = \sum_{w_i \in V} \alpha_{w_i} + \sum_{n=2}^{N} \sum_{w_i \in Y_n} \left[\prod_{j=n}^{N} \alpha_{w_{i-j+1},\cdots,w_i} - 1 \right] \cdot \prod_{j=1}^{n-1} \alpha_{w_{i-j+1},\cdots,w_i} \qquad (3.20)
$$

for all x in $O(\sum_{n=1}^{N} \#[\text{n-gram}])$ time, where Y_n, $n = 2, \cdots, N$, are defined in (3.14) and α's are set to unity if the corresponding g's are zero.

Proof:

$$
z(x) = \sum_{w_i} \prod_{j=1}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} .
$$

Define Y_n and Y'_n as in (3.14) and (3.15). Start from

$$
z(x) = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} + \sum_{w_i \in Y_x} [\prod_{j=2}^{N} \alpha_{w_{i-j+1},\cdots,w_i}^{g(w_{i-j+1},\cdots,w_i)} - 1] \cdot \alpha_{w_i}^{g(w_i)},
$$

apply Lemma 1 recursively $N-1$ times with $n = 2, \cdots, N-1$ to the second term in the right hand side of the equation above and take advantage of g being binary to derive (3.20).

The n^{th} inner summation $\sum_{w \in Y_n}$ in (3.20) has exact #[n-gram] terms for all histories. The overall complexity for computing z for all histories in $|\hat{X}|$ is strictly bounded by $O(\sum_{n=1}^{N} \#[\text{n-gram}])$.

∎

In the algorithm above, the first summation involves unigram features only; the second one involves unigram and bigram features, etc. The normalization factors are thus computed hierarchically. We call this method, the *hierarchical training* method.

Time Complexity

In language modeling, the computation time for all denominators z per iteration is $O(\sum_{n=1}^{N} \#[\text{n-gram}])$ for all $n = 1, 2, \cdots, N$, which is the same as that for training a corresponding interpolated or back-off N-gram model. However, a maximum entropy model needs several iterations, whereas interpolation and back-off models need only one.

Space complexity

The theorem above gives a constructive algorithm for computing the normalization factors z hierarchically. First, the summation over V is computed, then that over Y_2, etc. This algorithm needs some extra space to save the intermediate summations over V, Y_2, Y_3, etc. If N-grams are sorted lexicographically according to $w_i, w_{i-1}, w_{i-2}, \cdots, w_{i-N+1}$, the algorithm requires only $O(N)$ extra space more than that needed for unigram-caching.

The theorem above also holds when g's are real-valued functions. The following corollary extends the theorem for the case when the g's are not binary.

Corollary

In a maximum entropy model

$$p(y|x) = \frac{1}{z(x)} \prod_{j=0}^{N} \alpha_{x_1,\cdots,x_j,y}^{g(x_1,\cdots,x_j,y)} \tag{3.21}$$

where the feature functions are defined as

$$g(x_1,\cdots,x_j,y) \begin{cases} \neq 0 & \langle x_1,\cdots,x_j,y\rangle \text{ in the training data} \\ = 0 & \textit{otherwise} \end{cases} \quad \text{for } j = 1,\cdots,N, \tag{3.22}$$

the normalization factor $z(x)$ can be computed as

$$z(x) = \sum_{y\in\mathcal{Y}} \alpha_y^{g(y)} + \sum_{n=1}^{N} \left[\sum_{y\in Y_n} [\prod_{j=n}^{N} \alpha_{x_1,\cdots,x_j,y}^{g(x_1,\cdots,x_j,y)} - 1] \cdot \prod_{j=0}^{n-1} \alpha_{x_1,\cdots,x_j,y}^{g(x_1,\cdots,x_j,y)} \right]$$

where

$$Y_n \;=\; \{w : g(x_1,\cdots,x_{n-1},w) \neq 0 \text{ for some n-gram feature } g\}.$$

Time to compute the normalization factors z for all histories is $O(\sum_{n=0}^{N} S_n)$ for all x where $S_0 = V$ and S_n is the number of $n+1$-tuples $\langle x_1,\cdots,x_n,y\rangle$ seen in the training data. This time is exactly the same as that to gather empirical counts from the training data.

∎

3.3 Exploiting the Feature Hierarchy for Computing Maximum Entropy Models

N-gram models are a particular kind of ME model. Obviously if a word w_i has the n-gram feature activated for a given history $w_{i-n+1},\cdots,w_{i-1}$ then it must have $(n-1)$-gram, $(n-2)$-gram, ..., unigram features activated too. Therefore, for a given history, $Y_N \subset Y_{N-1}\cdots Y_2 \subset Y_1 = V$, as we have shown in the previous section. The efficiency of the hierarchical training method introduced in this previous section comes from this nested relation among N-gram features.

In the real world, however, many models have non-nested features in addition to nested ones. The training of these models is more complicated than that of N-gram models. The complexity of computing model parameters depends on the hierarchical structure of feature types in the model. Before we address the training methods for maximum entropy models in general, we need to describe the feature hierarchy of ME models in this section.

3.3.1 Relations between Features

We have shown that N-gram features are nested. Here we give a more formal definition for the nested relation between features.

Definition 1 (Nested Features):

Let $x' = x_{k_1}, \cdots, x_{k_l}$ be a history class and $x'' = x_{j_1}, \cdots, x_{j_m}, m < l$ be a subsequence of x'. $g_1(x', y)$ is identified by x' and y and $g_2(x'', y)$ is identified by x'' and y. If it holds that $g_1 = 1 \Rightarrow g_2 = 1$, then we say that g_1 is nested in g_2.

Example 1

The trigram feature function $g(w_{i-2}, w_{i-1}, w_i)$ is nested in the bigram feature function $g(w_{i-1}, w_i)$, and both the trigram and the bigram features are nested in the unigram feature $g(w_i)$. However, the topic-dependent feature $g(t_i, w_i)$ and the trigram feature function $g(w_{i-2}, w_{i-1}, w_i)$ are not nested in either direction since neither t_i is a subsequence of w_{i-2}, w_{i-1} nor vice verse.

Definition 2 (Nested Feature Types):

We say feature type-I is nested in feature type-II if any feature in the feature type-I is nested in a feature in type-II, i.e.,

$$\forall\ g_1 \in \text{type-I},\ \exists\ g_2 \in \text{type-II such that } g_1 \text{ is nested in } g_2.$$

Example 2

The trigram feature type (pattern) is nested in the bigram feature pattern because for each trigram feature $g(w_{i-2}, w_{i-1}, w_i)$, there is a bigram feature $g(w_{i-1}, w_i)$ that nests it.

Obviously, transitivity holds for the nested relation, i.e., if $g(w_{i-1}, w_i)$ is nested in $g(w_i)$ and $g(w_{i-2}, w_{i-1}, w_i)$ is nested in $g(w_{i-1}, w_i)$, then $g(w_{i-2}, w_{i-1}, w_i)$ is nested in $g(w_i)$.

We further classify features that are not nested as either non-overlapping features or overlapping features.

Definition 3 (Non-overlapping Features and Overlapping Features):

Two features g_j and g_k are *non-overlapping* if $g_j(x, y)$ and $g_k(x, y)$ cannot be active simultaneously for any $y \in V$ a given history $x \in \mathcal{X}$; otherwise, they are special *overlapping* features.

Example 3

For example, if w_i and w_j only occur in topic t_i and t_j, respectively, the features $g(t_i, w_i)$ and $g(t_j, w_j)$ are non-overlapping features. However, the bigram feature $g(w_{i-1}, w_i)$ and the topic feature $g(t_i, w_i)$ may be overlapping features, because for a given history (t_i, w_{i-1}), some words w_i may have both the topic feature and the bigram feature activated.

We can also define the *non-overlapping* and *overlapping* relations between feature types (patterns). If none of the features in type-I overlap with any features in type-II, then we say type-I and type-II are non-overlapping feature patterns.

Pure non-overlapping feature patterns are rare in an ME model, but some features may be treated as non-overlapping features in practice if their overlapping parts are very small. Readers will see that N-gram features and topic-dependent features are treated as non-overlapping features in Chapter 7.

Figure 3.4 shows a Venn diagram of sets of words Y_1 and Y_2 (for a given history x) activating two nested, non-overlapping and overlapping features, respectively.

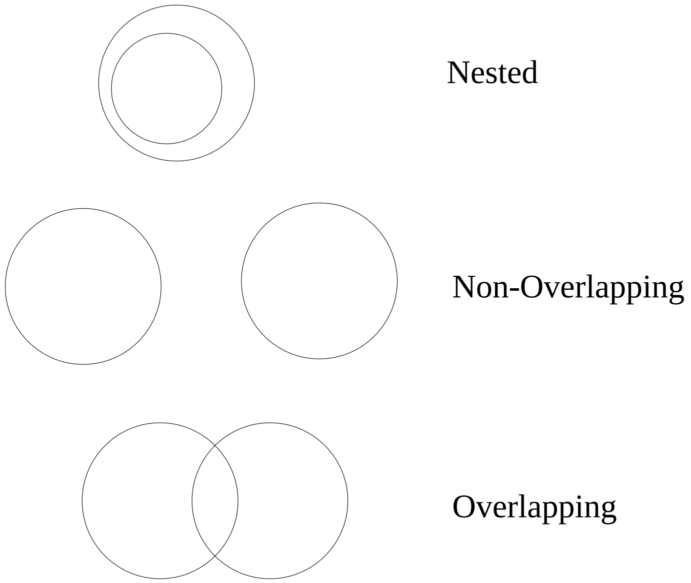


Figure 3.4: Nested, non-overlapping and overlapping features

3.3.2 Representing the Feature Hierarchy of ME Models in a Digraph

After defining the relations between features, we can now draw the feature hierarchy of a maximum model in a digraph. In this digraph, nodes are feature patterns. Two feature patterns $g(x'', y)$ and $g(x', y)$ are connected by a directed edge (or arc) $(g(x'', y), g(x', y))$ if $g(x'', y)$ is nested in $g(x', y)$. Figure 3.5 shows the structure of the topic model (1.8).

If all feature patterns of an ME model are ordered by the nested relation as are those in N-gram models, then the structure of this model is a unary tree as shown in Figure 3.6.

3.4 Expansion for Models with Non-Nested Features

Many maximum entropy models have some non-nested (overlapping) features. Therefore, they cannot be trained by the hierarchical method described in the previous

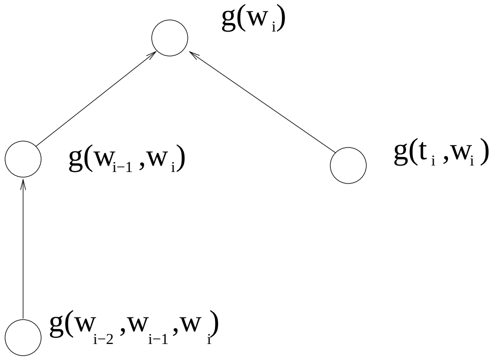


Figure 3.5: Structure of a topic-dependent model

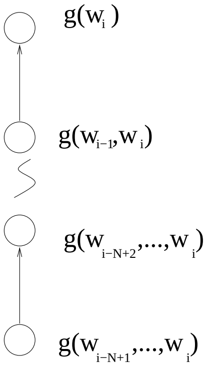


Figure 3.6: Structure of N-gram models

section. However the idea of sharing computation is still applicable. In this section we extend the ideas in Section 3.2 to ME models with non-nested features. We focus on a model whose conditional features are in the same order, or, in other words, the same layer in the feature hierarchy, as shown in Figure 3.7. None of conditional constraints in this kind of model are nested.

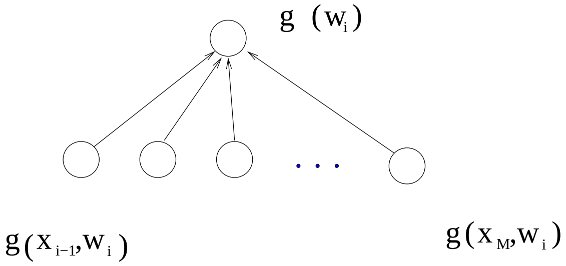


Figure 3.7: The model with a flat feature hierarchy

3.4.1 Training Models with Non-Nested and Overlapping Features

We again start from a simple model: a model with unigram features $g(w_i)$, bigram features $g(w_{i-1}, w_i)$ and the topic features $g(t_i, w_i)$. For this model

$$p(w_i|t_i, w_{i-1}) = \frac{\alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)}}{z(t_i, w_{i-1})}, \tag{3.23}$$

where

$$z(t_i, w_{i-1}) = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)}.$$

The feature hierarchy of the topic-dependent bigram model is shown in Figure 3.8.

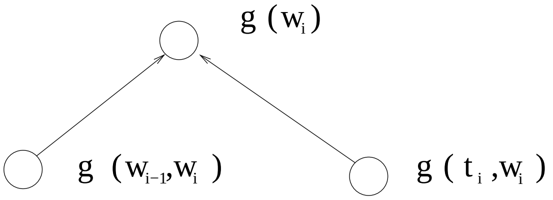


Figure 3.8: Feature hierarchical of the topic-dependent bigram model

The normalization constant z can be computed as

$$z(t_i, w_{i-1}) = \sum_{w_i \in V} \alpha_{w_i}^{g(w_i)} + \sum_{w_i \in Y_x} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} - 1) \cdot \alpha_{w_i}^{g(w_i)} \tag{3.24}$$

by unigram-caching in roughly $O(\sum_{x\in\hat{X}} |Y_x|)$ time for all seen (t_i, w_{i-1}) pairs. We notice that for $w \in \hat{Y}_x$, many of them have either regular bigram constraints or topic constraints but not both. We take advantage of this fact and simplify the computation of calculating z.

Similar to splitting Y_x in (3.24) according to bigram and trigram constraints, when training trigram models, we split Y_x into subsets Y_w and $\overline{Y_w} = Y_x - Y_w$, where

$$Y_w = \{w_i : g(w_{i-1}, w_i) = 1 \text{ for some bigram features }\},$$

and also into Y_t and $\overline{Y_t} = Y_x - Y_t$, where

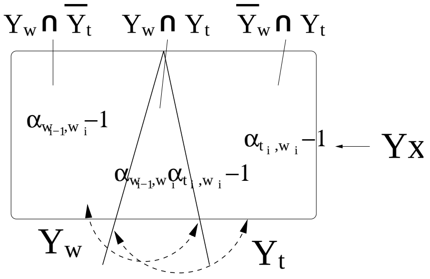
$$Y_t = \{w_i : g(t_i, w_i) = 1 \text{ for some topic features }\}$$

Table 3.2 shows the values of $g(w_{i-1}, w_i)$ and $g(t_i, w_i)$ for words w_i in $Y_w \cap \overline{Y_w}$, $\overline{Y}_w \cap Y_w$ and $Y_w \cap Y_t$ respectively.

word set	$g(w_{i-1}, w_i)$	$g(t_i, w_i)$
$Y_w \cap \overline{Y_w}$	1	0
$\overline{Y}_w \cap Y_w$	0	1
$Y_w \cap Y_t$	1	1

Table 3.2: Value of $g(w_{i-1}, w_i)$ and $g(t, w_i)$

Figure 3.9 shows the value of $\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t,w_i}^{g(t_i,w_i)} - 1$ in the three subsets of Y_x.



$Y_w \cap \overline{Y}_t$
$Y_w \cap Y_t$
$\overline{Y}_w \cap Y_t$
$\alpha_{w_{i-1},w_i} - 1$
$\alpha_{t_i,w_i} - 1$
Yx
$\alpha_{w_{i-1},w_i}\alpha_{t_i,w_i} - 1$
Y_w
Y_t

Figure 3.9: Division of Y_x according to bigram and topic constraints

Therefore,

$$(\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} - 1)\alpha_{w_i}^{g(w_i)} = \begin{cases} (\alpha_{w_{i-1},w_i} - 1)\alpha_{w_i} & w_i \in Y_w \cap \overline{Y_t} \\ (\alpha_{t_i,w_i} - 1)\alpha_{w_i} & w_i \in \overline{Y_w} \cap Y_t \\ (\alpha_{w_{i-1},w_i} \cdot \alpha_{t_i,w_i} - 1)\alpha_{w_i} & w_i \in Y_w \cap Y_t. \end{cases} \tag{3.25}$$

The second sum of $z(x)$ in (3.24) can thus be re-written as

$$\begin{aligned} & \sum_{w \in Y_x} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} - 1) \cdot \alpha_{w_i}^{g(w_i)} & (3.26) \\ = & \sum_{w_i \in Y_w \cap \overline{Y_t}} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} + \sum_{w_i \in \overline{Y_w} \cap Y_t} (\alpha_{t_i,w_i} - 1) \cdot \alpha_{w_i} \\ + & \sum_{w_i \in Y_w \cap Y_t} [(\alpha_{w_{i-1},w_i} - 1)(\alpha_{t_i,w_i} - 1) + (\alpha_{w_{i-1},w_i} - 1) + (\alpha_{t_i,w_i} - 1)]\alpha_{w_i} \\ = & \sum_{w_i \in Y_w} (\alpha_{w_{i-1},w_i} - 1) \cdot \alpha_{w_i} + \sum_{w_i \in Y_t} (\alpha_{t_i,w_i} - 1) \cdot \alpha_{w_i} \\ + & \sum_{w_i \in Y_w \cap Y_t} (\alpha_{w_{i-1},w_i} - 1) \cdot (\alpha_{t_i,w_i} - 1) \cdot \alpha_{w_i}. & (3.27) \end{aligned}$$

Analysis of Complexity

- The first summation in (3.27) depends only on w_{i-1}. It can be computed in $O(B)$ time for all histories and cached for re-use.

- The second one depends only on t_i, and can be computed in $O(B_t)$, where B_t is the number of topic unigram constraints, and then shared by the histories with the same t_i. Note that these two numbers are strictly smaller than the size of training data L.

- The complexity for the last summation is $O(|Y_w \cap Y_t|)$ time for each history and $O(\sum_{(w_{i-1},t_i) \in \hat{X}} |Y_w \cap Y_t|)$ for all histories.

In summary, computing z for all "seen" histories by (3.27) will take $O(U + B + B_t + \sum_{(w_{i-1},t_i) \in \hat{X}} |Y_w \cap Y_t|)$ time. U, B and B_t are very small compared to $\sum_{(w_{i-1},t_i) \in \hat{X}} |Y_w \cup Y_t|$ and thus can be omitted. Since the baseline unigram-caching method takes

$O(\sum_{(w_{i-1},t_i)\in\hat{X}} |Y_w \cup Y_t|)$ time, on average, the speed-up of using (3.27) is roughly the ratio of $|Y_w \cup Y_t|$ and $|Y_w \cap Y_t|$.

$|Y_w \cap Y_t|$ is very small if $g(w_{i-1}, w_i)$ and $g(t_i, w_i)$ are almost non-overlapping features. On the other hand, the size of this intersection can be as large as $\min(|Y_w|, |Y_t|)$ if $g(w_{i-1}, w_i)$ is almost nested in $g(t_i, w_i)$ or vice versa. This implementation is practical[4] if $|Y_w \cap Y_t| << |Y_w \cup Y_t|$. For example, in Switchboard, $|\hat{X}| \approx 700,000$, $Y_w \approx 200$, $|Y_t| \approx 200$ and $|Y_w \cup Y_t| \approx 330$, so $|\hat{X}| \cdot |Y_w \cup Y_x| \approx 2 \cdot 10^9$. However, $|Y_w \cap Y_t| \approx 50$, $B \approx 300,000$ and $B_t \approx 15,000$, so $|\hat{X}| \cdot |Y_w \cap Y_x| + B + B_t \approx 4 \cdot 10^8$. Using (3.27) to train the model will result in a five- to six-fold speed-up compared to using (2.30) directly. In Broadcast News, the speed-up from (3.27) is roughly fourfold. Since the gain of this implementation comes from sharing the computation among the "non-overlapping" part of Y_w and Y_t, we will refer to (3.27) as *non-overlapping sharing* in the remainder of this dissertation.

Unlike N-gram models whose training complexity is strictly bounded by L, i.e., the size of training data, non-nested models usually cannot be trained in $O(L)$ time because $\sum_{(w_{i-1},t_i)\in\hat{X}} |Y_w \cap Y_t|$ may exceed L. For a given history class (w_{i-1}, t_i) in the training data, $Y_w \cap Y_t$ contains words w_i where (w_{i-1}, w_i) occur and (t_i, w_i) occur in the training data no matter whether the 3-tuple $\langle w_{i-1}, t_i, w_i\rangle$ occurs or not. For example, the phrase "a big dog and a small cat" appears in the topic *pets*, but "big cat" does not occur in the topic *pets* even though it does appear in other topics. However, "cat" still needs to be considered in $|Y_w \cap Y_t|$ when the history is "w_{i-1} =big, t_i=pets." As a result, many triples $\langle w_{i-1}, w_i, t_i\rangle$ not seen in the training data have to be considered in computing z.

Extra Space Requirement

One minor drawback of non-overlapping sharing is the added space complexity. Summations over Y_t and Y_w must be saved for all topics and words respectively in order to compute the summation over $Y_t \cap Y_w$. Therefore, extra space of $O(|V|)$ plus $O(|T|)$ is required where T is the number of topics. This drawback is minor for only

[4] Although $|Y_w \cap Y_t| < |Y_w \cup Y_t|$ always holds since *intersection is smaller than union*, this implementation is not necessary if the speed-up is not substantial.

two kinds of overlapping features. However, it may become a severe problem when a large number of overlapping features are active simultaneously.

3.4.2 Model with M Kinds of Bigram Constraints

The method above can be extended to a model with a unigram feature $g(w)$ and M bigram-like features $g(u_1, w), g(u_2, w), \cdots, g(u_M, w)$:

$$p(w|u_1, \cdots, u_M) = \frac{\alpha_w^{g(w)} \cdot \alpha_{u_1,w}^{g(u_1,w)} \cdots \alpha_{u_M,w}^{g(u_M,w)}}{z(u_1, \cdots, u_M)}, \tag{3.28}$$

where

$$\begin{array}{rcl} z(x) & = & z(u_1, \cdots, u_M) \\ & = & \sum_w \alpha_w^{g(w)} + \sum_{w \in Y_x} \alpha_w^{g(w)} [\alpha_{u_1,w}^{g(u_1,w)} \cdots \alpha_{u_M,w}^{g(u_M,w)} - 1]. \end{array}$$

We again take advantage of the fact that *the intersection of sets is smaller than their union* and split the summation over Y_x into several summations over smaller subsets. We first define "base" word subsets as

$$Y_n = \{w : g(u_n, w) = 1 \text{ for } n = 1, \cdots, M\}.$$

Y_x can be thus split into Y_n and $Y_x - Y_n$ in M ways, one for each n. There are 2^M different intersections I of sets $Y_1, Y_2, \cdots, Y_M$ as listed in Table 3.3.

$$\begin{array}{rcllllll} I_0 & = & Y_x & & & & \\ I_1 & = & Y_1 & & & & \\ I_2 & = & & Y_2 & & & \\ I_3 & = & Y_1 & \cap Y_2 & & & \\ & & & \vdots & & & \\ I_{2^M-3} & = & Y_1 & & \cap Y_3 & \cdots & Y_M, \\ I_{2^M-2} & = & & Y_2 & \cap Y_3 & \cdots & Y_M, \\ I_{2^M-1} & = & Y_1 & \cap Y_2 & & \cdots & Y_M, \end{array}$$

Table 3.3: All 2^M intersections of M subsets.

Let $Y_{m_1} \cap Y_{m_2} \cap \cdots \cap Y_{m_P}$ be the intersection of P base subsets $Y_{m_1}, Y_{m_2}, \cdots, Y_{m_P}$ of Y_x where $m_1, \cdots, m_P \in \{1, 2, \cdots, n\}$ are their indices respectively. We choose a subscript k for this intersection as

$$k = \sum_{q=1}^{P} 2^{m_q - 1},$$

and denote $Y_{m_1} \cap Y_{m_2} \cap \cdots \cap Y_{m_P}$ by I_k for simplicity. From the subscript k we can find which subsets are needed to construct the intersection I.

Table 3.4 illustrates the bijection between the intersection I_k and the binary number k.

base sets	Y_1	Y_2	Y_3	$\cdots$	Y_{M-2}	Y_{M-1}	Y_M
I_k	1	0	1	$\cdots$	0	1	1

Table 3.4: The mapping between intersections and binary numbers. I_k is the intersection of subsets with value 1 under them. k is a binary with value 1 in corresponding bits.

For instance, when $M = 3$,

$$\begin{aligned} I_0 &= Y_x \\ I_1 &= Y_1 \\ I_2 &= Y_2 \\ I_3 &= Y_1 \cap Y_2 \\ I_4 &= Y_3 \\ I_5 &= Y_1 \cap Y_3 \\ I_6 &= Y_2 \cap Y_3 \\ I_7 &= Y_1 \cap Y_2 \cap Y_3 \end{aligned}$$

Now we examine the value of $\alpha_{u_j,w}^{g(u_j,w)} - 1$ (for $j = 1, \cdots, M$) for words $w \in Y_1, \cdots, Y_M$ and find it is non-zero only for $w \in Y_j$. In more general cases, the product $\prod_{q=1}^{P}(\alpha_{u_{m_q},w}^{g(u_{m_q},w)} - 1)$ is non-zero only for word $w \in I_k$ where $k = \sum_{q=1}^{P} 2^{m_q - 1}$.

Therefore,

$$\sum_{w\in V}\prod_{j=1}^{P}(\alpha_{u_{m_j},w}^{g(u_{m_j},w)}-1)=\sum_{w\in I_k}\prod_{j=1}^{P}(\alpha_{u_{m_j},w}^{g(u_{m_j},w)}-1).$$

This property is very useful in deriving the algorithms for computing the normalization factors z in the model with M overlapping bigram constraints.

We need to define another shorthand notation

$$A_k \doteq (\alpha_{u_{m_1},w}^{g(u_{m_1},w)}-1)\cdots(\alpha_{u_{m_P},w}^{g(u_{m_P},w)}-1)\alpha_w^{g(w)} \tag{3.29}$$

where k is the index for the intersection of P subsets Y_{m_1}, Y_{m_1}, $\cdots$, Y_{m_P}; for instance,

$$A_0 = \alpha_w, \text{ the unigram parameters and}$$

$$A_{2^M-1} = \alpha_w \prod_{k=1}^{M}(\alpha_{u_k,w}-1).$$

We can write the property about I_k using the notation A_k as

$$\sum_{w\in V} A_k = \sum_{w\in I_k} A_k. \tag{3.30}$$

There is also an important property of the A_k's. We can rewrite $\alpha_w^{g(w)}\prod_{i=1}^{M}\alpha_{u_i,w}^{g(u_i,w)}$ as

$$\begin{aligned}\alpha_w^{g(w)}\prod_{i=1}^{M}\alpha_{u_i,w}^{g(u_i,w)} &= \alpha_w^{g(w)}\prod_{i=1}^{M}[(\alpha_{u_i,w}^{g(u_i,w)}-1)+1] \\ &= \sum_{k=0}^{2^M-1} A_k.\end{aligned} \tag{3.31}$$

We now can derive the following lemma, which is an extension of Equations (3.26)-(3.27), from the two properties above.

Lemma 2: In the model (3.28), the normalization factor can be computed as

$$z(x) = \sum_{k=0}^{2^M-1}\sum_{w\in I_k} A_k. \tag{3.32}$$

proof:

Using the property (3.31), $z(x)$ can be rewritten as

$$\begin{aligned} z(x) &= \sum_{w \in V} \alpha_w^{g(w)} \cdot \prod_{i=1}^{M} \alpha_{u_i,w}^{g(u_i,w)} \\ &= \sum_{w \in V} [\sum_{k=0}^{2^M-1} A_k]. \end{aligned}$$

Using the property (3.30), it can be rewritten as

$$z(x) = \sum_{k=0}^{2^M-1} \sum_{w \in I_k} A_k.$$

▮

3.4.3 Analysis of Complexity

First we analyze the complexity when $M = 3$:

$$\begin{aligned} z(u_1, u_2, u_3) &= \sum_{w \in Y_x} A_0 + \sum_{w \in Y_1} A_1 + \sum_{w \in Y_2} A_2 + \sum_{w \in Y_3} A_4 \\ &+ \sum_{w \in Y_1 \cap Y_2} A_3 + \sum_{w \in Y_1 \cap Y_3} A_5 + \sum_{w \in Y_2 \cap Y_3} A_6 \\ &+ \sum_{w \in Y_1 \cap Y_2 \cap Y_3} A_7. \end{aligned}$$

The first four summations can be done in $O(U)$, $O(B_1)$, $O(B_2)$ and $O(B_3)$ time, respectively, where B_1, B_2 and B_3 are numbers of bigram constraints of the kind $\langle u_1, w\rangle$, $\langle u_2, w\rangle$ and $\langle u_3, w\rangle$, respectively. The three following sums require the time of $O(\sum_{u_1,u_2 \in \hat{X}} |Y_1 \cap Y_2|)$, $O(\sum_{u_1,u_3 \in \hat{X}} |Y_1 \cap Y_3|)$ and $O(\sum_{u_2,u_3 \in \hat{X}} |Y_2 \cap Y_3|)$, respectively as analyzed in Section 3.3.1. The last one needs $O(\sum_{u_1,u_2,u_3 \in \hat{X}} |Y_1 \cap Y_2 \cap Y_3|)$ time. $O(U + B_1 + B_2 + B_3)$ is small comparative to the rest and thus may be ignored. This implementation, therefore, is more efficient than unigram-caching when

$$\begin{aligned} \sum_{u_1,u_2 \in \hat{X}} |Y_1 \cap Y_2| + \sum_{u_1,u_3 \in \hat{X}} |Y_1 \cap Y_3| + \sum_{u_2,u_3 \in \hat{X}} |Y_2 \cap Y_3| + \sum_{u_1,u_2,u_3 \in \hat{X}} |Y_1 \cap Y_2 \cap Y_3| \\ << \sum_{u_1,u_2,u_3 \in \hat{X}} |Y_1 \cup Y_2 \cup Y_3|. \end{aligned}$$

Obviously, each sum on the left-hand side is strictly less than that on the right-hand side. However, it is not guaranteed that the four sums together on the left-hand side are still significantly less than the right-hand side if the intersections $Y_1 \cap Y_2$, $Y_1 \cap Y_3$, $Y_2 \cap Y_3$ and $Y_1 \cap Y_2 \cap Y_3$ are large. Therefore, this implementation is useful only when the extent of overlap between features is small. In practice, it will reduce the computation by about 50% in training syntactic models.

Now we analyze the training time for the model with M arbitrary kinds of bigrams. $\sum_{w \in I_{2^k-1}} A_{2^k-1}$ for $k = 1, \cdots, M$ is fixed for all histories sharing the same symbols u_k at positions k, and it needs to be computed only once and may then be reused. One can easily check that this sum needs only $O(B_k)$ time, where B_k is the number of "bigrams" $\langle u_k, w_i \rangle$ seen in the training data.

It is difficult to precisely enumerate the complexity of other terms, whose complexity depends on two factors, the number of different history subsequence $u_{m_1}, \cdots, u_{m_P}$ denoted as H_k, $k = \sum_{j=1}^{P} 2^{m_j}$, and the average size of I_k. Usually, the larger the size of I_k, the smaller is H_k, and vice versa. If, fortunately, all features are almost non-overlapping features, i.e., if $\cap_k I_k \approx \phi$, then the computational load is quite low. On the other hand, if some features are heavily overlapping, this implementation may not save computation. In the worst case, this implementation could be even more expensive than a direct implementation of (3.29).

Next we provide an upper bound of the running time of this non-overlapping sharing method and give a sufficient condition for using this method instead of unigram-caching in training maximum entropy models. Let

$$T_k = \# \text{ of terms in } \sum_{w \in I_k} A_k \text{ for all seen histories.}$$

E.g., $T_0 = U$, $T_1 = B_1$, etc. Let

$$T_{max} = \max_k T_k. \tag{3.33}$$

The overall complexity is then less than $O(2^M \cdot T_{max})$.

If this number $2^M \cdot T_{max}$ is less than the number of terms in unigram-caching, then non-overlapping sharing is worth applying. T_{max} is usually close to T_{2^M-1}. Care must be taken regarding two factors when using this method. First, the size of M should be small because 2^M increase exponentially in M. In practice, M should be less than four in language modeling. Second, redundant features should be avoided so that the intersections of two or more Y_k subsets are kept small in size. We will show in Section 5.5 that dropping some redundant syntactic features will not degrade the precision of the syntactic model.

Concerning Extra Space Requirement

Summations over I_k for $k = 1, \cdots, 2^M - 1$ must be pre-computed and stored in order to compute the summation over Y_x. Therefore, the extra space of $O(\sum_{k=1}^{2^M-1} I_k)$ is required.

More Discussion

This approach can be extended easily to train a model with only M *"order n"* conditional features. As we will show in subsequent sections, it can also be used with other simplifications in training methods.

In general, the training of models with overlapping features is less efficient than that of the N-gram models because, as stated before, of the need to consider many tuples that do not appear in the training data; specifically,

1. the right hand side of the Equation (3.32) has 2^M summations instead of N as in (3.20), and
2. the computational load for each summation $\sum_{w \in I_k} A_k$ over the overlapping parts may be enormous if feature patterns are heavily overlapping.

It is worth mentioning here that cluster expansion in Lafferty & Suhm (1996) works similarly and has the same complexity as this method for models with one kind of unigram features and M kinds of bigram features. The difference is that we only use non-overlapping sharing for the features in the same hierarchy, while they applied the idea to all features. Therefore, non-overlapping sharing can be regarded as a special case of cluster expansion.

3.5 Generalized Hierarchical Training Methods

Section 3.2 shows the best cases of maximum entropy models whose features are all nested. Their training time of $O(L)$ has already reached the lower bound of the empirical estimation. Section 3.4 shows the worst cases in which none of conditional features are nested. Most maximum entropy models lie in the middle between these two extreme cases: they have both hierarchically nested features and non-nested features. In this section, we will combine the ideas earlier shown for these two cases, and achieve a more general approach for ME models that we call generalized hierarchical training (GHT).

Most of maximum entropy models are designed to combine several sources of information together. It is natural to assume that these information sources are mutually non-redundant, even though the maximum entropy method does not require this assumption.

Let M be the number of information sources from which the constraints of the model are selected, and let N be the highest order of constraints. We first derive generalized hierarchical training methods based upon the following two assumptions:

> i. *independence assumption:* all information sources are non-overlapping. (The subsets of the "history" x representing each information source from a disjoint partition of x.), and
> ii. *hierarchy assumption:* all feature patterns within an information source are nested.

Later, we extend GMH to models where these two assumptions need not hold.

3.5.1 Training MN-gram Models Hierarchically

MN-gram Model

Let M be the number of information sources, and N be the highest order of constraints. We call the maximum entropy model an MN-gram model if it satisfies the independence and hierarchy assumptions. For example, a regular N-gram model can be regarded as a 1N-gram model, and the model in (3.28) is an M2-gram model.

We need to introduce some notation before we discuss the training of MN-gram models. Let

$$u_{1,1}^{M,N-1} = x = \langle u_{1,1}, u_{1,2}, \cdots, u_{1,N-1}, u_{2,1}, \cdots, u_{2,N-1}, \cdots, u_{M,1}, \cdots, u_{M,N-1} \rangle$$

be a history equivalence class. If the order of constraints from some information source i is less than $N-1$, we simply set the corresponding high order $u_{i,j+1}, \cdots, u_{i,N-1}$ to *null.*

Let $g(w)$ be the unigram feature function, $g(u_{i,1}, w)$, $\cdots$, $g(u_{i,1}, \cdots, u_{i,N-1}, w)$ for $i = 1, 2, \cdots, M$ be bigram, $\cdots$, N-gram features from information source i.

The maximum entropy MN-gram model can be written as

$$p(w|u_{1,1}^{M,N-1}) = \frac{\alpha_w^{g(w)} \prod_{i=1}^{M} \prod_{j=2}^{N} \alpha_{u_{i,1}^{i,j-1},w}^{g(u_{i,1}^{i,j-1},w)}}{z(u_{1,1}^{M,N-1})}, \tag{3.34}$$

where

$$\begin{aligned} u_{i,1}^{i,j-1} &= u_{i,1}, \cdots, u_{i,j-1}, \text{ and} \\ z(u_{1,1}^{M,N-1}) &= \sum_{w \in V} \alpha_w^{g(w)} \prod_{i=1}^{M} \prod_{j=2}^{N} \alpha_{u_{i,1}^{i,j-1},w}^{g(u_{i,1}^{i,j-1},w)}. \end{aligned} \tag{3.35}$$

Note that we use subscript i for the information source, and j for the order. The left digraph in Figure 3.10 shows the feature hierarchy of MN-models. Now we hierarchize this MN-gram model to a model that looks like N-gram models.

Hierarchizing

Let $U_j = (u_{1,j}, \cdots u_{M,j})$ for $j = 1, 2, \cdots, N-1$[5]. We define a *super feature* f as

$$f(U_1, \cdots, U_{j-1}, w) = \begin{cases} 1 & \text{if } g(u_{i,1}, \cdots, u_{i,j-1}, w) = 1 \text{ for some } i, \\ 0 & \text{otherwise} \end{cases}$$

[5] If the highest order of constraints from some information source i is less than j, we simply let the feature corresponding $u_{i,j}$ be empty.

and its parameter:

$$\beta_{U_1,\cdots,U_{j-1},w} = \prod_{i=1}^{M} \alpha_{u_{i,1},\cdots,u_{i,j-1},w}^{g(u_{i,1},\cdots,u_{i,j-1},w)}.$$

We call this procedure *hierarchizing.* For example, in the model (3.23), the bigram feature $g(w_{i-1}, w_i)$ and the topic feature $g(t_i, w_i)$ are compounded to generate the super bigram features

$$f(w_{i-1}, t_i, w_i) = \begin{cases} 0 & \text{if } g(w_{i-1}, w_i) = 1 \text{ or } g(t_i, w_i) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

After hierarchizing, *all super-features f_i are now nested features by construction.* Model (3.34) becomes

$$p(w|U_1, \cdots, U_{N-1}) = \frac{\beta_w^{f(w)} \prod_{j=2}^{N} \beta_{U_1^{j-1},w}^{f(U_1^{j-1},w)}}{z(U_1, \cdots, U_{N-1})} \tag{3.36}$$

where

$$\begin{aligned} U_1^{j-1} &= U_1, U_2, \cdots, U_{j-1}, \\ z(U_1, \cdots, U_{N-1}) &= \sum_{w \in V} \beta_w^{f(w)} \prod_{j=2}^{N} \beta_{U_1^{j-1},w}^{f(U_1^{j-1},w)}. \end{aligned}$$

Figure 3.10 shows the feature hierarchies of MN-gram model before and after this transform. Now we show the hierarchical training method for model (3.36). We define base word subsets

$$Y_{i,j} = \{w : g(u_{i,1}, \cdots, u_{i,j-1}, w) = 1\},$$

and

$$Y_j = \cup_{i=1}^{M} Y_{i,j}.$$

We also define

$$Y_j' = \cup_{k=j}^{N} Y_k.$$

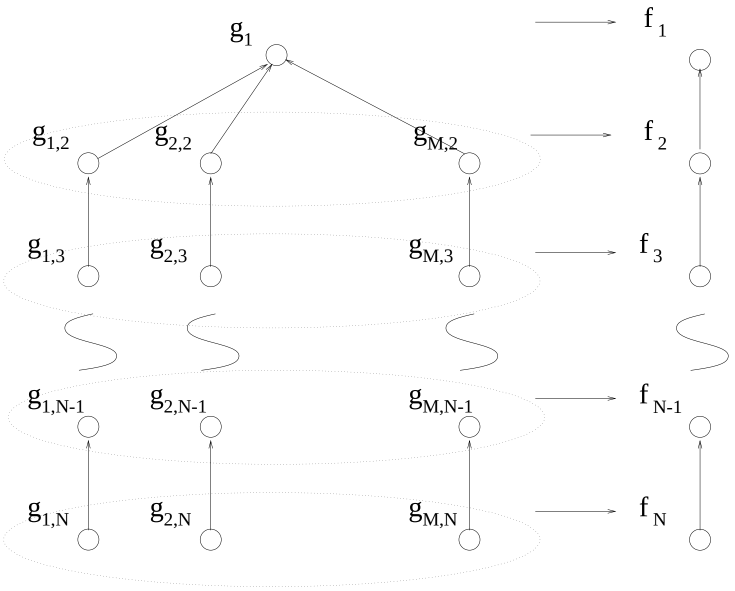


Figure 3.10: Converting the MN-gram model to the n-gram model with super features. $g_{[1]} = g(w)$ and $g_{[j,i]} = g(u_{i,1}, \cdots, u_{i,j-1})$ for simplicity.

We extend the recursive equation (3.16) in Lemma 1 as

$$\sum_{w\in Y_j'}[\prod_{n=j+1}^{N}\beta_{U_1^{n-1},w}^{f(U_1^{n-1},w)}-1]\cdot\prod_{n=1}^{j}\beta_{U_1^{n-1},w}^{f(U_1^{n-1},w)} \tag{3.37}$$

$$= \sum_{w\in Y_j}(\beta_{U_1^j,w}^{f(U_1^j,w)}-1)\cdot\prod_{n=1}^{j}\beta_{U_1^{n-1},w}^{f(U_1^{n-1},w)} + \sum_{w\in Y_{j+1}'}\left[\prod_{n=j+2}^{N-1}\beta_{U_1^n}^{f(U_1^n,w)}-1\right]\cdot\prod_{n=1}^{j+1}\beta_{U_1^{n-1}}^{f(U_1^{n-1},w)}.$$

Only the first summation on this right hand side in (3.37) need be calculated directly; the second one will be evolved recursively. If M is small, We can use the trick presented in Section 3.4 to compute the first summation.

We call this method *generalized hierarchical training.*

Analysis of Complexity

The computation of z is split into N summations $\sum_{w\in Y_j}$ for $j = 1, \cdots, N$. We need to mention here that the time needed for each summation $\sum_{w\in Y_j'}[\prod_{n=j+1}^{N}\beta_{U_1^{n-1},w}^{f_n(U_1^{n-1},w)} - 1]\cdot\prod_{n=1}^{j}\beta_{U_1^n,w}^{f_n(U_1^n,w)}$ is no longer bounded by the size of the training data, and it is roughly $O(2^M \cdot T_{max})$ where T_{max} is defined in (3.33) in Section 3.4.3. However, in practice, this implementation of GHT is still much more efficient than the baseline. In the baseline method, the time complexity can be estimated as $O(|\hat{X}| \cdot |\bar{Y}_x|)$ where $|\hat{X}|$ is the number of $u_{1,1}, \cdots, u_{M,N-1}$ seen in the training data and $|\bar{Y}_x| > |\bar{Y}_2|$. In the generalized hierarchical method, the complexity can be roughly estimated as $O(\sum_{j=2}^{N} |\hat{X}_j| \cdot |\bar{Y}_j|)$ where $|\hat{X}_j|$ is the number of history class $u_{1,1}, \cdots, u_{M,j-1}$ seen in the training data. Since $|\bar{Y}_N| < \cdots |\bar{Y}_j| \cdots < |\bar{Y}_3| << |\bar{Y}_2|$ and $|\hat{X}_2| << |\hat{X}|$,

$$\sum_{j=2}^{N} |\hat{X}_j| \cdot |\bar{Y}_j| < |\hat{X}_2| \cdot |\bar{Y}_2| + (N-2)|\hat{X}| \cdot |\bar{Y}_3| << |\hat{X}| \cdot |\bar{Y}_x|. \tag{3.38}$$

We can approximate the running time by $O(|\hat{X}_2| \cdot |\bar{Y}_2| + (N-2)|\hat{X}| \cdot |\bar{Y}_3|)$. The summation $\sum_{w \in Y_j}$ in (3.37) can be computed directly or aggregated from the 2^M partial summations over intersections of $Y_{i,j}$'s using the non-overlapping sharing described in the preceding section, depending on which is more efficient. If distributed to 2^M summations, the complexity is strictly less than $O(N \cdot 2^M \cdot T_{max})$ where M here is the maximum number of overlapping features in any super-feature and T_{max} is as defined in Section 3.4.

3.5.2 Training a General ME Model Hierarchically

Now we show that the generalized hierarchical training method still applies without the above independence and nested assumptions.

We need to define some terms and notation that will be used later.

Definition (partial order set POSET):

A partial order "<" is an order defined for some, but not necessarily all, pairs of elements on a set $\mathcal{S}$.

Partial order set (POSET) $(\mathcal{S}, <)$ is a set of elements that are subject to a partial order $<$.

An element e_m in a POSET $(\mathcal{S}, <)$ is *a minimal element* if there is no element e such that $e < e_m$. It is also a *minimum element* if $e_m < e$ for any element e in the POSET. An element e_M in a POSET $(\mathcal{S}, <)$ is *a maximal element* if there is no element e

such that $e_M < e$. It is also a *maximum element* if $e < e_M$ for any element e in the POSET.

Let $x = u_1, u_2, \cdots, u_k$ be a history class determined by k symbols, and $s = u_{i_1}, u_{i_2}, \cdots, u_{i_{k'}}$ be a *subsequence* of x determined by k' symbols. All subsequences s of x create a *partial order set* (POSET) $\mathcal{S}$ of 2^k elements with the ordering relation $s < s'$ if s is a subsequence of s'. Any feature types in an ME model are a function of a subsequence s and a predicted symbol y.

Let G be the set of features. Obviously, the nested relation is a partial order relation on G. Therefore, all feature types form a partial order set with the *minimum element (or feature)* $g(y)$. There are some *maximal elements (or features)* in this POSET, but the *maximum element* may not exist[6]. The following (*maximal first*) algorithm provides a recursive way of hierarchizing ME models.

Algorithm (Hierarchizing)

Step 1: Create super-features $f(s', y)$ from all common maximal features $g(s_1, y), g(s_2, y), \cdots, g(s_j, y)$ where s' is the *shortest super-sequence*[7] for all $s_1, s_2, \cdots, s_j$.

$$f(s', y) = 1 \text{ iff some } f(s_i, y) = 1 \text{ for } i = 1, 2, \cdots, j.$$

Note that $f(s', y)$ is a super feature of the highest order.

Step 2: Construct the remaining feature set G' by removing all maximal features $g(s_1, y)$, $g(s_2, y), \cdots, g(s_j, y)$ from G *i.e.,*

$$G' = G - \{g(s_1, y), g(s_2, y), \cdots, g(s_j, y)\}.$$

Step 3: Set $G = G'$.
Repeat Step 1 and 2 recursively and create super-features $f(s'', y), \cdots, f(s^n, y)$.

∎

See Figure 3.11 for an example of hierarchizing.

It can be easily checked that all these super-features $f(s', y), f(s'', y), \cdots, f(s^n, y)$ are nested features. We need only to show that $f(s', y)$ is nested in $f(s'', y)$. If

[6] Maximum element exists if and only if there is a feature $g(s^*, y)$ such that $s < s^*$ for all $s \in \mathcal{S}$. Actually it is not important whether the maximum element exist here.

[7] s' is a super-sequence of s if and only if s is a subsequence of s'.

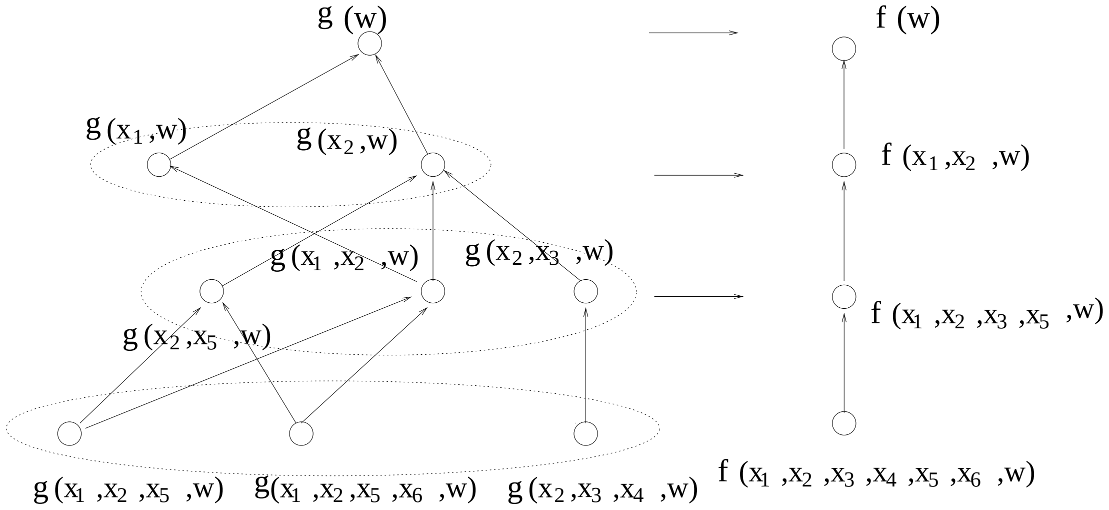


Figure 3.11: Hierarchizing: non-nested to nested

$f(s', y) = 1$, there must exist a $g(s_j, y) = 1$ where $g(s_j, y)$ is a maximal element. $g(s_j, y)$ must be nested in a maximal element $g(s_k, y) \in G'$ according to the feature hierarchical structure, and $g(s_k, y)$ is contained in $f(s'', y)$ according to the hierarchizing procedure above. $g(s_j, y) = 1 \Rightarrow g(s_k, y) = 1 \Rightarrow f(s'', y) = 1$. Therefore, $f(s', y)$ is nested in $f(s'', y)$. Of course, there are other ways of creating nested super-features from the original feature set. The algorithm above is only one example of hierarchizing. After hierarchizing, we can use the hierarchical training method in Section 3.2 together to train such an ME model.

The complexity analysis is similar to that of the MN-gram model. Let M be the maximum number of features a super feature can contain and N be the highest of order of a feature. We can use the non-overlapping sharing described in Section 3.4 in addition to the generalized hierarchical training method to deal with the overlapping features inside a super-feature, or use the generalized hierarchical training individually if the number of interacting features is too large. The complexity of the former is roughly $O(|\hat{X}_2| \cdot |\bar{Y}_2| + (N-2)|\hat{X}| \cdot |\bar{Y}_3|)$ and that of the latter is roughly $O(N \cdot 2^M \cdot T_{max})$. Overall, we can roughly estimate the complexity as $\min(O(|\hat{X}_2| \cdot |\bar{Y}_2| + (N-2)|\hat{X}| \cdot |\bar{Y}_3|), O(N \cdot 2^M \cdot T_{max})$ using the notation for MN-gram models. If $M \cdot N$ is a constant, the model with a large N can be trained more efficiently than the one with a large

M because the former has better feature hierarchy.

It is again worth mentioning that the cluster expansion method of Lafferty & Suhm (1996) does not take advantage of hierarchical structure among features, and treats the features of all orders as if on a flat level. Therefore, it results in less efficiency compared to GHT. The complexity for training an MN-gram model using cluster expansion is $O(2^N \cdot 2^M \cdot T'_{max})$ instead of $O(N \cdot 2^M \cdot T_{max})$, and it is easy to check that $T'_{max} > T_{max}$.

3.6 Divide-and-Conquer

The generalized hierarchical training method is designed for general purposes, but may not be the optimal method for some models. For instance, to train the following topic model

$$p(w_i|w_{i-2}, w_{i-1}, t_i) = \frac{\alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)}}{z(w_{i-2}, w_{i-1}, t_i)}, \tag{3.39}$$

using (3.37), we compound $g(w_{i-1}, w_i)$ and $g(t_i, w_i)$ to generate a super-feature $f(w_{i-1}, t_i, w_i)$. We also need to define $f(w_{i-2}, w_{i-1}, t_i, w_i) = g(w_{i-1}, w_{i-1}, w_i)$ so that super trigram features can be nested within super bigram features. The computational load corresponding to $f(w_{i-1}, t_i, w_i)$ depends on the number of words having a bigram feature $g(w_{i-1}, w_i)$ activated and a topic feature $g(t_i, w_i)$ activated, i.e., the size of $Y_w \cap Y_t$. Even though $Y_w \cap Y_t < Y_w \cup Y_t$, and thus the generalized hierarchical training method, is more efficient than the baseline approach, it is still far more costly than the optimal one we can reach for the topic model.

Therefore, we will introduce another efficient training method for topic-dependent models. One may observe that within a given topic t_i the value of the topic feature $g(t_i, w_i)$ is independent of w_{i-2}, w_{i-1} and can be treated as a marginal feature. We can take advantage of this and use a divide-and-conquer approach [8] to train the topic-dependent model at a very low cost. This approach involves five major steps:

[8] Divide-and-conquer here is slightly different from the concept in computer algorithms which means dividing the computation recursively. Here the computation is only divided once.

Step 1: Partition the training text into topics $D_1, D_2, \cdots, D_K$ where K here is the number of topics.

Step 2: For the training data of topic t denoted as D_t, collect N-grams.

Step 3: Compute all normalization factors z in same topic t.

Define

$$f_t(w_i) = g(w_i) \text{ for } t = 1, \cdots, K$$

and

$$\alpha'_{w_i} = \begin{cases} \alpha_{w_i} \cdot \alpha_{t_i,w_i} & \text{if } g(t_i, w_i) = 1, \\ \alpha_{w_i} & \text{otherwise.} \end{cases} \tag{3.40}$$

then

$$z(w_{i-2}, w_{i-1}, t_i) = \sum_{w \in V} {\alpha'_{w_i}}^{f_t(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)}.$$

Step 4: Now $f_t(w_i)$, $g(w_{i-1}, w_i)$, and $g(w_{i-2}, w_{i-1}, w_i)$ are nested features. Use the formula (3.20) to compute all $z(w_{i-2}, w_{i-1}, t_i)$ for fixed t.

$$\begin{aligned} z(w_{i-2}, w_{i-1}, t_i) &= \sum_{w_i \in V} {\alpha'_{w_i}}^{f_t(w_i)} + \sum_{w_i \in Y_2} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} - 1) \cdot {\alpha'_{w_i}}^{f_t(w_i)} \\ &\quad + \sum_{w_i \in Y_3} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1) {\alpha'_{w_i}}^{f_t(w_i)}. \end{aligned}$$

Step 5: Collect partial feature expectations by topics. Details of this step will be given shortly in Section 3.7.

Analysis of Performance

The number of multiplication in (3.40) in Step 3 is exactly the number of topic features denoted as U_{topic}[9]. The running time in either Step 4 or Step 5 is roughly $O(U + B_k^* + T_k^*)$ for each topic k, where B_k^* is the number of the combinations of words w_{i-1} in topic k and words w_i following w_{i-1} in the training data, and T_k^* is the number of combinations of bigram w_{i-2}, w_{i-1} in topic k and their following words in the training

[9] Assuming that $U_{topic} > U$.

data. It should be noted that B_k^* here and B_k used in non-overlapping sharing and generalized hierarchical training are slightly different, because the subscript k in the former represents the order, whereas it represents the information source in the latter. B_k^* and T_k^* are usually of the same order as the numbers of bigrams and trigrams in topic k respectively. The total time is thus less than $O(U \cdot K + \sum_{t=1}^{K} B_k^* + T_k^*)$ [10] for all topics. This time is about the same as obtaining K individual different empirical estimates for K topics.

This method can also be applied to other models similar to the topic-dependent model (3.39) that have N-gram constraints and one other kind of constraint not nested in N-grams, e.g., the model with part-of-speech tag constraints:

$$P(w_i|w_{i-2}, w_{i-1}, pos_{i-1}) = \frac{\alpha_{w_i}^{g(w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} \cdot \alpha_{pos_{i-1},w_i}^{g(pos_{i-1},w_i)}}{z(w_{i-2}, w_{i-1}, pos_{i-1})},$$

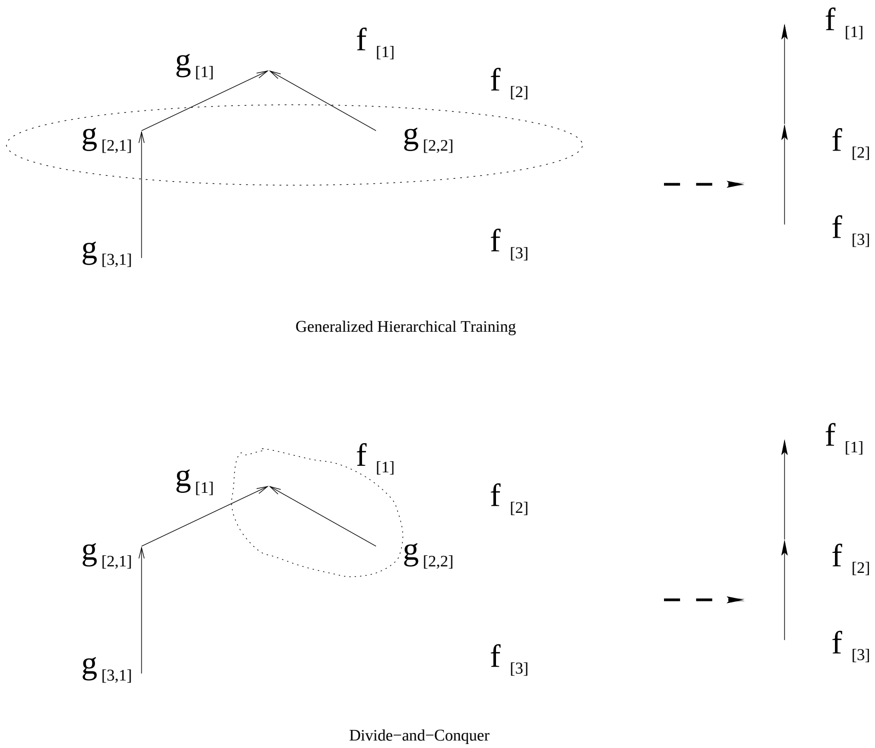
where pos_{i-1} is the part-of-speech tag of the previous word w_{i-1}.

To train this model, we need only to partition the training data according to pos_{i-1} instead of t_i and then follow the steps of training the topic-dependent model. It should be noted that this method cannot be used when the number of partitions is too large, since the overhead time in Step 1, Step 2 and Step 3 would then be considerable.

It is worth mentioning that divide-and-conquer (DC) can be regarded as a variation of GHT. In the generalized hierarchical training, the regular bigram feature and the topic feature are combined into a *bigram super-feature*, while in divide-and-conquer the unigram feature and the topic feature are combined to create a *unigram super-feature*. This slight difference in organizing the computation results in different efficiencies for topic models between divide-and-conquer and the generalized hierarchical training. Figure 3.12 illustrates the differences between these two approaches in hierarchizing the topic model.

In Section 3.8, we will show that both divide-and-conquer and generalized hierarchical training can also be used together to train language models with both topic

[10] Actually no more than $O(U + U_{topic} + \sum_{t=1}^{K} B_k^* + T_k^*)$ because α' needs to be computed only for words with topic constraints in Step 3.



Generalized Hierarchical Training

Divide−and−Conquer

Figure 3.12: Generalized hierarchical training vs. divide-and-conquer

and syntactic constraints.

3.7 Feature Expectation

In previous sections, we focused exclusively on the computation of the normalization factor z in previous sections. In this section, we briefly describe the computation of feature expectations.

Any simplification in computing z in the previous sections can be applied to the computation of feature expectations. We will show, by the example of the topic model (3.39), how to compute the expectation of the N-gram features $g(w_i)$, $g(w_{i-1}, w_i)$, $g(w_{i-2}, w_{i-1}, w_i)$ and the topic feature $g(t_i, w_i)$. The computation of other feature expectations is similar. We will use the computational tricks of hierarchical training and divide-and-conquer in calculating the feature expectation.

The expectation $p[g(w_{i-2}, w_{i-1}, w_i)]$ of trigram features $g(w_{i-2}, w_{i-1}, w_i)$ is relatively easy to compute according to

$$p[g(w_{i-2}, w_{i-1}, w_i)] = \sum_{t_i} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \alpha_{w_i}^{g(w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)}.$$

It is easy to check that the time complexity for trigrams is only $O(\#[w_{i-2}, w_{i-1}, w_i, t_i])$.

Computing the expectation of bigram features $g(w_{i-1}, w_i)$ needs to use the idea of hierarchical training. It should be noted that w_{i-1} and w_i here can be regarded as fixed coefficients instead of variables.

$$\begin{aligned}
&p[g(w_{i-1}, w_i)] \\
&= \sum_{t_i, w_{i-2}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \alpha_{w_i}^{g(w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} \cdot g(w_{i-1}, w_i) \\
&= \sum_{t_i} \left[\sum_{w_{i-2}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \right] \alpha_{w_i}^{g(w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \cdot g(w_{i-1}, w_i) \qquad (3.41) \\
&+ \sum_{t_i, w_{i-2}} \left[\frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \alpha_{w_i}^{g(w_i)} \alpha_{t_i,w_i}^{g(t_i,w_i)} \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} (\alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1) \right] \qquad (3.42)
\end{aligned}$$

$\left[\sum_{w_{i-2}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)}\right]$ in (3.41) can be pre-computed for all bigram features in $O(\#[w_{i-2}, w_{i-1}, t_i])$. The remaining of computation in (3.41) takes roughly $O(\#[w_{i-1}, w_i, t_i])$ time for all bigrams. For all bigrams w_{i-1}, w_i, the computation in (3.42) takes $O(\sum_{k=1}^{K} T_k^*)$ where K is the number of topics and T_k^*, as defined in the previous section, is the number of combinations of bigram w_{i-1}, w_i in topic k and their preceding word w_{i-2} in the training data. The overall complexity is the same as computing z for all histories.

Computing the expectation of unigram features $g(w_i)$ may take very long time if improperly implemented. We use a similar computational trick to that we have used in computing the normalization factor z.

$$\begin{aligned}
& p[g(w_i)] \\
&= \sum_{t_i} \sum_{w_{i-2},w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)} \alpha_{w_i}^{(w)} \cdot \alpha_{t_i,w_i}^{(t_i,w_i)} \cdot \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} && (3.43) \\
&= \sum_{t_i} \left[\sum_{w_{i-2},w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)} \right] \alpha_{w_i}^{g(w_i)} \cdot \alpha_{t_i,w_i}^{g(t_i,w_i)} && (3.44) \\
&+ \sum_{t_i} \sum_{w_{i-2}} \left[\sum_{w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)} \right] \alpha_{w_i}^{g(w_i)} \alpha_{t_i,w_i}^{g(t_i,w_i)} (\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} - 1) && (3.45) \\
&+ \sum_{t_i,w_{i-2},w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)} \alpha_{w_i}^{g(w_i)} \alpha_{t_i,w_i}^{g(t_i,w_i)} \alpha_{w_{i-1},w}^{g(w_{i-1},w_i)} (\alpha_{w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} - 1) && (3.46)
\end{aligned}$$

Formula (3.43) shows the straight-forward implementation, and (3.44)-(3.46) illustrate the improved version. In particular, Formula (3.44) takes advantage of unigram-caching, while (3.45) and (3.46) take advantage of hierarchical training. It should be noted that

- $\sum_{w_{i-2},w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)}$ in (3.44) is fixed given $t_i = k$, so it can be computed once in $O(|X_k|)$ and then reused for unigram features $g(w_i)$ within the same $t_i = k$, where $|X_k|$ is the number of histories in topic k;
- $\sum_{w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)}$ in (3.45) is a byproduct of $\sum_{w_{i-2},w_{i-1}} \frac{\hat{p}(w_{i-2},w_{i-1},t_i)}{z(w_{i-2},w_{i-1},t_i)}$ and so needs no extra computation, and the number of terms to be summed for all $g(w_i)$ is $\sum_{k=1}^{K} B_k$ where B_k, as defined in the previous section, is the number of the combinations of words w_i in topic k and words w_{i-1} preceding it in the training data; and
- in (3.46), the number of terms to be summed for all $g(w_i)$ is $\sum_{k=1}^{K} T_k^*$.

Therefore, the overall time complexity is roughly $O(\sum_{k=1}^{K} B_k^* + \sum_{k=1}^{K} T_k^*)$, which, as shown in Section 3.6, is the same as computing the normalization factor z for all histories in the training data.

The expectation of the topic feature $g(t_i, w_i)$ is computed similarly to that of unigrams, but t_i here is a fixed coefficient instead of a variable.

$$
\begin{aligned}
& p[g(t_i, w_i)] \\
= & \sum_{w_{i-2}, w_{i-1}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \alpha_{w_i}^{g(w_i)} \alpha_{w_{i-1}, w}^{g(w_{i-1}, w_i)} \alpha_{w_{i-1}, w_i}^{g(w_{i-2}, w_{i-1}, w_i)} \alpha_{t_i, w_i}^{g(t_i, w_i)} \\
= & \left[\sum_{w_{i-2}, w_{i-1}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \right] \alpha_{w_i}^{g(w_i)} \alpha_{t_i, w_i}^{g(t_i, w_i)} && (3.47) \\
+ & \sum_{w_{i-1}} \left[\sum_{w_{i-1}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t)} \right] \alpha_{w_i}^{g(w_i)} \alpha_{t_i, w_i}^{g(t_i, w_i)} (\alpha_{w_{i-1}, w_i}^{g(w_{i-1}, w_i)} - 1) && (3.48) \\
+ & \sum_{w_{i-2}, w_{i-1}} \frac{\hat{p}(w_{i-2}, w_{i-1}, t_i)}{z(w_{i-2}, w_{i-1}, t_i)} \alpha_{w_i}^{g(w_i)} \alpha_{t_i, w_i}^{g(t_i, w_i)} \alpha_{w_{i-1}, w}^{g(w_{i-1}, w_i)} (\alpha_{w_{i-2}, w_{i-1}, w_i}^{g(w_{i-2}, w_{i-1}, w_i)} - 1) && (3.49)
\end{aligned}
$$

All summations in (3.47), (3.48) and (3.49) can be obtained for free when computing $p[g(w_i)]$. It can be easily checked that the time of computing all feature expectations in the topic model (3.39) is the same as that of computing z for all seen histories as shown in Section 3.6.

3.8 Examples: The Training of Several Practical ME Models with Non-nested Features

We implement three kinds of ME models with non-nested features in this dissertation: the topic-dependent model, the syntactic model and the composite model with both topic and syntactic dependencies. The training of topic models has been described in the previous section. In this section we provide some details of training the syntactic model and the composite model.

3.8.1 Training ME Models with Syntactic Constraints

We can obtain meaningful syntactic information via parsing the sentence. For each word we can use its last two exposed head words (h_{i-2}, h_{i-1}) and their non-terminal labels $(nt_{i-2},\ nt_{i-1})$ of the partial parse T as predictors. We use them in combination with the immediate history $(w_{i-2},\ w_{i-1})$ to predict w_i. We will introduce details on how to extract syntactic heads and show why they are helpful in language modeling in Chapter 5. The syntactic heads are represented in the same form as N-grams. We will build a syntactic model with the form

$$\begin{aligned} & p(w_i|w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1}) \qquad (3.50) \\ = & \frac{\alpha_{w_i}^{g(w_i)} \alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)} \alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)} \alpha_{h_{i-1},w_i}^{g(h_{i-1},w_i)} \alpha_{h_{i-2},h_{i-1},w_i}^{g(h_{i-2},h_{i-1},w_i)} \alpha_{nt_{i-1},w_i}^{g(nt_{i-1},w_i)} \alpha_{nt_{i-2},nt_{i-1},w_i}^{g(nt_{i-2},nt_{i-1},w_i)}}{z(w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1})}. \end{aligned}$$

This is a 33-gram model. The hierarchical structure of the syntactic model is shown in Figure 3.13.

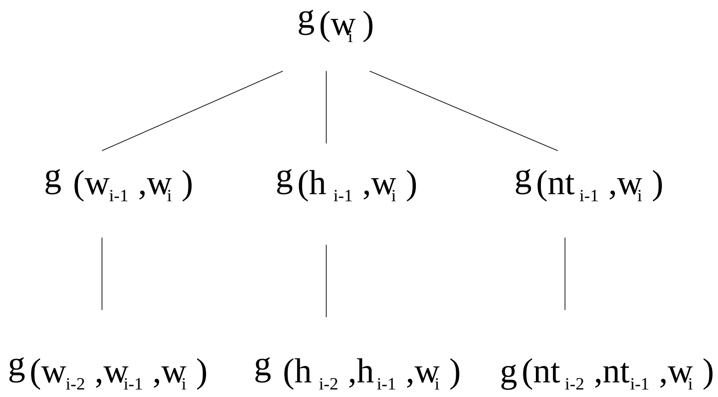


Figure 3.13: Structure of syntactic model

We group three order 2 features $g(w_{i-1}, w_i)$, $g(h_{i-1}, w_i)$ and $g(nt_{i-1}, w_i)$ in an order 2 super-feature $f_2(w_{i-1}, h_{i-1}, nt_{i-1}, w_i)$, three order 3 features $g(w_{i-2}, w_{i-1}, w_i)$, $g(h_{i-2}, h_{i-1}, w_i)$ and $g(nt_{i-2}, nt_{i-1}, w_i)$ in the super-feature $f_3(w_{i-2}, w_{i-1},\ h_{i-2}, h_{i-1},$ $nt_{i-2}, nt_{i-1}, w_i)$, and set $f_1(w_i) = g(w_i)$. Thus the syntactic model (3.50) can be rewritten as

$$\begin{aligned} & P(w_i|w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1}) \qquad (3.51) \\ = & \frac{\beta_1^{f_1(w_i)} \cdot \beta_2^{f_2(w_{i-1},h_{i-1},nt_{i-1},w_i)} \cdot \beta_3^{f_3(w_{i-2},w_{i-1},h_{i-2},h_{i-1},nt_{i-2},nt_{i-1},w_i)}}{z(w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1})} \end{aligned}$$

where

$$\begin{aligned}\beta_1 = \beta_{w_i} &= \alpha_{w_i} \\ \beta_2 = \beta_{w_{i-1},h_{i-1},nt_{i-1},w_i} &= \alpha_{w_{i-1},w_i}\alpha_{h_{i-1},w_i}\alpha_{nt_{i-1},w_i} \\ \beta_3 = \beta_{w_{i-2},w_{i-1},h_{i-2},h_{i-1},nt_{i-2},nt_{i-1},w_i} &= \alpha_{w_{i-2},w_{i-1},w_i}\alpha_{h_{i-2},h_{i-1},w_i}\alpha_{nt_{i-2},nt_{i-1},w_i}\end{aligned}$$

and can be trained by the generalized hierarchical training method.

We also split the computation for bigram features to 2^3 parts according the method for overlapping features in Section 3.4.[11] However, we do not split the computation for trigram features for two reasons. First the straight-forward implementation is already quite efficient since Y_3 contains only a few words for each history. Furthermore, even though using the method in Section 3.4 may still reduce the computational load slightly, it requires an inordinate amount of space in experiments to save the intermediate results.

3.8.2 Training Composite ME Model with Both Topic and Syntactic Dependencies

Finally, we combine both topic dependencies and syntactic-structure dependencies with N-grams in one language model:

$$p(w_i|w_1^{i-1}) \approx p(w_i|w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1}, t) \tag{3.52}$$

where t_i is a deterministic function of the history w_1^{i-1}.

We build the following ME model to estimate the probability above.

$$\begin{aligned}&p(w_i|w_{i-1}, w_{i-2}, h_{i-1}, h_{i-2}, nt_{i-2}, nt_{i-1}, t_i) \\ &= \frac{\alpha_{w_i}^{g(w_i)}\alpha_{w_{i-1},w_i}^{g(w_{i-1},w_i)}\alpha_{w_{i-2},w_{i-1},w_i}^{g(w_{i-2},w_{i-1},w_i)}\alpha_{h_{i-1},w_i}^{g(h_{i-1},w_i)}\alpha_{h_{i-2},h_{i-1},w_i}^{(h_{i-2},h_{i-1},w_i)}\alpha_{nt_{i-1},w_i}^{g(nt_{i-1},w_i}\alpha_{nt_{i-2},nt_{i-1},w_i}^{g(nt_{i-2},nt_{i-1},w_i)}\alpha_{t_i,w_i}^{g(t_i,w_i)}}{z(w_{i-2}, w_{i-1}, h_{i-2}, h_{i-1}, nt_{i-2}, nt_{i-1}, t_i)}.\end{aligned}$$

All our training simplification methods described in the preceding sections can be used in training this model. Since it is a topic-dependent model, we can use a divide-and-conquer scheme. We collect N-gram and syntactic statistics from different topics

[11] This brings about a twofold speed-up.

and merge $g(w_i)$ with $g(t_i, w_i)$. Now, within each topic, it looks like the syntactic model (3.50). Therefore, we can train this model as we did the syntactic model by using the GHT method.

3.9 Experimental Results for Speed-up

The efficiency of training methods is evaluated by the Switchboard and the Broadcast News data. We use 2.1 million words in the form of 1100 conversations on about 70 different topics in the Switchboard corpus. The Broadcast News corpus contain 125,000 stories amounting to about 130 million words that are clustered into 100 topics. Table 3.5 provides some numbers to show the complexity of the Switchboard and the Broadcast News tasks.

		Switchboard	Broadcast News
General Info	Vocabulary Size	22,000	64,000
	Training Size	2.1M	130M
	Topics	70	100
Trigram Model	N-grams	(22+300+180)K	64K+3.5M+5.6M
	Histories	300K	7.6M
	Tuples	830K	36M
Topic Model	Topic unigram	+15K	+250K
	Histories	770K	29M
	Tuples	1.4M	73M
Syntactic Model [13]	Syntactic N-gram	300K	2.6M [12]
	Histories	760K	14M
	Tuples	2.5M	27M
Composite Model	Histories	8.4M	23M[14]
	Tuples	10M	42M

Table 3.5: Comparison of Switchboard and Broadcast News.

[12] The syntactic model for Broadcast News is trained only on a subset of 14 million words of the corpus.

[13] Training size for syntactic models expands as multiple candidate parses are considered for each sentence.

[14] Many low count histories have been merged into history equivalence classes.

3.9.1 Nominal Speed-up vs. Real Speed-up

The training time is roughly proportional to the number of terms to be summed up for all z's. We count this number and estimate the running time of a method according to it. The nominal speed-up is defined as

$$\text{nominal speed-up} = \frac{\#[\text{terms in the baseline method}]}{\#[\text{terms in the new method}]}, \tag{3.53}$$

where the baseline method is the Improved Iterative Scaling with unigram caching.

We measure the training time of four kinds of maximum entropy models: the trigram model, the topic dependent trigram model, the syntactic model and the composite model for the Switchboard and Broadcast News tasks. The real running time speed-up is the ratio of the training time of unigram-caching and that of the improved methods. Since some models cannot be trained at all by the baseline method, we do not know the real speed-up of them. The benchmark test for speed is based on 300MHz Ultra-Sparc II 64-bit systems with 1GB memory[15]. All numbers shown in this section are in CPU-hours per iteration.

3.9.2 Hierarchical Training for N-gram Models

The computational loads of straight-forward implementation and unigram-caching are first estimated for comparison. Table 3.6 illustrates that unigram-caching already reduces the computational load by about two orders of magnitude compared to the naive implementation. Therefore, we treat unigram-caching as the baseline method in our evaluation and compare it with the hierarchical training method.

The results for trigram models are illustrated in Table 3.6. Columns 3, 4 and 5 provide the number of terms in the straight-forward implementation, the baseline method and the hierarchical method, respectively. The last column is the nominal speed-up of (3.53). It is apparent that the hierarchical training method achieves a nominal speed-up of more than two orders of magnitude for both corpora, and the gain is greater as the size of the model increases.

[15] These machines were the fastest ones available when we started the work in this dissertation four years ago. Readers may keep in mind that the real training time reported in this dissection can be reduced by two-threefold using faster machines now available.

LM Task	Model Size	ME Algorithms			Nominal Speed-up
		Naive	Baseline	Hierarchical	
SWBD	500K	$6 \cdot 10^9$	$1.6 \cdot 10^8$	$9 \cdot 10^5$	170
BN	9.1M	$3.6 \cdot 10^{11}$	$2.4 \cdot 10^{10}$	$4.3 \cdot 10^7$	560

Table 3.6: Number of operations and nominal speed-up of trigram models

We also estimate the nominal speed-up of four-gram models (in Table 3.7).

LM Task	Model Size	ME Algorithms		Nominal Speed-up
		Baseline	Hierarchical	
SWBD	1.6M	$3.2 \cdot 10^9$	$2.1 \cdot 10^6$	1600
BN	13M	$3.4 \cdot 10^{12}$	$9.4 \cdot 10^7$	$3.6 \cdot 10^4$

Table 3.7: Number of operations and nominal speed-up of four-gram models

The computational load of training four-gram models using unigram-caching increases tremendously compared to that of training trigram models (20-120 fold), but it only increases slightly if hierarchical training methods are used (in about twofold). The nominal speed-up for four-gram models is much greater than that for trigram models.

The real training time of using different methods is also measured based on trigram models (Table 3.8). To train an ME trigram model for Switchboard, the baseline method takes about two CPU-hours, while the hierarchical training method needs only four CPU-minutes (with a speed-up of 30 fold)[16]. The speed-up for Broadcast News (85 fold) is greater than that for Switchboard. It is worth mentioning that the running time of less than one hour per iteration for such a huge corpus is quite fast and is even comparable to the time of training a back-off model[17].

[16] Real running time speed-up is less than the nominal one because of the overhead time, and I/O time are fixed no matter what training method is used.

[17] Using SRI LM toolkit V0.98.

LM Task	ME Algorithms		Real Speed-up
	Baseline	Hierarchical	
SWBD	2	0.06	30
BN	60	0.7	85

Table 3.8: Running time (in CPU-Hours) for ME trigram models

3.9.3 Generalized Hierarchical Training for Syntactic Model

The efficiency of generalized hierarchical training method is evaluated by syntactic models, which have both nested features and overlapping features. The nominal speed-up is shown in Table 3.9. For Broadcast News, we only provide statistics for a subset of the corpus because parsing the whole corpus of 130M words and storing the parsing results challenge our computer speed and storage capacity.

LM Task	ME Algorithms			Nominal Speed-up
	Baseline	Cluster Expansion	Hierarchical	
Switchboard	$6.9 \cdot 10^9$	$4.6 \cdot 10^9$	$7.5 \cdot 10^7$	$9 \cdot 10^1$
Broadcast News (14M)	$6.1 \cdot 10^{11}$	N/A	$4.2 \cdot 10^9$	$1.4 \cdot 10^2$

Table 3.9: Number of operations and nominal speed-up for syntactic models

The computational load increases considerably for the syntactic model (and will be even more for the composite model) compared to trigram models. The cluster expansion method described in Lafferty & Suhm (1996) is of little help here since the number of interacting features is so large. The hierarchical training will reduce the computational load by 90 fold in Switchboard and 140 fold in Broadcast News.

In Table 3.10, we compare the real running time of training syntactic models using the generalized hierarchical training method and that using the baseline method. We do not provide the baseline training time (without the speed-up) for Broadcast New, because it is impractical to train the syntactic model on such a large corpus using unigram-caching. It is apparent from the results in Switchboard that the hierarchical method achieves a considerable speed-up. Nevertheless, we may expect the speed-up in Broadcast News to be even greater than in Switchboard.

LM Task	ME Algorithms		Real Speed-up
	W/O Hierarchical	W/ Hierarchical	
Switchboard	100	6	17
Broadcast News	N/A	480 [18]	N/A

Table 3.10: Training time (in CPU-hours) for syntactic models

3.9.4 Divide-and-Conquer and Topic-Dependent Model

Divide-and-conquer is designed for and evaluated on topic-dependent models (Table 3.11). Comparing the trigram model (3.6) with the topic-dependent model (3.39), one can see that the number of operations in the baseline method will increase by about two orders of magnitude even though the model size increases by only 3%. This load can be decreased in about one order of magnitude by the generalized hierarchical training. However, this computational load is still heavy even for the Switchboard. Divide-and-conquer strategy individually will cut the computational load in Switchboard down to a manageable scale. Divide-and-conquer and hierarchical training can be applied together to make the training of the topic-dependent model practical for Broadcast News. Overall, the nominal speed-up for the topic-dependent model is about 400 fold for the Switchboard and is more than 1000 fold for the Broadcast News, if both divide-and-conquer and hierarchical training are used.

LM Task	ME Algorithms				Nominal Speed-up
	Baseline	Hierarchical	Divide-Conquer	H+DC	
SWBD	$1.6 \cdot 10^9$	$1.9 \cdot 10^8$	$6 \cdot 10^7$	$4.1 \cdot 10^6$	$4 \cdot 10^2$
BN	$1.8 \cdot 10^{11}$	$5.1 \cdot 10^9$	$6.6 \cdot 10^9$	$1.5 \cdot 10^8$	$1.3 \cdot 10^3$

Table 3.11: Number of operations and nominal speed-up for topic models

[18] The model is trained on 1.2GHz PIII machines in 180 $\sim$ 200 CPU-hours per iteration. One iteration was run on Sun Sparc machines to compare for the training speed with those of other models.

The training of a topic dependent model without any speed-up takes a very long time (160 CPU-hours) even for Switchboard and it is impractical for Broadcast News. However, it needs only 0.5 and 2.3 CPU-hours, respectively, to train the topic models in the corpora above. The real running time speed-up for Switchboard is more than two orders of magnitude based on unigram-caching.

We do not report the baseline training time for Broadcast News because unigram-caching is not practical here. However, we can expect that the real speed-up in Broadcast News is greater than that in Switchboard. We roughly estimate the training time of using generalized hierarchical training and divide-and-conquer individually by using about 5% of the data and multiplying the resulting training time by 20. The estimated time (numbers inside parentheses) is shown in Table 3.12.

It is also worth mentioning that the real running time of 2.3 CPU-hours per iteration for the topic model with 100 topics is quite fast and is about the same as that of training the corresponding interpolation topic models[19]. Table 3.12 shows detailed results.

LM	ME Algorithms				Real
Task	Baseline	Hierarchical	Divide-and-conquer	Both	Speed-up
SWBD	165	21	8.3	0.5	$3.3 \cdot 10^2$
BN	-	(85)	(100)	2.3	-

Table 3.12: Running time (in CPU-Hours) for topic models

3.9.5 Divide-and-Conquer Combined with Generalized Hierarchical Training

If the topic model can still be trained using either divide-and-conquer or generalized hierarchical training, the composite model with both the topic and the syntactic constraints must be trained using both of them. We train the composite model in less than 10 CPU-hours per iteration in Switchboard. We can train this model without

[19] Using SRI LM toolkit V0.98 by Andreas Stolcke.

Task &	ME Algorithms		Real
Model	DC, W/O Hierarchical	DC + Hierarchical	Speed-up
SWBD Composite	150	9.5	15

Table 3.13: Training time for composite model in Switchboard

generalized hierarchical training method in a much longer time (150 CPU-hours) but cannot train this model without divide-and-conquer even in Switchboard. Therefore, we only show the gain from the generalized hierarchical training method in Table 3.13.

We also implement a composite model on 14M Broadcast News data using both speed-up methods above. The training time is roughly 400 CPU-hours per iteration, which is quite long. However, this model is impractical to train without our speed-up methods.

3.10 Summary

Several rapid training methods for maximum entropy models have been presented in this chapter. In particular, N-gram models are trained hierarchically and almost as efficiently as back-off models, whose training time is just linear in the size of training data. Experimental results on Switchboard and Broadcast News show that computational load and real training time reduce tremendously by hierarchical training methods. The speed-up of a factor of tens has been achieved compared to the baseline unigram-caching method.

Other models with overlapping features, e.g., syntactic models, can also benefit from the hierarchizing of features and can thus be trained effectively by the generalized hierarchical training. The time of training the syntactic model and the composite model (with both the topic and syntactic constraints) reduces by more than one order of magnitude after this method is used.

Divide-and-conquer can be used in addition to the generalized hierarchical training method in any model with topic constraints. The combination of these two methods makes it possible to train large topic-dependent models and composite models that otherwise are intractable to train.

It is worth mentioning here that the computational load of training ME models can be distributed among many computers using the idea of divide-and-conquer. The speed-up is almost linear in the number of computers.