A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition

Hainan Xu\textsuperscript{1,2}, Tongfei Chen\textsuperscript{1}, Dongji Gao\textsuperscript{1}, Yiming Wang\textsuperscript{1}, Ke Li\textsuperscript{1}, Nagendra Goel\textsuperscript{3}, Yishay Carmiel\textsuperscript{2}, Daniel Povey\textsuperscript{1}, Sanjeev Khudanpur\textsuperscript{1}

\textsuperscript{1}Center for Language and Speech Processing, Johns Hopkins University; \textsuperscript{2}IntelligentWire, Seattle WA; \textsuperscript{3}Go-Vivace Inc., USA

Lattice Rescoring

- In speech recognition, decoding is usually done on a static decoding graph compiled from an n-gram.
- RNNLM rescoring helps further reduce WERs by (partially) replacing LMs weights on a decoded lattice.
- A naive implementation to rescore the lattice is lattice; partial) replacing LMs weights on a decoded RNNLM.
- The heuristics also consistently improves WER.
- The evaluation is done with TensorFlow to Kaldi.

Pruned Algorithm

- For each arc to be expanded, we compute a score reflecting how likely this arc will become part of the best-path:
- Arcs that are not very promising (out of the beam) are not expanded;
- Arcs that are more promising get expanded first, so that output lattice states encode “better” history.

Heuristic

- The heuristic is computed as
  \[ H(c) = \alpha(c) + \beta(a) + \delta(c) \]  
- \( a \): the corresponding state in the input lattice;
- \( c \): a state in the output lattice;

\[ \delta(c) = \max \{ \beta(c) - \beta(a), \beta(c) < +\infty \} \]

\[ \beta(\text{prev}(c)) \leq \beta(c) = +\infty \]

prev(c) is the previous state of c on the best path from start to c.

Lattice-rescoring Speed

Word-error-rate

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Test set</th>
<th>ARPA baseline</th>
<th>RNNLM rescoring with n-gram approximation</th>
<th>Corpus</th>
<th>Test set</th>
<th>ARPA baseline</th>
<th>RNNLM rescoring with n-gram approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMI-HM</td>
<td>dev</td>
<td>24.2</td>
<td>24.0</td>
<td>23.4</td>
<td>23.5</td>
<td>23.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.5)</td>
<td>12.1</td>
<td>12.3</td>
<td>11.5</td>
<td>11.5</td>
<td>11.3</td>
<td></td>
</tr>
<tr>
<td>SWBD</td>
<td>eval2000</td>
<td>24.0</td>
<td>23.8</td>
<td>23.0</td>
<td>23.1</td>
<td>23.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>swbd</td>
<td>8.1</td>
<td>8.2</td>
<td>7.6</td>
<td>7.6</td>
<td>7.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.8)</td>
<td>12.4</td>
<td>12.5</td>
<td>11.5</td>
<td>11.5</td>
<td>11.5</td>
<td></td>
</tr>
<tr>
<td>WSJ</td>
<td>dev93</td>
<td>7.6</td>
<td>6.9</td>
<td>6.2</td>
<td>6.2</td>
<td>6.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eval92</td>
<td>5.1</td>
<td>4.2</td>
<td>3.9</td>
<td>3.9</td>
<td>3.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.8)</td>
<td>14.3</td>
<td>14.4</td>
<td>13.4</td>
<td>13.4</td>
<td>13.4</td>
<td></td>
</tr>
<tr>
<td>LIB</td>
<td>dev-clean</td>
<td>5.7</td>
<td>4.8</td>
<td>4.3</td>
<td>4.3</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dev-other</td>
<td>14.5</td>
<td>13.7</td>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: WER of Lattice-rescoring of Different RNNLMs

Acknowledgements

This work was partially supported by DARPA LORELEI award number HR0011-15-2-0024, NSF Grant No CRI-153128 and IARPA MATERIAL award number FA8650-17-C-9115 and by IntelligentWire. The authors would also like to thank the TensorFlow team at Google for their help during the project.

Contact Information

- Web: \text{http://www.hainanxu.com}
- Email: hxu31@jhu.edu

Overview

- Usually lattice-rescoring uses n-gram approximation to limit search space;
- We propose a heuristics that finds more promising arcs to expand, and use it for pruning;
- Complexity of the algorithm grows approximately (empirically) linear with n-gram order, compared with exponential growth of the baseline algorithm;
- 4X and 10X faster for 4-gram and 5-gram;
- The heuristics also consistently improves WER;
- The evaluation is done with TensorFlow to Kaldi;