Incorporating articulatory feature information in ASR transfer learning

Mahir Morshed

8 April 2020
Outline

1. Background

2. Preparation
   • Dataset
   • Modeling

3. Current experiments

4. Future work
Articulatory features

- Facets of phone production by which differences between such phones may be characterized
- Although two languages may lack a common phone, close equivalents may exist which differ in one or two characteristics
  - /t/ in South Asian languages vs /t/ elsewhere
  - /e/ vs /ɛ/ elsewhere
  - /r/ vs /ɻ/ elsewhere
  - /a/ vs /ɑ/ elsewhere
  - ...

(such near-equivalencies often manifest in language acquisition process)
Articulatory feature detection

Binary feature detectors using fully-connected networks [1]

CNN-based multiclass detection [2] (place and manner only)

CTC-based multiclass detection [3]

More recently, simultaneous feature detection using transformer-based systems [4]

Figure: CNN-based feature detector structure described by Merkx and Scharenborg.
End-to-end transfer learning

- Provide a better initial state from which to impart shared information between to a network
- Frequently based on retraining layers in a network already exposed to one language
  - More recent efforts see strapping of language models to new systems
End-to-end transfer learning

- Freezing lower layers of a system à la Wave2Letter for retraining an English system on German speech [5]
- More recently, fusing language-independent CTC-based model near the output of the target language network [6]

\[
\begin{align*}
    s^{LM}_u &= W^{LM}d^{LM}_u + b^{LM} \\
    g_u &= \sigma(W^g[s^{S2S}_u; s^{LM}_u] + b^g) \\
    s^{CF}_u &= W^{CF}[s^{S2S}_u, g_u \odot s^{LM}_u] + b^{CF} \\
    P_{S2S}(y|x) &= \text{softmax}(\text{ReLU}(W^{\text{out}}s^{CF}_u + b^\circ))
\end{align*}
\]

Figure: 'Cold fusion’ as described by Inaguma et al.
Corpora

- Large corpora of Bengali, Nepali, Sinhalese, Javanese, and Sundanese speech [7]
- In the absence of train/dev/test splits, created some myself (80/10/10)
- Due to differing corpora sizes, may not use all data in each split to aid comparability
### Corpora splits

<table>
<thead>
<tr>
<th>Language</th>
<th>Train (h)</th>
<th>Dev/Test (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bn</td>
<td>172.43</td>
<td>21.56</td>
</tr>
<tr>
<td>jv</td>
<td>236.70</td>
<td>29.59</td>
</tr>
<tr>
<td>ne</td>
<td>123.71</td>
<td>15.47</td>
</tr>
<tr>
<td>si</td>
<td>179.6</td>
<td>23</td>
</tr>
<tr>
<td>su</td>
<td>266.06</td>
<td>33.3</td>
</tr>
</tbody>
</table>

- Figures shown are upper limits of data used for training feature detectors.
Phonologies

- Festvox articulation information provided with each corpus
- Except for "schwa-like" (absent in Bengali and Nepali), all classes have at least one member in each language
  - Omitted from consideration since not uniquely contrastive in the five languages
  - Not all categories need have detectors before training low-resource language
- Lexicons also provided with phonetic transcriptions
  - Thrax G2P for each language available for out-of-vocabulary words
Articulatory feature classes

<table>
<thead>
<tr>
<th>Language features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>manner</td>
<td>&quot;stop&quot;, &quot;affricate&quot;, &quot;nasal&quot;, &quot;approximant&quot;, &quot;fricative&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;velar&quot;, &quot;postalveolar&quot;, &quot;alveolar&quot;, &quot;dental&quot;, &quot;labial&quot;, &quot;glottal&quot;, &quot;palatal&quot;</td>
</tr>
<tr>
<td>voice</td>
<td>&quot;unvoiced&quot;, &quot;voiced&quot;</td>
</tr>
<tr>
<td>height</td>
<td>&quot;close&quot;, &quot;close-mid&quot;, &quot;mid&quot;, &quot;near-open&quot;, &quot;open&quot;</td>
</tr>
<tr>
<td>length</td>
<td>&quot;short&quot;</td>
</tr>
<tr>
<td>frontness</td>
<td>&quot;front&quot;, &quot;central&quot;, &quot;back&quot;</td>
</tr>
<tr>
<td>round</td>
<td>&quot;unrounded&quot;, &quot;rounded&quot;</td>
</tr>
</tbody>
</table>

- Articulatory feature set for Bengali.
Networks

- Feature detectors à la DeepSpeech2
  - Some convolutional layers, then a series of recurrent layers, and a fully connected output layer
- Transfer learning environment: progressive networks [8]
  - Here similarly constructed pre-trained networks placed in parallel with a new network
  - Gates connect each recurrent layer of the pre-trained networks (kept constant) to their equivalent in the new network
  - (Originally developed for reinforcement learning activities)
- Input provided as log-spectrograms (20ms Hamming window, 10ms overlap); implementations in PyTorch
Feature detectors

(bidirectional GRU)

(rearrangement)
Progressive networks

(1) Baseline 1  (2) Baseline 2  (3) Baseline 3  (4) Baseline 4  (5) Progressive Net 2 columns  (6) Progressive Net 3 columns

(source task  target task  random  frozen)

(Diagram from [9].)
Setups

- Test all pairings among the five languages (16 total) as follows:
  - Train feature detectors on a 'high-resource' language (40 epochs)
  - Connect detectors to comprehensive phone recognizer for 'low-resource' language
  - Train said recognizer on varied low-resource data sizes (1h, 5h, 10h for train/test) as part of progressive network
- Evaluation based on phone-error rates
- More info on this front to come...
- Alter the gating patterns in the progressive network
- Substitute other languages than the aforementioned five as low-resource languages
- Adjust feature detector architecture
- Consider using fewer detectors in the progressive network
- ...
References


Thank you!