A TRELLIS BASED FAST LATTICE GENERATING ALGORITHM

Wei Li, Ji Wu, Zhiguo Wang

Department of Electronic Engineering, Tsinghua University, Beijing
Email: w-li-06@mails.tsinghua.edu.cn

ABSTRACT

Lattice is widely used as a kind of the search results in Large Vocabulary Continuous Speech Recognition (LVCSR). A new lattice-generation algorithm is presented in this paper. The algorithm is based on a classical forward-backward decoding method, which is proved to be highly efficient. Moreover, some improvements have been done to satisfy the requirements in the lattice decoding. Two Chinese mandarin large-scale speech recognition tasks are used to evaluate the proposed algorithm and the experimental results show that our algorithm can both improve decoding speed and save decoding space significantly without sacrificing the recognition accuracy, compared with the widely used Lattice decoding method as [7].

Index Terms— lattice, speech recognition, trellis, forward-backward

1. INTRODUCTION

The state of the art LVCSR systems should have the ability to generate multiple kinds of recognition results including “1-Best”, “N-Best”, “Lattice” and “Confusion-Network”. Comparing with both “1-Best” and “N-Best”, the lattice is a very efficient way to provide sufficient information as much as possible. In the meanwhile, it is also the basis for the generation of the confusion-network. As a result, it is one of the key issues about how to efficiently and effectively generate the lattice in the LVCSR systems.

In [1], an algorithm namely “extension of the one-pass DP algorithm” is originally proposed which is to generate lattice from the “1-Best” result. Based on this algorithm, some different algorithms have been developed such as “Token Passing algorithm” [2].

The above algorithms belong to the one-pass decoding, which only include forwarding decoding process. Some other algorithms instead adopt the two-pass decoding strategy namely the forward-backward decoding. This decoding strategy includes two steps namely the forward decoding step and the backward decoding step. The forward-backward decoding can provide a much faster decoding speed than the forward-only methods, without sacrificing the performance on the recognition accuracy [3].

Trellis-based decoding algorithm is a kind of forward-backward methods to generate N-Best decoding results [4]. This method uses the A* algorithm as the backward decoding and shows the high efficiency [5]. Similar decoding strategy was also used in [6]. However, this algorithm is unable to generate lattice directly.

In this paper, a new two pass Trellis-Based Lattice-Generating algorithm (TBLD) is proposed, including a forward Viterbi decoding step and a backward A* decoding step. Recognition results show that the new algorithm is much faster than the Token Passing algorithm, with comparable performance on the recognition accuracy.

The rest of this paper is organized as follows. In Section 2 we review the related works in LVCSR area, Section 3 describes some important issues, which are considered in our algorithm. In Section 4, the total framework of TBLD is presented. The experimental setup and the recognition results have been shown in Section 5. Finally, Section 6 concludes our discussion.

2. THE TRELLIS-BASED SEARCHING ALGORITHM

[4] presented an N-Best decoding algorithm including two steps: 1) a forward Viterbi algorithm to generate a Trellis; 2) a backward A* algorithm to obtain N-Best results.

In the backward step, each trellis node should be backwardly extended. Before the extension, every node has several input arcs. Each input arc, namely isolated word extension arc, is identified by a word ID and is necessary to record several data including the arc dependent score and the corresponding starting node. Each of these arcs is located in a path that from the beginning of the utterance to the end of it. So the overall score of the path is made up of three parts: the backward accumulated distance from the end of the utterance to the current node, the score of the “isolated word extension” arc, and the distance from the beginning of the utterance to the start node of the “isolated word extension” arc, which is calculated in the first step and can be obtained directly from the trellis. The decoder could therefore tie each new node with the overall score of a path, and the node with higher score should be extended first. To

1 In our system, each node in the Trellis represents the end time of a specific word, which was recognized in the forward step.
keep the nodes in order, a priority queue, which is called the main stack in [4], has been introduced into the decoding process, and the overall score of node will be treated as priority. Finally, N candidate path could be found out and they are treated as N-Best results.

Trellis based decoding algorithm is an efficient way to generate N-Best result. It is natural to use such algorithm in the generation of the Lattice with some modifications as follows:

1. In the trellis-based N-Best decoding algorithm, one node can exist in different path, so it is possible that one node might be extended several times. However, in the Lattice-generation, it is better for each node being extended just one time. Otherwise the structure of Lattices might not be kept. In order to resolve this problem, the priority queue can be changed to use the topologic positions of the nodes to substitute their overall scores. The larger one node’s representation time, the former that node should be extended. Moreover when there are two nodes located at same time with the same word (ID), both nodes should be merged.

2. In the trellis-based N-Best decoding algorithm, the decoding process terminates when top-N results are generated, without considering whether there still exist nodes in the priority queue or not. In our algorithm instead, each node is still extended until the priority queue is empty.

3. KEY ISSUES

3.1 The Cross Word Context-Dependent Unit

In order to use context-dependent (CD) unit such as the Cross Word Tri-hone in the LVCSR systems, one needs to take some approximations in the backward extension with detail as follows.

Some nodes possibly have multiple output arcs, each of which is generated in the former node extension. Because those arcs may represent the distinct words, and have different starting phones accordingly, it is not easy to get a consensus-starting phone from all of these output arcs to establish the rightmost side of the extending node’s final CD unit.

There are several ways to solve this problem. In this paper, a logical model is used to replace a real one as the extending node’s final unit, in which the average score of the top-K physical units is used as the logical one.

3.2 Elimination Silence Arcs

Actually it is hard to handle the score of the language model when some arcs are determined as the silence. Meanwhile, silence does not provide any information and therefore might be unnecessary to appear in the recognition results. In this case, an optional elimination of the silences is reasonable in the lattice generation process with detail as follows:

After a specific node denoted by X is extended, the node dependent input and output arcs can be obtained. Let O={o1, o2, ..., oN} denote the set of output arcs where N is the number of output arcs in O. Let again I={i1, i2, iM} represent the set of input arcs where M is the number of input arcs in I. If the node represents the end of a silence, the procedure to eliminate the silence arc is in the following:

1. Identify the begin node set J of I;
2. Change the begin node of each output arc o as J and clone o if necessary;
3. Delete X and I.

Figure 1 illustrates the whole process of the elimination.

Figure 1: Silence Node’s Elimination

3.3 Pruning

During the process, the extension of a certain node results in the generations of multiple new nodes. As a result, pruning is necessary so as to alleviate exponential increasing of the nodes in the whole process. Currently, two kinds of pruning techniques are widely used in the decoding algorithm namely beam pruning and histogram pruning.

In this paper, such two strategies are both used accordingly. After the extension of a certain node, beam pruning is executed once the overall score of an extension node falls below a predefined threshold. The new node is eliminated with its output arc. The Histogram Pruning is simultaneously done by only preserving the nodes with top-M score and deleting others.

3.4 Node Merging

After the backward extension of a certain node, some of new generated nodes might be identical to those already existing in the priority queue. In this case, such nodes are merged into old one.

4. THE FRAMEWORK OF PROPOSED ALGORITHM

The decoding process includes two steps including 1) the forward step; 2) the backward step. In the forward step, a modified token passing algorithm is used to generate a
trellis. And the trellis is then used to generate the output lattice in the backward step.

A 4-state automatic machine can represent the process of the backward step. The set of states includes Init State (IS), Backward Extension State (BES), Merge and Prune State (MPS), and Final State (FS). The procedure of the backward step, which is shown in Fig. 2, is in the following:

Initialization:
At the beginning of the process, the automatic machine is in the state of IS. At the same time, a priority queue is established with singly node, which is the end node of the utterance. The state of the automatic machine is then changed into BES to generate lattice.

Iteration:
This part is repeated until the state of automatic machine is arrived.
1. If the decoding machine is in BES state, the top node is tried to be fetch out in the priority queue. If fail, the automatic machine comes into FS state; otherwise, the backward extension of node is triggered. The new input arcs of node are therefore generated and the beginning nodes of those new arcs are established.
2. If X is a silence node, the optional silence elimination is performed. Then the decoding machine comes into MPS.
3. In MPS, merge and pruning operations are executed for the new established nodes. The machine then comes to BES.

Termination:
When the decoding machine has arrived at The Final State, the total algorithm has finished, and the lattice is generated.

5. EXPERIMENTS

5.1 Experimental Setup
In order to demonstrate the efficiency of the new algorithm, two testing set are used namely “2003-Chinese 863 Speech LVCSR Evaluation Corpus (CSLEC03)”; “2004-Chinese 863 Speech LVCSR Evaluation Corpus (CSLEC04)”. CSLEC03 includes 120 sentences from 12 speakers (6 male, 6 female) and CSLEC04 includes 200 sentences from 20 speakers (10 male, 10 female).

In order to evaluate our approach, the large-scale decoding system namely HDecode, which is implemented in HTK3.4 toolkit, is used as the baseline system. Moreover, a bi-gram language model was used in all experiments.

5.1 One-Best Measure
In table 1, the experimental results on 1-best accuracies are presented for both TBLD and HDecode with two testing sets. It can be observed that an improvement is obtained for the recognition accuracy by using our approach.

<table>
<thead>
<tr>
<th></th>
<th>1-Best Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLD</td>
<td>CSLEC03 63.28</td>
</tr>
<tr>
<td></td>
<td>CSLEC04 58.68</td>
</tr>
<tr>
<td>HDecode</td>
<td>CSLEC03 60.64</td>
</tr>
<tr>
<td></td>
<td>CSLEC04 57.78</td>
</tr>
</tbody>
</table>

5.2 OCER Measure
In order to compare the performance on the lattice generation, a metric namely Oracle Character Error Rate (OCER) is defined as the character error rate derived by the true label and the corresponding best-match path in the lattice.

There is also another issue on the lattice density; the histogram prune method is used as the primary prune algorithm accordingly. Multiple lattices are generated with different densities with different histogram number changed from 1 to 15. For each lattice derived, corresponding OCER is calculated. Figure 3 shows that the OCER’s decrease with the growth of lattice density in both two testing sets.
Table 2: Recognition OCERs for HDecode and TBLD

<table>
<thead>
<tr>
<th>Corpus</th>
<th>T/Num</th>
<th>TBLD(%)</th>
<th>HDecode(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLEC03</td>
<td>5</td>
<td>16.78</td>
<td>17.86</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>12.50</td>
<td>12.47</td>
</tr>
<tr>
<td>CSLEC04</td>
<td>5</td>
<td>18.88</td>
<td>19.79</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>14.19</td>
<td>13.58</td>
</tr>
</tbody>
</table>

In table 2, TBLD is compared with HDecode on the performance of the lattice generation. When the lattice is sparse, TBLD performs better than HDecode. When the lattice is dense, HDecode performs better. But such performance difference is quite insignificant.

5.3 Execution Time and Space Complexity

In this paper, the total decoding time is used as the metric to measure the efficiency of the decoding system.

Table 3: Recognition xRTs for HDecode and TBLD

<table>
<thead>
<tr>
<th>Corpus</th>
<th>T/Num</th>
<th>xRT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TBLD</td>
</tr>
<tr>
<td>CSLEC03</td>
<td>5</td>
<td>5.55</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>6.70</td>
</tr>
<tr>
<td>CSLEC04</td>
<td>5</td>
<td>3.61</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>4.34</td>
</tr>
</tbody>
</table>

Table 3 shows the decoding xRT with HDecode and TBLD. It's observed that our approach is much faster than HDecode.

In order to compare the space complexity, the average number of arcs and nodes is used to measure the space complexity of the two decoding systems. Table 4 and Table 5 present some space complexity data.

Table 4: Average Arc Number for HDecode and TBLD

<table>
<thead>
<tr>
<th>Corpus</th>
<th>T/Num</th>
<th>Ave. Arc Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TBLD</td>
</tr>
<tr>
<td>CSLEC03</td>
<td>5</td>
<td>7024.51</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>67408.7</td>
</tr>
<tr>
<td>CSLEC04</td>
<td>5</td>
<td>6616.15</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>67078.9</td>
</tr>
</tbody>
</table>

Table 5: Average Node Number for HDecode and TBLD

<table>
<thead>
<tr>
<th>Corpus</th>
<th>T/Num</th>
<th>Ave. Node Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TBLD</td>
</tr>
<tr>
<td>CSLEC03</td>
<td>5</td>
<td>1192.56</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>4581.45</td>
</tr>
<tr>
<td>CSLEC04</td>
<td>5</td>
<td>1154.67</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>4592.13</td>
</tr>
</tbody>
</table>

2 T/H means the Maximum Token Number Beam in HDecode and the Histogram Parameter in TBLD.

It is observed that both the number of arcs and the number of nodes in TBLD is much smaller than those in HDecode.

To sum up, TBLD has much less time (and space) complexity than HDecode.

6. CONCLUSION

In this paper, we proposed a 2-step decoding algorithm to generate lattice. This algorithm is developed from a trellis-based N-Best decoding method.

We have also compared the performance of proposed decoding algorithm with the traditional Token-passing approach. The result has shown with similar OCER, this algorithm is much faster than the token-passing method, with less space complexity.

7. REFERENCES


