Chapter 7

Improving the Computation when Using ME Models

We have discussed how to simplify the computation in training maximum entropy models in Chapter 3. There is another problem that may prevent one from using ME models in a real speech recognition system: the amount of computation required to compute the language model probabilities when recognition. In this chapter, we will show how to simplify the computation in using ME models by three techniques: pre-normalization, approximation and history-caching.

The straightforward way of calculating the probability \( p(y|x) \) for any \( (x, y) \) pair using an ME model is via the following formula

\[
p(y|x) = \frac{1}{z(x)} \prod_{k=1}^{K} \alpha_k^{g_k(x, y; A_k)},
\]

\[
z(x) = \sum_{y \in \mathcal{V}} \prod_{k=1}^{K} \alpha_k^{g_k(x, y; A_k)}
\]

where \( g_k \) is the \( k \)th feature in the feature set and \( \alpha_k \) is its parameter. The computation for the numerator is simple. It requires a table look-up and a few multiplications of \( \alpha \)'s. However, computing the denominator \( z \) takes time, since \( z \) is the sum of hundreds or even thousands of terms\(^1\). Since histories in the test data cannot be organized

\(^1\) Unigram-caching is applied; otherwise, computing \( z \) needs a summation over all words in the vocabulary.
hierarchically as they do in the training data, the speed-up methods introduced in Chapter 3 are not applicable. It is also not practical to pre-compute the normalization factor $z$ for all histories since, as we have shown, the size of the history set is huge. As we will show later, it is also not necessary to do so.

We want to make ME models easy to use. We can reduce the computation in ME models in the following ways. For N-gram models, we pre-normalize them and convert them to the ARPA back-off format, a popular language model format that will be introduced later. N-gram models can thus be used as simply as back-off models. For topic models, we approximate their denominators $z$ so that they can be estimated without much computation in speech recognition. For the rest of the models, we additionally apply some caching techniques to save the computation.

### 7.1 Converting ME Models to ARPA Format

ARPA format is the most popular language model format for back-off models in the speech recognition world. It is accepted by many speech recognition decoders such as HTK, and can be converted to the LM format of the popular AT&T finite state machine (FSM) decoder. ME models can be used in these recognizers so long as they can be converted to the ARPA format. In this section, we show how to transform ME models from their exponential form to the ARPA (back-off) form.

#### 7.1.1 ARPA Format for Back-off Models

First, we briefly introduce how a back-off model is stored in the ARPA format. If $f(w_N|w_1, \ldots, w_{N-1})$ is the relative frequency after some smoothing and $T_N$ be a pre-set cut-off for N-grams, then the back-off N-gram model can be written as

$$p(w_N|w_1, \ldots, w_{N-1}) = \begin{cases} f(w_N|w_1, \ldots, w_{N-1}) & \text{if } \#[w_1, \ldots, w_N] \geq T_N, \\ \text{bow}(w_1, \ldots, w_{N-1})p(w_N|w_2, \ldots, w_{N-1}) & \text{otherwise,} \end{cases}$$

(7.1)
where the back-off weight $\text{bow}^2$ is the remaining probability mass for the all $w_N$ whose
$\#[w_1, \cdots, w_N] < T_N$. ARPA back-off models store all N-grams with their parameter
values $f$ and $\text{bow}$ in one ASCII file. Table 7.1 shows the ARPA file format for back-off
N-gram models (7.1).

|\data\ |
|---|---|
|ngram | 1= number-of-unigrams |
|ngram | 2= number-of-bigrams |
|\vdots |
|ngram | N= number of N-grams |

\begin{align*}
\text{1-grams:} & \quad f(w) \quad w \quad [\text{bow}(w)] \\
\text{2-grams:} & \quad f(w_2|w_1) \quad w_1, w_2 \quad [\text{bow}(w_1 w_2)] \\
\vdots \\
\text{N-grams:} & \quad f(w_N|w_1, \cdots, w_{N-1}) \quad w_1, \cdots, w_N \\
\vdots \\
\end{align*}

Table 7.1: ARPA format for back-off models.

The file is comprised of three segments: the header, the body and the tail. The
header starts with the keyword “\data\” and lists the number of n-grams for $n = 1, \cdots, N$. The body of the model file consists of n-grams (one per line) grouped by
length $n$. Each group begins with the keyword “\n-gram:”, where $n$ is the length of
the n-grams that follow. Each line has either two or three columns: the logarithm
(base 10) of the n-gram probability, the n-word tuple in ASCII and the logarithm
\footnote{Actually back-off weight bow’s are not independent - they can be derived from f’s.}
(base 10) of its back-off weight. The third column can be omitted if the back-off weight is zero\(^3\). The tail of the file has only one line in which the keyword “\`end\`” concludes the model.

If an N-gram \((w_1, w_2, \cdots, w_N)\) is found in the model, then \(p(w_N|w_1, \cdots, w_{N-1})\) is directly obtained by a table look-up, otherwise \(p(w_N|w_1, \cdots, w_{N-1})\) is obtained by \(\text{bow}(w_1, \cdots, w_{N-1}) \cdot p(w_N|w_2, \cdots, w_{N-1})\) where \(p(w_N|w_2, \cdots, w_{N-1})\) is computed recursively in the same manner.

### 7.1.2 Mapping the ME N-gram Model Parameters to ARPA Back-off Model Parameters

In a (recursive) back-off model \(p(w_N|w_1, \cdots, w_{N-1})\), the relative frequency \(f(w_N|w_1, \cdots, w_{N-1})\) is a scale function for each word \(i.e., w_N\) given the history \(w_1, \cdots, w_{N-1}\).

If the history \(w_1, \cdots, w_{N-1}\) is in the model (\(i.e.,\) has been seen in the training data), the probability \(p\) is obtained directly from this scale function \(f\) since it is already normalized. If the history is not in the model, then the lower order scale function \(f(w_N|w_2, \cdots, w_{N-1})\) is used instead, but the total probability mass becomes \(\text{bow}(w_2, \cdots, w_{N-1})\) instead of \(1\) now. Both \(f(w_N|w_2, \cdots, w_{N-1})\) and \(\text{bow}(w_2, \cdots, w_{N-1})\) are available during the training procedure.

In the corresponding ME model

\[
p(w_N|w_1, \cdots, w_{N-1}) = \frac{\prod_{i=1}^{N} \alpha^{g(w_{N-i+1}, \cdots, w_N)}}{z(w_1, \cdots, w_{N-1})},
\]

the numerator is the scale function and plays a role similar to that of \(f\) in the back-off model. However, the total mass \(z\) is not explicitly given by the model parameters \(\alpha\). Nevertheless, in the training procedure for ME models, \(z\) for all observed histories have already been computed. For the enormous number of unseen histories, their normalization factors can also be computed indirectly during the last iteration of training. Actually, if \(w_1, \cdots, w_{N-1}\) is not seen in the training data, the N-gram

\(^3\text{bow is zero for the highest order N-gram. It may also be zero for some lower order n-grams if they are not a prefix of longer n-grams.}\)
feature function \( g(w_1, \cdots, w_{N-1}, w_N) \) can never be activated for any \( w_N \), therefore,

\[
z(w_1, \cdots, w_{N-1}) = \sum_{w_N \in V} \prod_{i=1}^{N-1} \alpha_{w_N}^{g(w_{N-i+1}, \cdots, w_N)}
\]

is independent of \( w_1 \), or, in other words, it is a function of the history equivalence class \( w_2, \cdots, w_{N-1} \), i.e., \( z = z(w_2, \cdots, w_{N-1}) \). Further, \( z(w_2, \cdots, w_{N-1}) \) is available during the hierarchical training if \( w_2, \cdots, w_{N-1} \) is an observed history class; otherwise \( z \) is obtained from lower order history classes \( z(w_3, \cdots, w_{N-1}) \) etc.

We want to write the ME N-gram model in the form of the back-off model, i.e.,

\[
p_{ME}(w_N|w_1, \cdots, w_{N-1}) = \text{bow}_{ME}(w_1, \cdots, w_{N-1}) \cdot f_{ME}(w_1, \cdots, w_{N-1}, w_N) \tag{7.3}
\]

and derive \( \text{bow}_{ME} \) and \( f_{ME} \) from \( \alpha \)'s.

We focus on the trigram model

\[
p(w|u, v) = \frac{\alpha_{w}^{g(w)} \cdot \alpha_{v}^{g(v, w)} \cdot \alpha_{u,v,w}^{g(u,v,w)}}{z(u, v)}
\]

where \( u, v \) are two preceding words for \( w \) for illustration. Setting

\[
\zeta = \sum_{w} \alpha_{w}^{g(w)}, \tag{7.4}
\]

\[
\zeta(v) = \sum_{w} \alpha_{w}^{g(w)} \cdot \alpha_{v}^{g(v, w)}, \tag{7.5}
\]

\[
\zeta(u, v) = \sum_{w} \alpha_{w}^{g(w)} \cdot \alpha_{v}^{g(v, w)} \cdot \alpha_{u,v,w}^{g(u,v,w)}, \tag{7.6}
\]

\[
f_{ME}(w|u, v) = \frac{\alpha_{w}^{g(w)} \cdot \alpha_{v}^{g(v, w)} \cdot \alpha_{u,v,w}^{g(u,v,w)}}{\zeta(u, v)},
\]

\[
\text{bow}_{ME}(u, v) = \frac{\zeta(v)}{\zeta(u, v)},
\]

\[
f_{ME}(w|v) = \frac{\alpha_{w}^{g(w)} \cdot \alpha_{v}^{g(v, w)}}{\zeta(v)},
\]

\[
\text{bow}_{ME}(v) = \frac{\zeta}{\zeta(v)},
\]

\[
f_{ME}(w) = \frac{\alpha_{w}^{g(w)}}{\zeta}.
\]
we get
\[ p(w|u, v) = \frac{\alpha_g(w) \cdot \alpha_{g}^g(v, w) \cdot \alpha_{u, v, w}}{z(u, v)} = \text{bow}_{ME}(u, v) \cdot f_{ME}(w|u, v). \]

Replacing \( f \) and \( \text{bow} \) in the ARPA back-off model (in Table 7.1) by \( f_{ME} \)'s and \( \text{bow}_{ME} \)'s, we obtain an ME trigram model with ARPA back-off format.

For any ME N-gram model (7.2), we can pre-normalize it in the same way. Setting
\[
\zeta = \sum_w \alpha^g(w),
\]
\[
\zeta(w_1, \cdots, w_{n-1}) = \sum_{w_n} \prod_{k=1}^{n} \alpha^g(w_1, \cdots, w_n) \text{ for } n = 1, 2, \ldots, N - 1,
\]
\[
f(w_n|w_1, \cdots, w_{n-1}) = \prod_{k=1}^{n} \frac{\alpha^g(w_1, \cdots, w_{n})}{\zeta(w_1, \cdots, w_{n-1})} \text{ for } n = 1, 2, \ldots, N - 1,
\]
\[
\text{bow}(w_1, \cdots, w_{n-1}) = \frac{\zeta(w_1, \cdots, w_n)}{\zeta(w_1, \cdots, w_{n-1})} \text{ for } n = 1, \ldots, N - 1,
\]
we rewrite (7.2) as (7.3), which has the back-off form.

### 7.1.3 Speed-up

We compare the time of resoring 100-best hypotheses of the test data for Switchboard and Broadcast News, using the original maximum entropy model and the pre-normalized one (in Table 7.2). We use the ME trigram model described in Section 4.3 for Switchboard experiments here and the model in Section 4.5 for Broadcast News. We do not count the overhead time of loading the models in our experiments. Column 1 in the table indicates the test sets with their size (in number of utterances). The numbers in the second and the third columns represent CPU-mins used in resoring the 100-best hypotheses using the original model and the pre-normalized model, respectively. The speed-up of the pre-normalized model, the ratio of Column 2 and Column 3, is shown in Column 4. It is apparently that the pre-normalization decreases the computation time tremendously (about 480 fold for Switchboard and 1000 fold for Broadcast News) because the calculation of \( z \) is omitted.
<table>
<thead>
<tr>
<th>Task (# of Utts.)</th>
<th>Original</th>
<th>Pre-normalized</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swbd (2400*100)</td>
<td>960</td>
<td>2</td>
<td>4.8 \cdot 10^2</td>
</tr>
<tr>
<td>BN (1300*100)</td>
<td>2077</td>
<td>2</td>
<td>1.0 \cdot 10^3</td>
</tr>
</tbody>
</table>

Table 7.2: Speed-up of the pre-nominalized model.

7.1.4 Using the ME Trigram Model in the First Pass of Speech Recognition

We transformed the ME trigram model from the exponential form to the ARPA format in the previous section. Now we create FSMs using both the back-off model and the ME model, respectively. Then we recognize the test speech data of Switchboard-I\(^4\) and Switchboard-II\(^5\) using the above two language models and the same acoustic model by the AT&T FSM decoder.

The acoustic model in 2001 CLSP evaluation system (Byrne, 2001) is used here. The acoustic training set used contains 150 hours of Switchboard I and 15 hours of CallHome transcribed acoustic data. The acoustic features are 12 dimensional PLP-Cepstral coefficients with an energy term, and their first and second derivatives (39 coefficients per frame). Cepstral mean normalization, vocal tract normalization and variance normalization are performed for each conversation side on both the training and the test data. The models are continuous mixture density, tied state, cross-word, context-dependent word-boundary triphones based on the HTK toolkit.

Table 7.3 shows the perplexity and word error rate on these two Switchboard test sets using the back-off model and the ME model. The maximum entropy model achieves slightly better performance than the corresponding back-off models. However, it remains for arguments in favor of these arguments of these experiments to show the possibility of using ME models in the first pass speech recognition.

\(^4\)The same kind of Switchboard data as we used in the experiments in the preceding chapters.
\(^5\)Speech data collected in the same way as Switchboard-I but with poor quality in acoustics.
<table>
<thead>
<tr>
<th></th>
<th>SWBD-1 (00-test)</th>
<th>SWBD-2 (98-test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PPL</td>
<td>WER</td>
</tr>
<tr>
<td>BO</td>
<td>125</td>
<td>32.9%</td>
</tr>
<tr>
<td>ME</td>
<td>123</td>
<td>32.7%</td>
</tr>
</tbody>
</table>

Table 7.3: Perplexity and WER of using the BO and ME models in the first pass recognition.

7.2 Approximating Models with Topic Features

The major purpose of using maximum entropy methods is not to build N-gram models, but to build models with some non-nested features, e.g., the topic model. There are no corresponding back-off models to which topic-dependent models can be mapped. Therefore the topic-dependent model cannot be pre-normalized and its normalization factors must be computed when the model is used. However, we can approximate the normalization factor $z$ of the topic-dependent model so that it can be easily estimated. The basic idea is to apply the computational tricks introduced in Chapter 3 to rewrite $z$ to a history-independent constant plus some terms that can be pre-computed. We show this approximation method using the topic-dependent model

$$p(w | u, v, t) = \frac{\alpha^g_w \cdot \alpha^g_{v,w} \cdot \alpha^g_{u,v,w} \cdot \alpha^g_{t,w}}{z(u, v, t)},$$

$$z(u, v, t) = \sum_{w \in V} \alpha^g_w \cdot \alpha^g_{v,w} \cdot \alpha^g_{u,v,w} \cdot \alpha^g_{t,w},$$

where $t$ is the topic of the current utterance, $u, v$ are the two immediately preceding words of $w$. We apply the computational trick in Chapter 3 and rewrite $z(u, v, t)$ as

$$z(u, v, t) = - \sum_{w \in V} \alpha^g_w + \sum_{w \in Y_u} \alpha^g_w \cdot \alpha^g_{v,w} \cdot \alpha^g_{u,v,w} + \sum_{w \in Y_t} \alpha^g_w \cdot \alpha^g_{t,w} + \sum_{w \in Y_w \cap Y_t} \alpha^g_w \cdot (\alpha^g_{v,w} \cdot \alpha^g_{u,v,w} - 1) \cdot (\alpha^g_{t,w} - 1),$$

where $Y_w$ is the set of words with bigram and/or trigram features activated and $Y_t$ is the set of words with the topic feature activated.
Applying $\zeta$ and $\zeta(u, v)$ defined in (7.4) and (7.6) and setting
\[
\zeta(t) = \sum_{w \in V} \alpha_{t,w}^{[t,w]} \alpha_{w}^{[w]},
\]
\[
\zeta(u, v, t) = \sum_{w \in Y_{u} \cap Y_{v}} (\alpha_{u,w,v}^{[u,v,w]} \cdot \alpha_{u,v,w}^{[u,v,w]} - 1)(\alpha_{t,w}^{[t,w]} - 1) \alpha_{w}^{[w]},
\]
we can rewrite $z(u, v, t)$ by
\[
z(u, v, t) = -\zeta + \zeta(u, v) + \zeta(t) + \zeta(u, v, t). \tag{7.7}
\]

If the history $u, v$ is unseen in the training data, $\zeta(v)$ is used instead of $\zeta(u, v)$.

Now we analyze the complexity of this implementation (7.7). The first term $\zeta$ is a history-independent constant. The second one $\zeta(u, v)$ depends only on $u, v$ and the third one $\zeta(t)$ depends only on $t$. All these three terms can be pre-computed for all histories in the last iteration of training. However, the last term has $O(|\hat{X}| \cdot |T|)$ possible combinations of $(u, v, t)$. It is impossible to pre-compute and store this term for all histories. We study the scale of this resident term $\zeta(u, v, t)$ relative to $z$ and find that it is comparatively very small ($< 3\%$ on average). Figures 7.1 and 7.2 show the resident ratio of $\zeta(u, v, t)$ and $z$ for histories in the test sets of Switchboard and Broadcast News, respectively.

Figures 7.3 and 7.4 show the accumulated percentage of histories vs. the resident ratio of $\frac{\zeta(u,v,t)}{z}$ for Switchboard and Broadcast News, respectively. The $X$-axis represents the ratio of $\zeta(u, v, t)$ and $z$, and the value on the $Y$-axis represents the percentage of histories whose ratio is less than the corresponding value on the $X$-axis. For example, one can see from the curve that $95\%$ of histories has a very small resident ratio of $< 3\%$. Therefore, we can omit this term and estimate $p(w|u, v, t)$ by the approximated model
\[
p(w|u, v, t) \approx \frac{\alpha_{w}^{[w]} \cdot \alpha_{u,v,w}^{[u,v,w]} \cdot \alpha_{u,v,w}^{[u,v,w]} \cdot \alpha_{t,w}^{[t,w]}}{\zeta + \zeta(u, v) + \zeta(t)}. \tag{7.8}
\]

The topic-dependent probability can be estimated as fast as an N-gram probability. Of course, this approximation may result in some imprecision. The question is whether the error is bearable.
7.2.1 Experimental Results

We approximate topic models described in Sections 4.3 and 4.5. The perplexity of the approximated model (7.8) is not available since the right-hand side is not a real probabilistic measurement. However, we can rescore the 100-best lists (used in Chapter 4) by this approximate model and compare the WER with that of the exact model. The results are shown in Table 7.4.

<table>
<thead>
<tr>
<th>Task</th>
<th>Original WER</th>
<th>Approx WER</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWBD(97dev-test)</td>
<td>37.8 %</td>
<td>37.9 %</td>
<td>6.8 · 10²</td>
</tr>
<tr>
<td>BN(96ws)</td>
<td>34.0 %</td>
<td>34.0 %</td>
<td>1.9 · 10⁴</td>
</tr>
</tbody>
</table>

Table 7.4: Influence of approximation on topic models.
Figure 7.2: Ratio of $\zeta(u, v, t)$ and $z$ in Broadcast News.

The original WERs for Switchboard and Broadcast News are duplicated from Tables 4.6 and 4.13, respectively. The WERs of approximated models follow in the table. The speed-up is defined as the ratio of the rescoring time for the 100-best hypotheses when using the original model and that when the approximated model is used. In Switchboard, the WER only degrades by 0.1% from that of the exact model if the approximate model is used. In Broadcast News, the WER of using the approximate model is the same as that of the original model. Therefore, the approximation is worth applying; it accelerates the decoding speed in orders of magnitude with almost no degradation in the speech recognition accuracy.
Figure 7.3: Percentage of histories vs. \((z - \zeta)/z\) (SWBD).

### 7.3 Caching Recent Histories

Pre-normalizing and approximating histories are practical for some special models either without non-nested features or with only one kind of non-nested feature. For ME models with many kinds of non-nested features, such as the syntactic model, pre-normalization is impossible and approximation is inaccurate. However, not all the time \(z\) needs to be computed to acquire probability \(p\). For example, if \(p(y_2|x)\) is estimated immediately after \(p(y_1|x)\), \(z(x)\) needs to be calculated only once instead of twice. We observe that the distribution of histories respects the so-called Zipf’s Law\(^6\): a few histories occur very frequently and they cover much of the test data. Further, the occurrence of histories has the property of locality: in the N-best list or lattice

\(^6\)The product of the frequency of occurrence \(f(e)\) of some event \(e\) and the rank \(i(e)\) of \(e\) in the element set is almost a constant.
rescoring, a history tends to be referenced again soon if it is referenced. These two attributes of the distribution of histories allow us to “cache” the normalization factors of the most frequent and/or the last accessed histories in the memory and reuse them in the future. If a history is found in the cache, its normalization factor can be acquired directly without any computation; otherwise, $z$ is calculated by summing over all numerators.

The efficiency of this method depends on the fraction of history accesses found in the cache. We define the \textit{hit ratio} $\eta$ as

$$\eta = \frac{\# \text{[history accesses found in cache]}}{\# \text{[total history accesses]}}.$$ 

Letting $T_{\text{ad}}$ be the average time of computing $z$ by (2.30) without cache and $T_{\text{hit}}$ be the table look-up time if $z$ is found in cache, the average time required to compute $z$
with cache is

\[ T_{\text{new}} = \eta \cdot T_{\text{hit}} + (1 - \eta) \cdot T_{\text{old}}. \]

The speed-up of this method is thus

\[ \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{T_{\text{old}}}{\eta \cdot T_{\text{hit}} + (1 - \eta) \cdot T_{\text{old}}} \]

because \( T_{\text{hit}} << T_{\text{old}} \),

\[ \text{speed-up} \approx \frac{1}{1 - \eta}. \]

If \( \eta < 50\% \), history caching is almost meaningless.

The remaining question is what histories should be stored in the cache. We inspect two solutions for cache management by

1. statically saving the most frequent histories in the training set, and
2. dynamically caching the last accessed histories in rescored.

Figures 7.5 provides curves of the number of (the most frequent) histories (obtained from the training data) vs. the coverage rate of these histories in the test data. It is apparent that the coverage rate increases quite slowly the size of cache increases. The cache size must be at least 30K for Switchboard and 300K for Broadcast News to obtain a high coverage rate (of about 80%).

Therefore, we dynamically cache the normalization factors of the last visited histories in memory. We generate a hash function

\[ h : X \rightarrow [1..K] \]

mapping the history set \( X \) to \( K \) integers from 1 to \( K \) where \( K \) is the size of the cache. Since \( K << |X|^7 \), many histories will map to the same cell in the cache. We solve this cache address conflict problem by saving history \( x \) as the key in each cell in the cache. The value \( z(x) \) stored in the cache is valid if and only if the key matches \( x \); otherwise, it needs to be calculated by (2.30).

The algorithm below shows the steps of computing \( z \) with a cache.

\^7\text{Otherwise, the cache size is huge.}
Figure 7.5: History coverage vs. cache size.

Algorithm (Computing the normalization factor with a cache):

Step 1: Compute the cache address $h(x)$ for the history $x$.

$$h(x) = \left(\sum_{i=1}^{d} r_i \cdot x_i\right) \mod K$$

where

- $x = x_1 \cdots x_d$ is a d-dimension vector,
- $r_1 \cdots r_d$ are d pre-selected primes and
- $K$ is the size of cache.

Step 2: If $\text{key}(h(x)) = x$, then set $z = \text{value}(h(x))$;

otherwise, compute $z$ according to Equation (2.30) and update the cache by setting $\text{key}(h(x)) = x$ and $\text{value}(h(x)) = z$. 

}\]
7.3.1 Experimental Results

We use the above algorithms to compute $z$ in rescoring the 100-best hypotheses and examine their cache hit rate. We first fix the cache size of 10,000 histories and check the hit rate for different models. The results are shown in Table 7.5. Overall, the hit rate is about 80% or more for all language models, resulting in a speed-up of about 5 fold or more. It is not surprising to see that trigram models have the highest hit rates, since these models have relatively small history space, whereas the composite models have the lowest hit rate, since the number of possible histories is huge. The composite model has a higher miss rate compared to the syntactic model because the most of misses in using the former occur when switching topics.

<table>
<thead>
<tr>
<th>Task</th>
<th>3gram</th>
<th>Topic</th>
<th>Syntax</th>
<th>Comp</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWBD</td>
<td>96%</td>
<td>96%</td>
<td>87%</td>
<td>78%</td>
</tr>
<tr>
<td>BN</td>
<td>97%</td>
<td>96%</td>
<td>87%</td>
<td>77%</td>
</tr>
</tbody>
</table>

Table 7.5: Hit rate for different models.

Next, we adjust the size of the cache from 10 to 100,000 and measure the hit rate of the trigram model. See Figure 7.6 for the curve of the hit rate vs. cache size. It is apparent that the hit rate increases tremendously as the cache size grows from 10 to 10,000 but it increases slowly as the cache size further grows. Overall, the hit rate of dynamic caching is much higher than that of static caching with the same cache size. (Figure 7.5 vs. Figure 7.6).

7.4 Summary

We have discussed the computational issue of using maximum entropy models. We can pre-normalize some ME models such as N-gram models and transform them to the ARPA format of back-off models. Pre-normalized models are two to three orders of magnitude faster than the original ones.

For the models with non-nested features, there are no corresponding back-off models that we can map to. So a similar simplification is not possible. However, we can
approximate the denominators in some of these models, such as the topic-dependent model, so significantly reducing the computation needed during recognition. Experimental results in Switchboard and Broadcast News show that the approximation of the denominator achieves a tremendous speed-up and creates almost no degradation in speech recognition performance.

The approximation error, however, may be significant when the number of non-nested features in a model is very large. Therefore, we design a history-caching method to save the computation for ME models in general without any approximation. The experimental results on Switchboard and Broadcast News show that the hit rate is about 80% with a moderate cache-size for all models described in this dissertation, indicating a speed-up of about 5 fold.