Hybrid CTC-Attention based End-to-End Speech Recognition using Subword Units

Zhangyu Xiao¹, Zhijian Ou¹, Wei Chu², Hui Lin²

¹Speech Processing and Machine Intelligence (SPMI) Lab, Tsinghua University, Beijing, China.

²Liulishuo Information Technology Co., Ltd. Shanghai, China.

November 2018
Contents

1. Background
   • Model structure
   • Modeling Units

2. Hybrid CTC-Attention end-to-end ASR
   • CTC & Attention
   • Experiment Results

3. Subword Units
   • Definition & example
   • Experiment Results
1. Background
   • Model structure
   • Modeling Units

2. Hybrid CTC-Attention end-to-end ASR

3. Subword Units
Background

End-to-end Speech Recognition

- a single system that directly transcribes speech signal to words
- usually based on NN structures and can be trained from scratch

Hello world!
End-to-end Speech Recognition

- A single system that directly transcribes speech signal to words
- Usually based on NN structures and can be trained from scratch

CTC based
- Makes a strong independent assumption between labels
- Estimates alignment with forward-backward algorithm

Attention based
- Attention decoder emits labels depending on previous ones
- Hard to train due to its excessively flexible attention alignments
Background——modeling units

End-to-end Speech Recognition

- a single system that directly transcribes speech signal to words
- usually based on NN structures and can be trained from scratch

- phonemes “AA, AE, ...”
- characters “a, b, c, d, ...”
- subwords “abs, ing, ...”
- words “hello, hi, ...”
Contents

1. Background

2. Hybrid CTC-Attention end-to-end ASR
   • CTC & Attention
   • Experiment Results

3. Subword Units
Hybrid CTC-Attention end-to-end ASR

CTC based model:

- makes a strong independent assumption between labels

\[
p(y|x) = \sum_{\pi \in \phi(y)} p(\pi|x) = \sum_{\pi \in \phi(y)} \prod_{l=1}^{L} q^{\pi_l}
\]

clidean cannot perform well without language model

- estimates alignment with forward-backward algorithm

 happy  easy to train and converge

Hybrid CTC-Attention end-to-end ASR

Attention based model:

- the attention decoder emits labels depending on previous ones

\[ p(y|x) = \prod_u p(y_u|h, y_{1:u}) \]

- can model label dependencies

- excessively flexible attention alignments

- hard to train and converge

---

Hybrid CTC-Attention end-to-end ASR

Hybrid CTC-Attention:
- Pyramidal BLSTM based RNN Encoder
- CTC and Attention Decoder share the same RNN Encoder

\[ L_{hybrid} = \lambda L_{CTC} + (1 - \lambda) L_{Att} \]

Hybrid CTC-Attention end-to-end ASR

Table: Results of different e2e model structures on Librispeech

<table>
<thead>
<tr>
<th>Model</th>
<th>$\lambda$</th>
<th>Word Error Rate/%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>test-clean</td>
</tr>
<tr>
<td>CTC</td>
<td>1.0</td>
<td>20.9</td>
</tr>
<tr>
<td>Attention</td>
<td>0.0</td>
<td>10.5</td>
</tr>
<tr>
<td>CTC+Attention</td>
<td>0.2</td>
<td>7.8</td>
</tr>
</tbody>
</table>

- different $\lambda$ determine different model structures
- CTC cannot perform well without a LM
- The hybrid CTC-Attention model outperforms both of CTC and Attention based models!
Contents

1. Background

2. Hybrid CTC-Attention end-to-end ASR

3. Subword units
   • Definition & example
   • Experiment Results
## Subword Units

### Table: Examples of different modeling units

<table>
<thead>
<tr>
<th>Basic Units</th>
<th>Segmented Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>that neither of them had crossed the threshold since the dark day</td>
</tr>
<tr>
<td>phoneme</td>
<td>DH AE1 T N IY1 DH ER0 AH1 V DH EH1 M HH AE1 DH K R AO1 S T DH AH0 TH R EH1 SH OW2 L D S IH1 N S DH AH0 D AA1 R K D EY1</td>
</tr>
<tr>
<td>character</td>
<td>that_nei_ther_of_them_had_crossed_the_threshold_since_the_dark_day</td>
</tr>
<tr>
<td>subword</td>
<td>that_nei_ther_of_them_had_crossed_the_threshold_since_the_dark_day</td>
</tr>
</tbody>
</table>

- phoneme based on CMUDICT
- special symbol “_” denotes word boundary
# Subword Units

## Table: Comparison of different modeling units

<table>
<thead>
<tr>
<th>Basic Units</th>
<th>Total Number</th>
<th>Length of sequence</th>
<th>Ability of handling OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>$N \times 10^{4\sim5}$</td>
<td>shortest/12</td>
<td>NO</td>
</tr>
<tr>
<td>phoneme</td>
<td>$N \times 10$</td>
<td>Long/41</td>
<td>NO</td>
</tr>
<tr>
<td>character</td>
<td>$N \times 10$</td>
<td>Longest/66</td>
<td>YES</td>
</tr>
<tr>
<td>subword</td>
<td>$N \times 10^{2\sim3}$</td>
<td>Short/22</td>
<td>YES</td>
</tr>
</tbody>
</table>

Numbers in length of sequence:
takes the utterance of the former page as example

- **Large total number**
  - heavy computation cost due to softmax
  - label sparenness

- **Long output seq**
  - difficult to capture word-level dependency
  - easy to generate substitution error

- **Fixed dictionay**
  - unable to handle the Out-Of-Vocabulary problem
# Subword Units

## Table: Comparison of different modeling units

<table>
<thead>
<tr>
<th>Basic Units</th>
<th>Total Number</th>
<th>Length of sequence</th>
<th>Ability of handling OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>$N \times 10^{4\sim5}$</td>
<td>shortest/12</td>
<td>NO</td>
</tr>
<tr>
<td>phoneme</td>
<td>$N \times 10$</td>
<td>Long/41</td>
<td>NO</td>
</tr>
<tr>
<td>character</td>
<td>$N \times 10$</td>
<td>Longest/66</td>
<td>YES</td>
</tr>
<tr>
<td>subword</td>
<td>$N \times 10^{2\sim3}$</td>
<td>Short/22</td>
<td>YES</td>
</tr>
</tbody>
</table>

Numbers in length of sequence:

takes the utterance of the former page as example

- Large total number ➢ heavy computation cost due to softmax ➢ label sparness
- Long output seq ➢ difficult to capture word-level dependency ➢ easy to generate substitution error
- Fixed dictionary ➢ unable to handle the Out-Of-Vocabulary problem
Subword Units: Generation & Segmentation

Subword Generation Algorithm: Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>BPE Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1. Initialize subword set $S$ with 26 characters and word boundary symbol “_”: $S = {a, b, c, \ldots, z, _}$</td>
</tr>
<tr>
<td>Step 2. Count all symbol pairs, and find the most frequent pair $(c^1, c^2)$</td>
</tr>
<tr>
<td>Step 3. Merge the most frequent pair to a new symbol “$c^1c^2$”, and add it to $S$</td>
</tr>
<tr>
<td>Step 4. If $</td>
</tr>
<tr>
<td>Step 5. Output the final subword set $S$ of size $N$.</td>
</tr>
</tbody>
</table>
# Subword Units: Experiment Results

Table: Experiments on Librispeech 1000h Dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Basic unit</th>
<th>$\lambda$</th>
<th>Word Error Rate/%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>test-clean</td>
</tr>
<tr>
<td>CTC</td>
<td>char</td>
<td>1.0</td>
<td>20.9</td>
</tr>
<tr>
<td>Att</td>
<td>char</td>
<td>0.0</td>
<td>10.5</td>
</tr>
<tr>
<td>CTC+Att</td>
<td>char</td>
<td>0.2</td>
<td>7.8</td>
</tr>
<tr>
<td>CTC+Att</td>
<td>subword</td>
<td>0.2</td>
<td>6.8</td>
</tr>
</tbody>
</table>

- Significant improvement from character to subword:
  - Relatively 12.8% WER reduction
  - Mostly from substitution error

<table>
<thead>
<tr>
<th>Basic Unit</th>
<th>WER</th>
<th>Sub</th>
<th>Del</th>
<th>Ins</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>7.8</td>
<td>6.4</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>subword</td>
<td>6.8</td>
<td>5.4</td>
<td>0.5</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Subword Units: Experiment Results

Figure 1: Influence of $\lambda$

- CTC should form a small proportion in the hybrid loss

Figure 2: Influence of subword number

- Number of subword units should not be too large nor too small.
Thank you!
Any Questions?