UNSUPERVISED SPEAKER ADAPTATION USING ATTENTION-BASED SPEAKER MEMORY FOR END-TO-END ASR

Leda Sari*, Niko Moritz, Takaaki Hori, and Jonathan Le Roux

ICASSP 2020
May 2020

MITSUBISHI ELECTRIC RESEARCH LABORATORIES (MERL)
Cambridge, Massachusetts, USA
http://www.merl.com

* Currently at UIUC. This work was done during an internship at MERL.
Introduction

• Mismatch in speaker characteristics reduces ASR accuracy
• At test time, we often encounter unseen speakers
• Speaker adaptation: adjusting an ASR model to make it more robust to speaker variation
• Our goals:
  – An adaptation method for end-to-end (E2E) ASR
  – Applicable in unsupervised settings
  – Providing fast adaptation
  – Useful for streaming applications
  – Robust to internal speaker changes
Previous Work on Speaker Adaptation for E2E ASR

• Strong recent interest in E2E ASR but not many adaptation techniques
  – Append i-vectors [Audhkhasi+ 2017]
  – Use transformed features [Chorowski+ 2014]
  – Speaker adversarial training [Meng+ 2019]

• Utterance-level approaches → cannot handle internal speaker changes

• Some of them are applied only to the input layer

• Some of them require speaker label → supervised
Proposed Method

• Build memory consisting of a set of speaker embeddings (e.g., i-vectors) from training data
• Use this memory to extract an embedding (M-vector) for unseen speakers at each frame
• Append the M-vector to the neural net activations

• Inspired by the neural Turing machine (NTM) [Graves+ 2014]
  – Use memory reading operation to determine instantaneous speaker embeddings
Advantages of the Proposed Method

• Unsupervised during test time

• Frame-level approach
  – Faster adaptation
  – Useful for streaming applications
  – Robust to internal speaker changes

• NTM interpretation allows a direction for writing mechanism (future research)

• Flexible
  – Different embeddings can be used in the memory (i-vectors, x-vectors, etc.)
  – Can be used in different architectures, here joint CTC and attention model
Joint CTC and Attention E2E ASR [Watanabe+ 2017]

- Combination of sequence-to-sequence models to mitigate their individual disadvantages
- Maximize the log-likelihood of the labels given the input features
- Multitask objective

\[ L_{joint} = \lambda L_{ctc} + (1 - \lambda)L_{att} \]
Neural Turing Machine [Graves+ 2014]

- Dot product:
  \[ K(q_t, M_n) = \frac{q_t^T M_n}{\|q_t\| \|M_n\|} \]

- Attention weights:
  \[ w_t(n) = \frac{e^{\gamma t} K(q_t, M_n)}{\sum_l e^{\gamma t} K(q_t, M_l)} \]

- Read vector:
  \[ r_t = \sum_{n=1}^{N} w_t(n) M_n \]
Adaptation with i-vectors
Adaptation with i-vectors
Adaptation with i-vectors

Append the i-vector

$x$  BLSTM  BLSTM  BLSTM  BLSTM  BLSTM  CTC  $y_{ctc}$

Attention  Decoder  $y_{att}$
Adaptation with i-vectors

\[ x \rightarrow \text{BLSTM} \rightarrow \text{BLSTM} \rightarrow \text{BLSTM} \rightarrow \text{BLSTM} \rightarrow \text{BLSTM} \rightarrow \text{CTC} \rightarrow y_{ctc} \]

\[ i-vector \rightarrow y_{att} \]
Adaptation with i-vectors
Adaptation with i-vectors

$\mathbf{x}$

BLSTM

BLSTM

\[ \text{i-vector} \]

CTC

$y_{ctc}$

Attention

Decoder

$y_{att}$
Adaptation with i-vectors
Adaptation with M-vectors

M-vectors

$\mathbf{x}$

BLSTM → BLSTM → M-vectors → BLSTM → BLSTM → BLSTM → [CTC] → $y_{ctc}$

[Attention] → Decoder → $y_{att}$
Memory-based Adaptation for E2E ASR

\[ K(q_t, M_n) = \frac{q_t^T M_n}{\sqrt{d}} \]

\[ w_t(n) = \frac{e^{\gamma_t K(q_t, M_n)}}{\sum_l e^{\gamma_t K(q_t, M_l)}}, \gamma_t = 1 \]

\[ r_t = \sum_{n=1}^{N} w_t(n) M_n \]
Experimental Setup

• ESPnet [Watanabe+ 2018] joint CTC and attention framework
• Experiments on two datasets
  – WSJ (81.3/1.1/0.7hr, 283 train speakers)
  – TED-LIUM2 (211.1/1.6/2.6hr, 1267 train speakers)
• BLSTM based encoder, LSTM based decoder, location-based attention in between
• Unadapted baseline: ESPnet default recipes
• Speaker adapted baseline: appending i-vectors to hidden layers
• Experiments on the location of the memory block
• Experiments on utterances with speaker change point
• Run for four times with different seeds and report the best results (based on dev set)
WER as a function of the adaptation layer: WSJ

- Layer=0 denotes input features
- M-vector: similar on dev set WER

Layer No

<table>
<thead>
<tr>
<th>Layer No</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.5%</td>
</tr>
<tr>
<td>1</td>
<td>6.4%</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Graph showing WER (%) vs Layer No with three lines:
- i-vector-dev
- M-vector-dev
- No-adapt-dev

ICASSP, May 2020
WER as a function of the adaptation layer: WSJ

- Layer=0 denotes input features
- M-vector: similar on dev set WER, 10.6% better on eval set WER
WER as a function of the adaptation layer: TED-LIUM2

• Similar performance with i-vectors and M-vectors
WER as a function of the adaptation layer: TED-LIUM2

- Similar performance with i-vectors and M-vectors
Speaker level vs Utterance level i-vectors

• i-vector system uses speaker i-vectors, hence requires speaker knowledge during test time
• Remove the advantage by using utterance level i-vectors
• M-vectors perform better than speaker or utterance i-vectors on the test data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Speaker i-vector</th>
<th>Utterance i-vector</th>
<th>M-vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
<td>dev</td>
</tr>
<tr>
<td>WSJ</td>
<td>6.4</td>
<td>4.7</td>
<td>6.4</td>
</tr>
<tr>
<td>TED-LIUM2</td>
<td>11.7</td>
<td>11.2</td>
<td>11.7</td>
</tr>
</tbody>
</table>
Utterances with Speaker Change

- Utterance i-vectors are repeated for all frames in an utterance
- M-vectors are computed at frame-level
- Compare the speaker change performance
- Simulate the condition by removing silences at the boundary and concatenating audio
- Denote the new test sets with *, e.g. dev93*, eval92*
Utterances with Speaker Change: Results

• M-vectors are more robust to speaker change

<table>
<thead>
<tr>
<th></th>
<th>single speaker</th>
<th></th>
<th>speaker change</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev93</td>
<td>eval92</td>
<td>dev93*</td>
<td>eval92*</td>
</tr>
<tr>
<td>WSJ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i-vector</td>
<td>6.4</td>
<td>4.7</td>
<td>10.4</td>
<td>7.8</td>
</tr>
<tr>
<td>M-vector</td>
<td>6.5</td>
<td>4.2</td>
<td><strong>7.6</strong></td>
<td><strong>4.9</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TED-LIUM2</th>
<th>single speaker</th>
<th></th>
<th>speaker change</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
<td>dev*</td>
<td>test*</td>
</tr>
<tr>
<td>i-vector</td>
<td>11.7</td>
<td>11.2</td>
<td>16.1</td>
<td>15.9</td>
</tr>
<tr>
<td>M-vector</td>
<td>11.8</td>
<td>11.0</td>
<td><strong>14.1</strong></td>
<td><strong>11.9</strong></td>
</tr>
</tbody>
</table>
Summary

• An NTM-inspired unsupervised speaker adaptation method for E2E ASR
  – Frame-level approach
  – Online adaptation
  – The M-vector at each frame is a weighted combination of the memory vectors
  – Weights are determined by the attention-based read mechanism

• M-vectors
  – Perform similarly or slightly better than using the oracle i-vectors
  – More robust to speaker changes within utterances than the i-vectors