Librispeech Transducer Model with Internal Language Model Prior Correction

Albert Zeyer\(^1,2\), André Merboldt\(^1\), Wilfried Michel\(^1,2\), Ralf Schlüter\(^1,2\), Hermann Ney\(^1,2\)

\(^1\)Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, 52062 Aachen, Germany, \(^2\)AppTek GmbH, 52062 Aachen, Germany

\{zeyer, michel, schluter, ney\}@cs.rwth-aachen.de, andre.merboldt@rwth-aachen.de

Abstract

We present our transducer model on Librispeech. We study variants to include an external language model (LM) with shallow fusion and subtract an estimated internal LM. This is justified by a Bayesian interpretation where the transducer model prior is given by the estimated internal LM. The subtraction of the internal LM gives us over 14\% relative improvement over normal shallow fusion. Our transducer has a separate probability distribution for the non-blank labels which allows for easier combination with the external LM, and easier estimation of the internal LM. We additionally take care of including the end-of-sentence (EOS) probability of the external LM in the last blank probability which further improves the performance. All our code and set ups are published.

Index Terms: transducer, language model integration, speech recognition

1. Introduction & Related Work

The recurrent neural network transducer (RNN-T) model [1, 2] is an end-to-end model which allows for time-synchronous decoding, which is a more natural fit for many applications such as online recognition. Thus RNN-T and many variations has recently gained interest [3–8].

In a Bayesian interpretation, a discriminative acoustic model \(p_{\text{AM}}(y \mid x)\) can be combined with an external language model \(p_{\text{LM}}(y)\) by

\[
p(y \mid x) = \frac{p_{\text{AM}}(y \mid x)}{p_{\text{AM}}(y)} \cdot p_{\text{AM}}(x) \cdot p_{\text{LM}}(y) \cdot \frac{1}{p(x)}.
\]

In recognition, when searching for \(\arg\max_{x} p(y \mid x)\), we can omit \(p(x)\) and \(p_{\text{AM}}(x)\). In shallow fusion, \(p_{\text{AM}}(y)\) is omitted as well. In the density ratio approach [9], \(p_{\text{AM}}(y)\) is estimated by a separate language model trained on just the acoustic training transcriptions. In the hybrid autoregressive transducer (HAT) [6], \(p_{\text{AM}}(y)\) is estimated directly based on the implicit internal LM (ILM) of \(p_{\text{AM}}(y \mid x)\). The HAT model has a particular simple architecture which was designed such that there is a simple approximation for this ILM estimation by setting the encoder input to 0. We follow up on the ILM estimation approach and try some variations of the estimation. Using 0 as encoder input also works but we found some other variations to be better.

2. Model

We follow a transducer variant as defined in [7]. The whole model can be seen in Figure 1. Let \(x_{1}^{T}\) be the acoustic input features (MFCC in our case), and \(y_{1}^{S}\) some label sequence over labels \(\Sigma\) (excluding blank \(\epsilon\)). We use BPE subword units [10, 11] with a vocabulary size of about 1000 labels.

We have a multi-layer bidirectional LSTM [12] encoder model with interchanged max-pooling in time to downscale the input to length \(T\) with factor 6. This results in

\[
h_{1}^{T} := \text{Encoder}(x_{1}^{T^6}).
\]

We define the probability for the sequence \(y_{1}^{S}\) as

\[
p(y_{1}^{S} \mid x_{1}^{T}) := \sum_{\alpha_{1}^{U} y_{1}^{S}} p(\alpha_{1}^{U} \mid x_{1}^{T}),
\]

\[
p(\alpha_{1}^{U} \mid x_{1}^{T}) := \prod_{n=1}^{U} p(\alpha_{n} \mid \alpha_{n-1}, x_{1}^{T}),
\]

with \(\alpha_{n} \in \Sigma' := \Sigma \cup \{\epsilon\}\), and where \(\alpha_{1}^{U} : y_{1}^{S}\) is defined by the label topology \(\mathcal{A}\). Specifically, we use \(U = T + S\) and allow all sequences \(\alpha_{1}^{U}\) which match the sequence \(y_{1}^{S}\) after removing all blanks \(\epsilon\), also noted as \(\mathcal{A}(\alpha_{1}^{U}) = y_{1}^{S}\). This defines an alignment between \(h\) and \(y\) as can be seen in Figure 2.

Our decoder model defines the probability distribution over

\footnote{In later work on attention-based encoder decoder models, we used 10k BPE labels for Librispeech. However, because of computation time and memory constraints, we reduced it to 1k for the transducer model.}
3. Training

The loss is defined as

$$L := - \log p(y^S_1 | x^T_1) = - \log \sum_{x^T_1 \in U} p(\alpha^U_1 | x^T_1).$$

As we use the simplified transducer model where Readout does not depend on $\alpha_{u-1}$, we can efficiently calculate the exact full sum over all alignments $\alpha^U_1 : y^S_1$, and do not need the maximum approximation [7].


We use the Adam optimizer [15] with learning rate scheduling based on cross validation scores. Additionally, we re-set the learning rate back to the initial value after a larger number of epochs, after the model already converged, and start over with the learning rate scheduling. We train for 128 or 192 epochs. The long amount of training had a huge effect on the overall performance.

3.1. Pretraining

We use a pretraining scheme where we schedule multiple aspects of the training:

- We grow the encoder from 3 layers with 500 dim. up to 6 layers with 1000 dimensions [16].
- We increase the dropout rates.
- We use curriculum learning and start with shorter sequences initially.
- We use linear learning rate warmup from 0.0001 to 0.001.
- We use a higher initial time reduction factor 20 in the encoder and reduce it to the final factor 6.

3.2. Distributed Multi-GPU Training

Our distributed training implementation uses independent trainer (worker) instances per GPU. Each worker independently loads the dataset. To make sure that every worker uses a different part of the dataset, it is common to use striding. Striding has the disadvantage that it is very IO intensive in this setting where every worker loads the dataset independently and often becomes the bottleneck. So we came up with the idea to use a different random seed for the shuffling of the dataset for every worker, to replace the striding. This greatly improved the IO in our case and made the training much faster.

Additionally, every worker independently trains an own copy of the model for multiple update steps, until the models get synchronized by averaging the parameters over all workers. We made the further improvement that we do not synchronize after a fixed number of steps, but instead after a fixed time interval. A fixed number of steps implies that the training is always as slow as the slowest worker, and variations in the runtime often lead to some workers being slower than others even on same hardware. Synchronizing after a fixed time interval does not have this problem, while being more stochastic.

We synchronize only after 100 seconds to reduce the communication between workers. The workers can potentially be on different computing nodes and might need to communicate over network, which can result in 1-2 seconds for the synchronization. We train on either 8 or 16 GPUs.

4. Decoding & Language Model Combination

Our beam search decoding tries to find the sequence $\hat{y}^S_1$ given $x^T_1$ which maximizes the probability, i.e. specifically

$$x^T_1 \rightarrow \hat{S}, \hat{y}^S_1 := \arg \max_{\alpha^U \in \Sigma} \log p(\alpha^U_1 | x^T_1) \approx A \circ \arg \max_{\alpha^U \in \Sigma} \log p(\alpha^U_1 | x^T_1)$$

We perform alignment-synchronous decoding, i.e. all hypotheses are in the same alignment step $u$ when being pruned [7, 17]. We merge hypotheses by summing their scores when they correspond to the same word sequence after BPE-merging.

The training recipe for our BPE-10K LSTM LM [18] has been adapted for the new BPE-1k label set but otherwise no changes have been made. Shallow fusion (SF) [19] is a logarithmic combination of the log-scores of external LM and ASR model scores during the recognition process, with scale $\beta$ for the LM and scale $\lambda$ for the acoustic (non-blank) label probability $q$, while we do not add an own scale for $p(\Delta t)$. Specifically,
we use the score

\[
\log P^\text{sf}(\alpha_u | ... ) := \begin{cases} 
\log P_u(\Delta t_u = 1 | ... ), & \alpha_u = \epsilon, \\
\log P_u(\Delta t_u = 0 | ... ) + \lambda \cdot \log q_u(\alpha_u | ... ) + \beta \cdot \log p_{\text{ILM}}(\alpha_u | ... ), & \alpha_u \in \Sigma 
\end{cases}
\]

We experimented with fixing the label scale at $\lambda = 1$ or $\lambda = 1 - \beta$.

Inspired by [6, 9, 20] we also tried to subtract the internal LM log score. It assumes that we can factorize our model into a language and acoustic model. Although our model is not directly formulated as such, we can approximate the internal language model. For that we used the estimated score $\log p_{\text{ILM}}$ as shown in Section 4.1, where we use the average of encoder features in the time dimension.

\[
\log P^\text{sf,ILM}(\alpha_u | ... ) := \begin{cases} 
\log P_u(\Delta t_u = 1 | ... ), & \alpha_u = \epsilon, \\
\log P_u(\Delta t_u = 0 | ... ) + \lambda \cdot \log q_u(\alpha_u | ... ) + \beta \cdot \log p_{\text{ILM}}(\alpha_u | ... ) - \gamma \cdot \log p_{\text{ILM}}(\alpha_u | ... ), & \alpha_u \in \Sigma 
\end{cases}
\]

4.1. Internal LM Estimation

The transducer is trained on audio-text pairs but learns an implicit prior model on the text. This is explicitly given by the context dependency on previous labels. In this transducer case, the SlowRNN is also explicitly modeled such that it models the most important part of this prior as it operates only on the text-only part and runs label-synchronous. This prior is an implicit internal LM in our acoustic model

\[
P_{\text{prior}}(y) = \sum_x P_{\text{AM}}(y | x) \cdot P(x)
\]

which can not be calculated efficiently in general. To approximate the internal LM, we replace the encoder input to the rest of the model (Readout). We either use a 0 vector or the encoder mean (avg). The mean is computed over the time dimension for each sequence separately.

We evaluate the estimated internal LM on text-only data. In Table 1, the BPE-level perplexities (PPL) are shown and compared against the LSTM LM which was trained only on text data, but without any overlap to the audio transcriptions [22].

4.2. EOS Modelling

In contrast to language models or attention models, transducers and models with explicit time modeling do not have to model the end-of-sentence/sequence explicitly with an additional token (denoted as $<\text{eos}>$). Instead the search ends when all input frames have been consumed. However, for LM integration, when only considering actual output symbols, the information about when the sequence should end is not considered. This additional information is usually ignored in the literature, however it provides valuable information to the search process.

Our approach is to combine the LM EOS probability with $p_u(\Delta t = 1)$ ($\epsilon$) in the last time frame ($t_u = T$) because that determines the EOS in the transducer.

\[
\log p^\text{sf,ILM,EOS}(\alpha_u | ... ) := \begin{cases} 
\log p_u(\Delta t = 1 | ... ), & \alpha_u = \epsilon, t_u-1 < T \\
\delta_{\text{eos}}(\epsilon) \log p_u(\Delta t = 1 | ... ) + \beta_{\text{eos}} \log p_{\text{ILM}}(\epsilon | ... ), & \alpha_u = \epsilon, t_u-1 = T 
\end{cases}
\]

Usually $\delta_{\text{eos}} = \beta_{\text{eos}} = 0.5$ yielded good performance, although it was not tuned properly.

5. Experiments

We perform experiments on LibriSpeech [22]. Our model training and decoding is implemented in RETURNN [23], based on TensorFlow [24]. The distributed multi-GPU training is implemented with Horovod [25]. We make use of Mingkun Huang’s warp-transducer loss implementation\(^5\). Our decoder uses the built-in RETURNN features for stochastic variables and searches over $\alpha_u$. This uses GPU-based batched one-pass decoding with the external LM and internal LM subtraction. We publish all the configuration files needed to reproduce the experiments\(^5\).

We have a variety of different exponent scales for our log-linear modeling, as well as additional parameters for EOS-modeling. Label scale $\lambda$ which is set to either $\lambda = 1$ or $\lambda = 1 - \beta$, the emission model scale $\delta$, and the scales for external and internal LM $\beta$, $\gamma$, respectively. Additionally for EOS-modeling $\delta_{\text{eos}}$ and $\beta_{\text{eos}}$ is used, although it was fixed to $\delta_{\text{eos}} = \beta_{\text{eos}} = 0.5$. The scaling factors $\beta$ and $\gamma$ have to be tuned jointly on a held-out dataset, as can be seen in Fig. 3, with $\lambda = 1 - \beta$. They were tuned separately for each subset dev-clean and dev-other. Results for LM integration are presented in Table 2 and in Table 3 with additional EOS-modeling. With shallow fusion of just the LM we already see a significant WER improvement by over 22% relative. When additionally subtracting the internal LM, a further significant improvement by over 14% relative over the shallow fusion.

---

\(^2\)https://github.com/HawkAaron/warp-transducer

\(^3\)https://github.com/ruth-16/returnn-experiments/tree/master/2021-transducer

---

Table 1: Perplexity and WER measurements on Librispeech dev-other of a transducer model. Note that the BPE-level (1k units) perplexity is evaluated without the EOS token, since the transducer has no explicit end-of-sequence symbol. Compared are both setting the encoder ($h$) to 0 and to the mean over the time-dimension (avg). The LSTM and Trafo-LM are trained on text-only data without overlap to the audio transcriptions [22].

<table>
<thead>
<tr>
<th>Model</th>
<th>Epochs</th>
<th>Perplexity (avg)</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transducer BPE-1K</td>
<td>8</td>
<td>82.76</td>
<td>67.47</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>49.32</td>
<td>38.99</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>45.13</td>
<td>32.86</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>46.53</td>
<td>31.94</td>
</tr>
<tr>
<td></td>
<td>133</td>
<td>47.05</td>
<td>31.37</td>
</tr>
<tr>
<td>LSTM LM</td>
<td>20</td>
<td>15.40</td>
<td></td>
</tr>
<tr>
<td>Trafo LM</td>
<td>39</td>
<td>14.44</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3: Tuning of LM scales of a transducer with EOS-modeling. The baseline without an external LM has 8.66\% WER on dev-other with a beam size of 24, $\lambda = 1 - \beta$.

Table 2: We investigate the effect of LM integration for the model with either shallow fusion (SF) or additional internal LM (ILM) subtraction. All experiments were conducted with a fixed beam-size 24 and without EOS-modeling; the LSTM-ILM has a BPE-1K level perplexity of 15.4 on dev-other:

<table>
<thead>
<tr>
<th>Method</th>
<th>Label scale $\lambda$</th>
<th>WER [%] dev</th>
<th>WER [%] other</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>$\lambda = 1$</td>
<td>3.22</td>
<td>8.76</td>
</tr>
<tr>
<td>LSTM</td>
<td>$\lambda = 1 - \beta$</td>
<td>2.47</td>
<td>6.99</td>
</tr>
<tr>
<td>SF</td>
<td>---</td>
<td>2.29</td>
<td>5.63</td>
</tr>
<tr>
<td>SF-ILM</td>
<td>---</td>
<td>2.22</td>
<td>5.32</td>
</tr>
</tbody>
</table>

The avgLM estimation seems to be better than (except on test-other. The effect of EOS gives us 7\% relative improvement. We also test a stronger Transformer LM in Table 1 (perplexities in Table 1) and see further improvement.

5.1. Error Analysis

One of the sources of errors when looking at an entire system are the errors the model made when its prediction was wrong. We look at the percentages of substitution, deletion, and insertion errors of the word error rate (WER). Especially interesting is the comparison between different models and their respective LM integration. Also of interest are how long the hypothesized sentences are, relative to the reference transcription. The transducer seems to model the hypothesis length better than hybrid (without rescoring) and attention-based models, although adding an external LM seems to help the attention model. Overall, we can see that introducing the external LM helps with substitutions and insertion errors, while the deletions actually increase. In comparison to the attention-based model, the transducer model has significantly less insertion errors, but more deletion errors, relative to the overall WER.

6. Conclusions & Future Work

The subtraction of the ILM helped to improve the model by a lot (over 14\% relative) over the already strong shallow fusion. The EOS modeling also helped (7\% relative). We noticed that all recognition experiments are very sensitive to the LM/ILM scales. The long training time also had a huge effect on the final performance.

As future work, we plan to study the effect of the label unit and to test simple characters and other subword variations, similar to [28]. The encoder model might get improvements by more recent advancements [5]. The decoder can be extended as well [7]. We also can potentially improve the ILM estimation. Finally, we expect to get improvements by min. WER training.

7. Acknowledgements

We thank Yingbo Gao for providing us with the Transformer LM on our BPE-1k label set. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement n\^o 694537, project ‘SEQCLAS’). The work reflects only the authors’ views and the European Research Council Executive Agency (ERCEA) is not responsible for any use that may be made of the information it contains. This work was partly funded by the Google Focused Award “Pushing the Frontiers of ASR: Training Criteria and Semi-Supervised Learning”.

8. References


