Backward Viterbi Beam Search for Utilizing Dynamic Task Complexity Information

MIN TANG AND PHILIPPE DI CRISTO (mint@voicebox.com, philipped@voicebox.com)

VoiceBox Technologies, Bellevue, Washington, USA

Backward Mazes Solving

A maze-solving technique (when stuck) is to begin at the End and try to work backwards to the Start. Normally, it is easier to solve a maze backwards because the maze designer has set traps (tough decision points) for the one who solves mazes forwards. And some of these traps do not trap those who solve mazes backwards. Maze designers usually consider it cheating to solve mazes backwards.

Build Backward Viterbi Decoding ASR

On Top of Traditional ASR Engines

- Reverse Acoustic Model
  The Gaussian Mixture Model (GMM) could be reversed by simply remapping the monophone/triphone to senones. The state transition matrix could be reversed by modifying the matrix so that it generates the same acoustic score without re-estimating the state transition probabilities.

On Top of Finite-State-Transducer based ASR Engines

- Reverse Dictionary
  Simply reversing the order of all phonemes. For example, “Washington” pronunciation:
  Pron 1: WA AA SH IH NG T AH N
  Pron 2: W A O SH IH NG T AH N
  “Washington” reversed pronunciations:
  Pron 1: N AH T NG IH SH AA W
  Pron 2: N AH T NG IH SH AO W

On Top of Finite-State-Transducer based ASR Engines

Since all the grammar, dictionary, acoustic model indices have already been compiled into one network, we could do the trick by simply reversing the network without making any modifications to the acoustic model, grammar and dictionary.

Experiments

- ASR Engine: PocketSphinx
- Acoustic Model: Use the models from the PocketSphinx package as is
- The acoustic model: TI-Digits model with 670 state-clustered senones
- Test set: standard TI-Digits adult test set, 8700 utterances
- Tried both the Finite State Grammar and the statistical language model
  - Finite State Grammar: A digit loop with the maximum digits length set to 7
  - Statistical Language Model: a unigram with uniform distribution

NSCS Address Entry Task: State and City Recognition

- NSCS: the task to recognize full address which contains address number, Street, City and State
- 9.60% Totto: TI-Digits model with 670 state-clustered senones
- Test set: 396 utterances from 6 native American speakers
- A multi-pass strategy: state recognition first, then city recognition followed by house number and street name recognition
  - State Recognition Grammar: SRoot → SState
  - City Recognition Grammar: SRoot → SCity+State

- the backward Viterbi system is 3 times faster than the conventional system with 90% errors reduction.

Comparison between forward Viterbi and backward Viterbi on NSCS task: state recognition

<table>
<thead>
<tr>
<th>System</th>
<th>Real Time Factor</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammar: the forward Viterbi</td>
<td>0.02xRT</td>
<td>0.89%</td>
</tr>
<tr>
<td>Grammar: the backward Viterbi</td>
<td>0.05xRT</td>
<td>0.71%</td>
</tr>
</tbody>
</table>

Comparison between forward Viterbi and backward Viterbi on NSCS task: city recognition

<table>
<thead>
<tr>
<th>System</th>
<th>Real Time Factor</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammar: the forward Viterbi</td>
<td>0.05xRT</td>
<td>0.51%</td>
</tr>
<tr>
<td>Grammar: the backward Viterbi</td>
<td>0.02xRT</td>
<td>0.45%</td>
</tr>
</tbody>
</table>

Discussion

- Based on the analysis of dynamic task complexity, use the forward or backward Viterbi beam search as required by the task
- Is human being doing the same thing?
- Shortcoming: asynchronous