EFFECTIVE LEXICAL TREE SEARCH FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Hiroki Yamamoto, Toshiaki Fukada and Yasuhiro Komori

Platform Technology Development Center, Canon Inc.
53 Inaikami-cho, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8501 Japan,
E-mail: {hiroki,fukada,komori}@cis.canon.co.jp

ABSTRACT

In this paper, we present an efficient calculation of the factored LM probabilities for speeding up the large vocabulary continuous speech recognition. We introduced a novel technique based on the independent calculation of the factored LM probability. The basic idea of the proposed method is that each factored LM probability is calculated on-demand for a new combination of a previous word hypothesis and a LM look-ahead tree node, instead of calculating all the factored LM probabilities over the tree at a time. The speaker-independent continuous speech recognition experiment was performed for 20 speakers on a 60k word newspaper dictation task. As a result, the proposed method achieved 25% improvement in speed.

1. INTRODUCTION

In most LVCSR systems, it is essential to develop a fast decoder with a small memory. The language model (LM) look-ahead pruning technique has been widely used for reducing computational cost in lexical tree search based LVCSR systems [1, 2]. LM look-ahead pruning is achieved by factoring the LM probabilities over the nodes of the lexical prefix tree depending on the predecessor word hypotheses. The factored LM probabilities are propagated backward from the leaves to the root. For each node, the maximum probability of the successor nodes is selected as the factored LM probability. When we compute all the factored probabilities beforehand, a huge memory is required to keep them. Against this problem, [2] proposes a compression of the LM look-ahead tree and an on-demand LM probability factorization for a new predecessor word hypothesis.

In this method, once a new predecessor word has appeared, all the factored LM probabilities over the tree have to be computed. However, we consider that this method will cause a serious problem in computation for larger vocabulary, because most of the computed LM probabilities have more of a chance to fall into disuse by pruning. In this paper, we propose a novel approach for reducing the computational cost in the lexical tree search effectively.

The paper is organized as follows. In Section 2, we review our LVCSR system. In Section 3, we propose an algorithm of the independent calculation of the factored LM probability. Finally, in Section 4, we give evaluation results on a newspaper dictation task.

2. SYSTEM OVERVIEW

2.1. Acoustic Model

We use a set of 26 phone notations to describe Japanese pronunciations. A set of 3,334 shared state triphone HMMs based on the 26 phones is used as an acoustic model. Each triphone has three states and each state has 24 Gaussian mixtures. The total number of states and mixtures are 2,117 and 50,808, respectively. We use 269,352 utterances (about 327 hours) of 930 speakers for training the HMMs.

The speech data were sampled at 11kHz and 16-bit. As for acoustic analysis, we adopted the standard LPC-mel cepstral analysis with applied cepstral mean subtraction to each sentence for reducing channel distortion. The feature vectors are calculated every 10ms which consists of 12th-order LPC mel cepstral coefficients, their derivatives and a difference of the log energy.

2.2. Language Model

The N-gram language model is trained using Mainichi newspaper text corpus. We used articles from 1991 to 1995, which consists of 4,527,264 sentences. These sentences are segmented into 87,763,264 words (morphs) automatically by our morphological analyzer. In this process, we distinguish the words by their notation and their pronunciation. Then the most frequent words are chosen as lexical entries. Word 2-gram and word 3-gram are constructed based on the lexical entries using backoff smoothing [3]. The cut-off threshold for the N-gram entries are 1 for 2-gram and 2 for 3-gram.

2.3. Decoder

Our decoder is based on the several widely used techniques, for example, time-synchronous one-pass dynamic-
ic programming search method, a tree-structured lexicon, word-pair approximation[3], histogram pruning[1], language model pruning[1], language model probabilities factorization[1, 2], phoneme look-ahead[4] and disk based N-gram[6, 7]. To realize quick response to the input utterances, we intended to avoid complex computation from second pass. So, in our system, most of computation is required in the first pass.

In the first pass, a frame-synchronous beam search using 2-gram is performed on a lexical prefix tree. The cross-word context dependency is considered in the first pass. To reduce search effort, the number of the predecessor word hypotheses for a state hypothesis is limited.

In the second pass, the language model probabilities for word hypotheses are re-scored using 3-gram time-asynchronously on the word graph generated in the first pass.

3. THE FACTORED LM PROBABILITY COMPUTATION

As we mentioned in the Section 1, we considered that the most of the LM probabilities have more of a chance to fall into disuse by pruning. This assumption was confirmed by the experiments on a newspaper dictation task using 200 utterances. In the experiments, we counted the number of the factored LM probabilities used in the search processing. The results are shown in Table 1.

<table>
<thead>
<tr>
<th>Vocabulary Size</th>
<th>Computed</th>
<th>Referred</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>20k</td>
<td>25,125</td>
<td>65.6</td>
<td>0.26</td>
</tr>
<tr>
<td>60k</td>
<td>70,176</td>
<td>70.6</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table 1: Number of the factored LM probabilities referred in the search process.

In Table 1, 'Computed' indicates the number of the factored probabilities computed for one predecessor word hypothesis. This number is the same as the number of nodes in the LM look-ahead tree. The column 'Referred' indicates the average number (for one predecessor word hypothesis) of the LM probabilities actually referred in the search processing. Table 1 shows that more than 99% of the computed probabilities were not used in the LM look ahead processing.

For more information, Figure 1 shows a histogram of the number of reference observed in the 20k word dictation. The X-axis is the number of the referred factored LM probabilities. It can be said that the referred number of the factored LM probabilities is less than 100 in most cases.

To avoid waste computation for the unused factored probabilities, we propose an independent calculation of the factored LM probabilities method.

Figure 1: Histogram of the number of the referred factored LM probabilities.

Our approach is that each factored LM probability is computed independently on-demand instead of computing all the probabilities over the LM look-ahead tree at once. The procedure is summarized as the following steps.

1. Compression of the lexical prefix tree

Figure 2 is an example of a lexical prefix tree. In the search process, the LM probabilities are updated only at the nodes drawn in bold circles whose predecessor node has plural branches. Therefore, as a preparation for the LM look-ahead pruning, a compressed tree is generated using the similar technique described in [3]. Figure 3 shows a tree after compression. By this compression technique, the nodes of a lexical tree with 60k words entries are reduced from 17,4862 to 70,176, which is a compression factor of about 2.5.

2. Possible word attachment to the nodes

For each node of an LM look-ahead tree, a set of words which can be reached from the node (possible words) is attached. In Figure 3, the possible words are written below the nodes.

The above steps can be done as pre-processing before recognition.

3. The factored LM probability calculation

Only the factored LM probabilities which are actually referred are calculated in the search process. For each node, when a new word hypothesis appears, 2-gram probabilities of possible words are referred and the maximum probability is computed as a factored LM probability. Once the maximum LM probability is computed, it is stored in a hash table so as to refer it again in the later process. The hash table
used here, is created on-demand for a new predecessor word hypothesis. When the predecessor word hypothesis disappears from search space, the associated hash table is cleared.

Additionally, we devised data structure of an LM look-ahead tree for reducing memory requirement to keep possible word information in nodes.

Before creating a lexical prefix tree, the lexical entries are sorted according to their notational symbols (pronunciations). Then the sequential numbers are assigned to the sorted entries as the word IDs. After that, an N-gram language model is re-constructed based on the sorted lexical entries.

As a result, the possible words for each node of a lexical prefix tree have sequential word IDs like a tree in Figure 2. This feature is not lost after the compression. Therefore, only two information, the first word ID and the number of possible words, are enough to keep possible word information. For example, at the node N3 in Figure 3, the first ID 3 and the number of the possible words 3 are stored.

4. EXPERIMENT

4.1. Effect of proposed method

We evaluated the proposed method on a newspaper dictation task. 1,000 sentences uttered by 20 speakers (10 male, 10 female) were used for evaluations. The acoustic models were speaker independent models. The experiments were performed on a Linux PC system with a Pentium III 550 MHz processor.

An on-demand factorization technique based on backward propagation[2] was also tested as a baseline. The results are shown in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>VS</th>
<th>WACC (%)</th>
<th>RTF</th>
<th>MEM (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>20k</td>
<td>93.5</td>
<td>0.94</td>
<td>32.8</td>
</tr>
<tr>
<td></td>
<td>60k</td>
<td>93.2</td>
<td>1.50</td>
<td>56.3</td>
</tr>
<tr>
<td>proposed</td>
<td>20k</td>
<td>93.5</td>
<td>0.86</td>
<td>34.5</td>
</tr>
<tr>
<td></td>
<td>60k</td>
<td>93.2</td>
<td>1.10</td>
<td>54.4</td>
</tr>
</tbody>
</table>

Table 2: Effect of the proposed method.

In Table 2, VS, WACC, RTF and MEM indicate the vocabulary size, word accuracy, real time factor and required memory, respectively. Table 2 shows that the proposed method achieves 10%-25% improvement in speed. The effectiveness of the proposed method is remarkable especially for larger vocabulary size.

Although the proposed method worked effectively in the experiments, we understood it was not always an efficient algorithm for calculation of the factored LM probabilities.

The proposed method is advantageous for calculating the factored LM probabilities, when the predecessor word hypothesis is pruned in the early stage. However, it becomes inefficient when the predecessor word hypothesis is survived over long time, because the same 2-gram probabilities are referred repeatedly for computing the maximum probability.

We considered that an effectiveness of the proposed method was depends on experimental conditions such as the pruning threshold, vocabulary size, the implementation, etc. We guessed the reason why the proposed method had worked well in the reported experiments was that the most of the word hypotheses were pruned in the early stage.

4.2. Comparison with JULIUS

We compared the performance of our decoder with JULIUS, which is the baseline LVCSR engine in Japan[8]. JULIUS is a sharable software product of the project supported by IPA (Information-technology Promotion Agency).

JULIUS supports two types of decoding mode, i.e. 'fast'
and 'standard'. The standard decoding mode pursues the recognition accuracy by using detailed acoustic models and a detailed language model. On the other hand, the fast decoding mode is designed for speeding-up the recognition with keeping accuracy as high as possible.

The comparative evaluation was performed on a 60k word Japanese newspaper dictation task. Here, 200 Mainichi newspaper sentences uttered by 46 speakers were used. These utterances are the same as the evaluation experiments for JULIUS reported in [9].

The experimental conditions for JULIUS are summarized in Table 3. The evaluation results are shown in Table 4. The results for JULIUS are quoted from the recent technical report [9]. In this experiments, the real time factor is obtained by using Ultra SPARC 300MHz.

| Feature vector | 25 dimensional MFCC |
| Acoustic model (fast) | phonetic tied mixture[10] |
| (standard) | 129 states 64 mixtures |
| Triphone | 2000 states 16 mixtures |

| Language model | Mainichi newspaper |
| (fast) | 75 months |
| (standard) | cutoff-4-4 + compressed[11] |

Table 3: Experimental conditions for JULIUS.

<table>
<thead>
<tr>
<th>Decoder</th>
<th>WACC (%)</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>JULIUS (fast)</td>
<td>88.9</td>
<td>2.9</td>
</tr>
<tr>
<td>JULIUS (standard)</td>
<td>93.2</td>
<td>16.9</td>
</tr>
<tr>
<td>Canon</td>
<td>92.9</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Table 4: Evaluation results compared with JULIUS.

We could not compare these results strictly, because the most of experimental conditions are different except input utterances and CPU. But roughly speaking, the Table 4 shows that our LVCSR system achieved higher accuracy and a bit faster computation than the fast version of JULIUS. Furthermore, compared to JULIUS with the standard decoding, our recognition system reduced recognition time by a factor of about 6.5 without significant loss in accuracy.

5. CONCLUSION

In this paper, we focused on the calculation of the factored LM probability. At first, we showed more than 99% of the computed factored LM probabilities were not referred when we calculated all of them on-demand for each new word hypothesis. Then we proposed a new approach based on the independent calculation of the factored LM probability method which could avoid waste computation for the unused probabilities. The experimental results on a 60k word newspaper dictation task showed that the proposed method improved 10%-25% in speed.

Acknowledgment: The authors are debt to Mr. Makoto Hirota, Mr. Michio Aizawa, Ms. Kazue Karako and Mr. Tsuyoshi Lastawa from Canon, who have provided Japanese morphological analyzer to us. The authors wish to thank Mr. Shigeaki Shigayama, Head of the Platform Technology Development Center at Canon Inc. for giving us the opportunity of this study.

6. REFERENCES