Promising Accurate Prefix Boosting for Sequence-to-sequence ASR

Karthick Baskar, Lukáš Burget, Shinji Watanabe and Martin Karafiat
• What is Prefix?
  - In the context of ASR, prefix refers to a partial sequence
• **What is Prefix?**
  - In the context of ASR, prefix refers to a partial sequence

• **Why boost accurate prefix??**
  - Training by boosting correct prefixes (accurate) over wrong prefixes will help model to rectify its own errors
Encoder - Decoder

- **Encoder:**
  - recurrent layers
  - entire input sequence to fixed-length vector

- **Decoder:**
  - recurrent layers with final softmax layer
  - predict probability for the next symbol of the output sequence in an auto-regressive fashion
  - learns an implicit language model for the output sequences
Problem Overview

- **Exposure bias**
  - **Training**: output character is conditioned on the previous true character
  - **Testing**: the model needs to rely on its own previous predictions

- **Error criterion mismatch**
  - **Training**: the objective is the conditional maximum likelihood (cross entropy) for maximizing the probability of the correct sequence
  - **Testing**: Character error rate (CER) or word error rate (WER)
**Training**: Minimize cross-entropy loss of each target token $y_i^*$ (character)

$$\log p(y^*|X) = \sum_i \log p(y_i^*|X)$$

**Teacher-forcing**: Feed previous token from ground-truth as auxiliary info to predict current token
Decoding:

- Previous token from hypothesis is fed to predict current token
- Output sequence is predicted in two ways
  - Greedy (argmax) search
  - Beam search

True seq: ABB
Argmax seq: BAB

Mismatch during train and decode
Mismatch during train and decode

Hypothesis

True seq : ABB
Argmax seq : BAB
Modify training procedure ??

Decrease the training loss for the predicted paths !!

Training is matched to testing

Is there a technique to train only with predictions as previous tokens ??
Decoding:

- Previous token from hypothesis is fed to predict current token
- Output sequence is predicted in two ways
  - Greedy (argmax) search
  - Beam search
How to match beam-search decoding with training??

- Need to consider multiple hypotheses generated during beam-search.
- Training objective must keep prefix at top of the beam.
- Helps to survive pruning by keeping scores higher in the beam.

Beam width $= 3$

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search
- Training objective must keep prefix at top of the beam
- Helps to survive pruning by keeping scores higher in the beam
How to match beam-search decoding with training ??

- Need to consider multiple hypothesis generated during beam-search
- Training objective must keep prefix at top of the beam
- Helps to survive pruning by keeping scores higher in the beam
How to match beam-search decoding with training??

• Need to consider multiple hypothesis generated during beam-search
• Training objective must keep prefix at top of the beam
• Helps to survive pruning by keeping scores higher in the beam

Beam width = 3

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search.
- Training objective must keep prefix at top of the beam.
- Helps to survive pruning by keeping scores higher in the beam.

Prefixes that participate in loss:

Beam width = 3:

- P I T
- T A C
- S E K

Correct prediction:
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search.
- Training objective must keep prefix at top of the beam.
- Helps to survive pruning by keeping scores higher in the beam.

Prefixes that participate in loss

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search.
- Training objective must keep prefix at top of the beam.
- Helps to survive pruning by keeping scores higher in the beam.

Prefixes that participate in loss:

 Beam width = 3

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search
- Training objective must keep prefix at top of the beam
- Helps to survive pruning by keeping scores higher in the beam

Prefixes that participate in loss

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search
- Training objective must keep prefix at top of the beam
- Helps to survive pruning by keeping scores higher in the beam

Prefixes that participate in loss
How to match beam-search decoding with training??

- Need to consider multiple hypothesis generated during beam-search
- Training objective must keep prefix at top of the beam
- Helps to survive pruning by keeping scores higher in the beam

Prefixes that participate in loss

Correct prediction
How to match beam-search decoding with training??

- Need to consider multiple hypotheses generated during beam-search.
- Training objective must keep prefix at top of the beam.
- Helps to survive pruning by keeping scores higher in the beam.

Prefixes that participate in loss:

Beam width = 3

Correct prediction:
Choose weights

score of true label is better than predicted label by a specific margin

\[ \mathcal{L}_{MM} = \sum_{i} - s(y_i^*, X) + \max_{y} (s(y, X) + \alpha \text{Acc}(y_i^*, y)) \]
Maximum margin objective

Choose weights

score of true label is better than predicted label by a specific margin

weight \cdot (true \ label \ score) \geq (Margin) + weight \cdot (scores \ of \ other \ labels)

\mathcal{L}_{MM} = \sum_{l} - s(y_l^*, X) + \max_{y} (s(y, X) + \alpha \text{Acc}(y_l^*, y))

- True label score
- Predicted label score
- Margin
Maximum margin objective

Choose weights

score of true label is better than predicted label by a specific margin

weight \cdot (true\ label\ score) \geq (Margin) + \ weight \cdot (scores\ of\ other\ labels)

\[ \mathcal{L}_{MM} = \sum_{i} - s(y_i^*, X) + \max_{y} (s(y, X) + \alpha \text{Acc}(y_i^*, y)) \]

Label → Prefix
Maximum margin objective

Choose weights

score of true label is better than predicted label by a specific margin

weight \cdot (true \ label \ score) \geq (Margin) + weight \cdot (scores \ of \ other \ labels)

\[
\mathcal{L}_{MM} = \sum_{i} - s(y_i^*, X) + \max_{y} (s(y, X) + \alpha \text{Acc}(y_i^*, y))
\]

Label → Prefix

Better for training the encoder-decoder because they contain more informative training signals at each step
Choose weights

score of true label is better than predicted label by a specific margin

weight \cdot (\text{true label score}) \geq (\text{Margin}) + weight \cdot (\text{scores of other labels})

\[ L_{MM} = \sum_l -s(y_l^*, X) + \max_y (s(y, X) + \alpha \text{Acc}(y_l^*, y)) \]

**Label → Prefix**

Better for training the encoder-decoder because they contain more informative training signals at each step

\[ L_{MM} = \sum_l -s(y_{1:l}^*, X) + \max_y (s(y_{1:l}, X) + \alpha \text{Acc}(y_{1:l}^*, y_{1:l})) \]
Promising accurate prefix boosting (PAPB)

- Hard maximum is replaced by soft maximum "softmax" \((\log\sum \exp)\)
- Softmax margin* showed noticeable gains over max margin empirically

\[ \mathcal{L}_{SM} = \sum_{l} - s(y_{1:l}^*, X) + \log(\sum_{y} \exp(s(y_{1:l}, X) + \alpha \text{Acc}(y_{1:l}^*, y_{1:l}))) \]

- Generalization of boosted MMI (bMMI) criterion

* K. Gimpel and N. A. Smith, “Softmax-margin training for structured log-linear models,” 2010
Promising accurate prefix boosting (PAPB)

<table>
<thead>
<tr>
<th></th>
<th>Token</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE (SS – greedy search)</td>
<td>51%</td>
<td></td>
</tr>
<tr>
<td>CE (SS – beam search)</td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>Maximum margin</td>
<td>49%</td>
<td></td>
</tr>
<tr>
<td>Softmax margin</td>
<td>47%</td>
<td></td>
</tr>
</tbody>
</table>
% WER on held-out set with PAPB

% WER by varying beam-width

% WER

Training beam width

- 2
- 5
- 10
- 12
- 15
Comparison with sequence-level objective

- Sequence-level optimization technique: Minimum Bayes Risk Criterion*

\[ \mathcal{L}_{MBR} = E_{p(y|x)} [\text{Acc}(y^*, y)] = \sum_{y \in Y} p(y|X) \text{Acc}(y^*, y) \]

- Obtain sequence predictions from model distribution and backpropagate a sequence-level objective

- \( Y \) denotes the N-best sequences selected using beam search

CER on held-out set with PAPB

%CER on validation set of Voxforge-Italian

- MBR
- Softmargin
- prefix

# Epochs

CER

0.125
0.13
0.135
0.14
0.145
0.15

1 2 3 4 5
## Impact of pretraining and CE regularization

<table>
<thead>
<tr>
<th>CE</th>
<th>Pretraining</th>
<th>MBR (%WER)</th>
<th>% Rel. drop</th>
<th>PAPB (%WER)</th>
<th>% Rel. drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>11.5</td>
<td>-</td>
<td>10.8</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Hard to train</td>
<td>-</td>
<td>14.9</td>
<td>27.5</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>13.8</td>
<td>16.7</td>
<td>11.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>
### Impact of pretraining and CE regularization

<table>
<thead>
<tr>
<th>CE</th>
<th>Pretraining</th>
<th>MBR (%WER)</th>
<th>% Rel. drop</th>
<th>PAPB (%WER)</th>
<th>% Rel. drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>11.5</td>
<td>-</td>
<td>10.8</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Hard to train</td>
<td>-</td>
<td>14.9</td>
<td>27.5</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>13.8</td>
<td>16.7</td>
<td>11.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>

- **Pretraining is crucial** for sequence-level objective such as MBR training
### Impact of pretraining and CE regularization

<table>
<thead>
<tr>
<th>CE</th>
<th>Pretraining</th>
<th>MBR (%WER)</th>
<th>% Rel. drop</th>
<th>PAPB (%WER)</th>
<th>% Rel. drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>11.5</td>
<td>-</td>
<td>10.8</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Hard to train</td>
<td>-</td>
<td>14.9</td>
<td>27.5</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>13.8</td>
<td>16.7</td>
<td>11.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>

- **Pretraining is crucial** for sequence-level objective such as MBR training
- **PAPB did show convergence without pretraining**
### Impact of pretraining and CE regularization

<table>
<thead>
<tr>
<th>CE</th>
<th>Pretraining</th>
<th>MBR (%WER)</th>
<th>% Rel. drop</th>
<th>PAPB (%WER)</th>
<th>% Rel. drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>11.5</td>
<td>-</td>
<td>10.8</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Hard to train</td>
<td>-</td>
<td>14.9</td>
<td>27.5</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>13.8</td>
<td>16.7</td>
<td>11.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>

- **Pretraining is crucial** for sequence-level objective such as MBR training
- **PAPB did show convergence without pretraining**
- **CE regularization provides 6.1 % and 16.7% relative gain for PAPB and MBR**
## Recognition performance on WSJ corpus

Effect of LM on token level, sequence level and prefix (partial sequence) level training

<table>
<thead>
<tr>
<th>Model type</th>
<th>No RNNLM</th>
<th>Character RNNLM</th>
<th>Word RNNLM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%CER</td>
<td>%WER</td>
<td>%CER</td>
</tr>
<tr>
<td>CE</td>
<td>4.6</td>
<td>12.9</td>
<td>2.5</td>
</tr>
<tr>
<td>MBR</td>
<td>4.3</td>
<td>11.5</td>
<td>2.5</td>
</tr>
<tr>
<td>PAPB</td>
<td>4.0</td>
<td>10.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Deep-CNN*</td>
<td>-</td>
<td>10.5</td>
<td>-</td>
</tr>
<tr>
<td>OCD*</td>
<td>-</td>
<td>9.6</td>
<td>-</td>
</tr>
<tr>
<td>LF-MMI*</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Effect of LM on **token** level, **sequence** level and prefix (**partial sequence**) level training

<table>
<thead>
<tr>
<th>Model type (%WER)</th>
<th>No RNNLM</th>
<th></th>
<th>Word RNNLM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>test-clean</td>
<td>test-other</td>
<td>test-clean</td>
<td>test-other</td>
</tr>
<tr>
<td>CE</td>
<td>6.7</td>
<td>21.5</td>
<td>4.0</td>
<td>12.7</td>
</tr>
<tr>
<td>MBR</td>
<td>5.5</td>
<td>17.4</td>
<td>3.7</td>
<td>11.3</td>
</tr>
<tr>
<td>PAPB</td>
<td>4.7</td>
<td>15.1</td>
<td>3.1</td>
<td>9.8</td>
</tr>
<tr>
<td>OCD*</td>
<td>4.5</td>
<td>13.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LF-MMI*</td>
<td>-</td>
<td>-</td>
<td>3.8</td>
<td>8.7</td>
</tr>
</tbody>
</table>

* https://github.com/kaldi-asr/kaldi/blob/master/egs/librispeech/s5/local/chain/tuning/run_tdnn_1d.sh
Conclusion and Constraints

• Prefix boosting with softmax-margin objective provides considerable gains
• Effective compared to sequence-level MBR objective
• Beam-search is not an efficient method to run with GPU
• 2-fold increase in training time
• Constraint in setting larger training beam-size
• Future work will be to use sampling approach instead of beam-search
References


