IMPROVEMENTS IN SEARCH ALGORITHM FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Qingwei Zhao, Zhiwei Lin, Baosheng Yuan, Yonghong Yan

Intel China Research Center
#06-01 Beijing KerryCentre.1, GuangHua Road, ChaoYang District, Beijing 100020, PRC
email: qingwei.zhao@intel.com

1. INTRODUCTION

Here we will introduce our general fast decoder for LVCSR dictation system. Certainly it has some common features that many state-of-the-art decoders have, such as, dynamic programming, beam search, multi-pass decoder (1stpass: m.uni.trigram: net: 2nd pass: cw, tri). Besides this, we also proposed new strategy to improve the search method for further reducing the search effort, which is also the features that make our decoder different with the others.

Traditionally tree copy technique must be used for integrating language model into tree search method [1][2]. Recently single-triphone-tree fast match algorithm was introduced to take instead of the tree copy technique for simplifying the search computation and saving the memory. By one kind of special-designed token propagation strategy, the n-gram language model can be integrated into the single-tree search algorithm. Moreover, a lexical tree based language model format is defined to store the pre-computed lookahead probabilities by deploying the back-off mechanism to limit the memory requirement within a manageable range, and in this way the online computation of lookahead language model can be effectively accelerated. Finally a language-independent general decoder is implemented, including English WSJ20k and Mandarin51k dictation system. Experiment results indicates that high accuracy recognition result can be attained only in the first pass by the single-triphone tree search algorithm, and search efforts can be reduced by 16% with the pre-computing lookahead LM technique.

Secondly, let us focus on the improvement of Language Model Lookahead (also called factorized language model) technique, which was extensively used in current LVCSR systems [3][5][6]. The aim of LM look-ahead technique is to incorporate the language model probabilities as early as possible into the search process, so that more tight pruning thresholds in the acoustic pruning and a lower number of maximum state hypotheses per time frame in the histogram pruning can be used. However, the computation of lookahead LM is still time consuming, as it is computed on the fly during runtime. A lexical tree based n-gram language model format is proposed in this paper, by pre-computing the lookahead probabilities in advance, the lookahead LM can be computed more efficiently and faster. This method is also implemented under the single phonetic tree structure. The details are explained in section 3.

Finally, one language-independent large vocabulary speaker-independent continuous speech recognition algorithm is accomplished. And one English and one Mandarin system are implemented as two examples that it can deal with. The test environment and results analysis is presented in section 4.

2. SINGLE-TRIPHONE-TREE SEARCH ALGORITHM BASED ON TOKEN PROPAGATION

2.1 The Concept Of Token Propagation And Single-Triphone-Tree

Token propagation strategy is first proposed in [4]. In the following we will integrate this technique into our new search method based on single-triphone-tree structure.

Token refers to an active partial path which starts from the beginning of an utterance to current time t, the tokens themselves are assumed to hold a path identifier as well as the partial alignment score s [4]. This path identifier is simply a pointer to a record of word boundary information which can be called a Word Link Record (WLR). During token propagation, potential word boundaries are recorded in a linked list structure. Hence on completion at time T, the path identifier held in the token with the best score can be used to trace back through the linked list to find the best matching word sequence and the corresponding word boundary locations.

As to the single-triphone-tree, every node in the tree is associated with a true triphone hmm model. For example, assume a vocabulary that consists of the following five words,
“abe”, “ab”, “acg”, “acgi” and “ac”, the triphone tree for this vocabulary can be constructed as illustrated in figure 1.

During tree search, the tokens are propagated through the lexicon tree, and the decoding result can be obtained from the tokens with the best matching score.

Traditionally the reason for using the tree copies is to deal with the Language model (ngram) probability. Now because the history information is recorded in the element “WLR” of tokens, this make it possible for us to use single tree instead of multi tree copies. The details will be described in the following.

2.2 Single-Triphone-Tree Search Algorithm

2.2.1 Concept of token list and propagation within nodes

One token contains two basic elements: path score and path history. For any state s’ of triphone tree node d at time t, there is one token list corresponding to it. One token list refers to a group of tokens that can propagate to current state from all the possible transition states:

Tklist(i) \( 1 \leq i \leq N \)

The path score of Tklist(i) is: \( Q(t, d, s', i) \)

The path history of it is: \( B(t, d, s', i) \) (it is in WLR format)

Path extension in decoding process can be represented by propagation tokens through the lexical tree.

For the path extension between states in one node (for example, from state s’ to s):

\[
Q = Q(t, d, s', i) + P(s/s') + b(O, s)
\]

if( \( Q > Q_{ac}(t) \cdot f_{ac} \) )

Generate new Token = (Q, B(t, d, s, i));
Add this token into TokenList of state j;

where \( Q_{ac}(t) \) is the best score in frame t, and \( f_{ac} \) is pruning threshold.

The propagation between nodes is similar to the process of within nodes, so it is omitted here.

2.2.2 Propagation between words

Every leaf node in the lexicon tree corresponds to one pronunciation word, propagating tokens between words means propagation from word-end (we) nodes of the tree to the first level triphone node in the tree. Generally, following operations are needed for the tokens that belongs to the leaf node:

1) Add Language Model probability to each token:

\[
Q = Q(t, d, S, i) \cdot B(t, d, S, i) \quad (S \text{ is the end state of the node})
\]

The history word list can be extracted from the structure \( B(t, d, S, i) \). Suppose the previous words in the path history of this token are \( w_1, w_2 \), and current word is \( w \), then LM prob \( P(w_1, w, w_2) \) will be added to \( Q(t, d, S, i) \):

\[
Q = Q(t, d, S, i) + P(w_1, w, w_2)
\]

2) Generate new token

The new path history will be generated by this operation: Integrate all of the path history from the token lists of the we node to one new path history \( B_{wplitude}(t, d) \), which can remember all the tokens history that belongs to this state. In other words, by tracing back of \( B_{wplitude}(t, d) \), the original N token history of \( B(t, d, S, i) \) can be retrieved.

3) Propagate the token from the leaf node d: \((ab+bc)\) to the first level tree node.

For each of the first level node \( X(c-x4+) \), which can be reached by the we node according to the acoustic model rules:

Firstly compute the path score:

\[
Q = Q(t, d, s', i) + P(s = 0/s' = -1) + b(O, s = 0)
\]

if( \( Q > Q_{ac}(t) \cdot f_{ac} \) )

Generate new Token = (Q, B_{wplitude}(t, d));
Add this token into TokenList of first state of node X;

2.2.3 Merge token---integrating LM into token propagation process

While beam search is proceeding on, in every frame, one “Merge” operation is needed for every token. For the token list “Tklist” corresponding to one (active) state, cluster all the tokens to sub token list corresponding to different categories, that is, under different acoustic context. It is represented as TkList, \( 1 \leq c \leq C \). (suppose there are C categories totally, here c refers to the acoustic context in terms of cross-word model).

Then the sub token list Tklist can be further clustered into different sub-links which are under same language context. That is, collect those tokens into one sub-list that have the same predecessor word list history, for instance, if we use Bigram, the predecessor word (can be obtained by trace back the B structure) for the sub tokens list should be the same word \( W_1 \);

If Trigram is used in the search process, then two previous word for the sub tokens list should be the same \( W_1, W_2 \).

For the sub token list Tklist or Tklist, only one best token (which has the highest matching score among these tokens) is needed to be preserved and all the other tokens can be discarded, that is,
2.3 Conclusion And Compare With The Other One

Because of the advantages of simplifying the search computation and saving the memory, a single triphone tree technique was employed in the decoding algorithm to take instead of traditional tree-copy technique. And in this paper, by the special-designed token propagation strategy, the n-gram language model can be integrated into the single-tree search algorithm. Though there are some common features, the implementation method in this paper is different from the other method [3]. So here we will give a brief compare.

For [3], instead of exact triphone model, a composite triphone model is used to exploit partial phonetic context information to provide a phonetic model. And the exact triphone model will be used in the second pass. In addition, second pass must be employed for obtaining better accuracy. The first pass only serves as a fast match and the only goal of it is to keep the likely word endings and their partial scores for guiding the second pass.

For ours, we used exact triphone model to build the single tree. At the same time, with the total implementation strategy, high accuracy recognition result can be obtained only in one pass. Thus it can reduce the computation cost needed in the second pass so that accelerate the search process. It is very suitable for real dictation system or other similar system where one pass is preferred.

3. IMPROVEMENTS ON LANGUAGE MODEL LOOKAHEAD TECHNIQUE

3.1 Basic Description Of Lookahead LM

For a tree based Viterbi beam search algorithm, normally an estimated language model probability \( \pi_v(d) \) for a tree node \( d \) and a predecessor word string \( v = w_{e-1}w_{e-2}...w_1 \) can be estimated by equation (3-1) as follows:

\[
\pi_v(d) = \max_{w(W)} \left( \lambda_w \cdot p(w | w_{e-1}w_{e-2}...w_1) \right) \quad (3-1)
\]

where \( W(d) \) is the set of words that can be reached from a lexical tree node \( d \), \( \lambda_w \) denotes a fractional weight, and \( p(w | w_{e-1}w_{e-2}...w_1) \) denotes the n-gram conditional word probabilities. \( \pi_v(d) \) may also be called the factored LM probability [5][6] or lookahead probability (\( P_{lookahead} \)), which is used in establishing the pruning threshold.

As a result of applying language model lookahead, a much tighter pruning beam can be achieved to speed up the decoding process. The fractional weight \( \lambda_w \) can be set to 1 or can be between 0 and 1. In some cases, \( \lambda_w \) might be more than one. The fractional weight can be determined empirically, through trial and error, or calculated.

3.2 A Lexical Tree Based Language Model Format For Pre-computing Lookahead LM

One approach for reducing the online computation of lookaheadLM is to pre-compute the LookaheadLM probabilities. A lexical tree based language model format is defined here to store the pre-computed lookahead probabilities by deploying the back-off mechanism to limit the memory requirement within a manageable range.

The lookahead probability \( P_{lookahead} \) for a tree node \( d \) and a predecessor word string \( w_{n-1}w_{n-2}...w_1 \) can be obtained in the general case as in equation (3-1) as follows:

\[
P_{lookahead} = P(d/w_{n-1}w_{n-2}...w_1) = \begin{cases} \max(\lambda_{w}, p(w | w_{n-1}w_{n-2}...w_1)), & \text{if word } w \text{ - granderivingit exists} \\ p(d | w_{n-1}w_{n-2}...w_1) \times \text{backoff}(w, w_{n-1}), & \text{if backoff weight exists} \\ p(d | w_{n-1}w_{n-2}...w_1), & \text{otherwise} \end{cases} \quad (3-2)
\]

Equation (3-2) provides an approximation of equation (3-1). Only if the top line of equation (3-2) is satisfied, are the \( P_{lookahead} \) stored in storage, such as in a look-up table. In this way, the look-up table can be kept manageable small.

In equation (3-2), we do not need to store the backoff weights since they are identical to the weights stored in the standard word based n-gram language model. In decoding, the backoff weights can be obtained through a conventional file. If decoding if the first line of equation (3-2) is not met, the lower order estimated probability with backoff weight if appropriate can be used.

Since the total number of nodes from a compressed lexical tree is comparable to the total number of words in the lexicon, the total storage for a lexical tree based n-gram language model with the equation (3-2) approximation will be on the same order as compared to the corresponding conventional word based n-gram language model. The manipulation techniques used for a normal n-gram language model may be used on to the new lexical tree based language model file of this paper.

The lookahead probabilities are calculated before recognition and stored in a lookup table. However, to reduce the size of the table, only the entries which are derived directly from the n-gram probabilities (not by the backoff) are stored. The others which are derived from the backoff probabilities (n-gram backoff to (n-1)-gram) are approximately back-offed to (n-1)-gram estimated probabilities. Through compression, the size of the table can be reduced to a manageable level.

Since the Lookahead LM with no back-off is stored in a lookup table, during decoding, the Lookup table is loaded into memory at first, then the LookaheadLM value can be queried by dynamically looking up the table.
4. DECODING EXPERIMENTS

4.1 General Decoder And Test Environment

Finally we implemented one language-independent LVCSR decoder. The decoder will not use any language-dependent feature for acceleration. The mono phone list, word dictionary, Language model and acoustic model are different for different language, the decoder uses a unification way to deal with these model data set, so it can handle different language.

The experiment tests were carried out on both the English WSJ20k task and on Mandarin dictation task. The WSJ 92 task is a standard task, of which training set is SI284, test set is the 20k ARPA 1992 evaluation set consisting of 333 sentences from 8 different speakers. And the Mandarin task is what we designed for testing system performance. The corpus contains about 270 hour training data, and 12 minute test data (110 sentences). The test data is collected from 11 persons who are not within the training set, each one 10 utterances. For acoustic model, there are about 6000 states, and 12 mixtures for every state.

We have accomplished realtime decoding for WSJ20k task (error rate: 11.9 for 1st pass, 9.7 for 2nd pass). And in this paper, we will focus on the experiment results of Mandarin test, since it is a more complicated task, the dictionary size (51k) is larger, the search cost is greater than WSJ20k. Total nodes of the single triphone tree is 245k. In addition, The decoding experiment is implemented on Pentium III 550.

4.2 Experiments Result And Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>Active tokens</th>
<th>Time (xRT)</th>
<th>WER (1st pass)</th>
<th>WER (2nd pass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-demand Computing</td>
<td>3093</td>
<td>3.0</td>
<td>10.5</td>
<td>9.4</td>
</tr>
<tr>
<td>LA-Bigram</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-Computing LA-Bigram</td>
<td>3087</td>
<td>2.5</td>
<td>10.7</td>
<td>9.5</td>
</tr>
<tr>
<td>Other new Method</td>
<td>1169</td>
<td>1.3</td>
<td>11.1</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Table: Effect of Single triphone tree and LM lookahead technique on the search effort and recognition error rate.

Here single-triphone tree search is employed in all the methods, and lookahead bigram is used in the experiments. From table 1, firstly we can see that, even though there are two pass decoding procedure in our current system, only with the single triphone tree search technique, we can attain really high decoding precision only in the first pass (here the error rate is 10.5), this is very useful for online decoding system, where the first pass decoding is preferred.

Secondly, we can see that the pre-computing Lookahead Lm technique has reduced the search effort by about 16% compared with the on-demand computation, with the error rate only rising by 1% relatively. So, we think that this technique is useful in LVCSR system when employing LM lookahead technique.

Furthermore, we have attained nearly RT (1.3xRT) decoding speed currently, while employing other technique, such as acoustic model lookahead, state prediction pruning, etc. We will introduce these technique in other conferences, since we have not enough places here.

5. CONCLUSION

In this paper, improvements in search algorithm for large vocabulary continuous speech recognition task have been proposed. Firstly, single triphone tree technique was put forward to replace the traditional tree-copy technique for simplifying the search computation and saving the memory. Token propagation strategy is employed for the search algorithm, the details of it are described, especially how it accomplished the language model integration while using single triphone tree. This method is also compared to the other method. The advantage of it is that, it can attain high accuracy in the first pass, with its full triphone tree structure and different implementation strategy.

Another approach for accelerating the online computation of lookaheadLM is to pre-compute the LookaheadLM probabilities. A lexical tree based language model format is defined to store the pre-computed lookahead probabilities by deploying the back-off mechanism to limit the memory requirement within a manageable range. The LookaheadLM with no back-off is stored in a lookup table. By this method, the search effort can be effectively reduced with nearly no loss on the recognition accuracy.

Our current software has also been optimized by Intel MSL group in terms of Pentium III architecture which greatly reduced the decoding effort.

6. REFERENCES