EXPLORING A ZERO-ORDER DIRECT HMM BASED ON LATENT ATTENTION FOR AUTOMATIC SPEECH RECOGNITION

Parnia Bahar¹,², Nikita Makarov¹, Albert Zeyer¹,², Ralf Schlüter¹,² and Hermann Ney¹,²

¹Human Language Technology and Pattern Recognition Group, Computer Science Department RWTH Aachen University, 52074 Aachen, Germany, ²AppTek GmbH, 52062 Aachen, Germany

{bahar, zeyer, schlueter, ney}@cs.rwth-aachen.de, nikita.makarov@rwth-aachen.de

ABSTRACT
In this paper, we study a simple yet elegant latent variable attention model for automatic speech recognition (ASR) which enables an integration of attention sequence modeling into the direct hidden Markov model (HMM) concept. We use a sequence of hidden variables that establishes a mapping from output labels to input frames. Inspired by the direct HMM model, we assume a decomposition of the label sequence posterior into emission and transition probabilities using zero-order assumption and incorporate both Transformer and LSTM attention models into it. The method keeps the explicit alignment as part of the stochastic model and combines the ease of the end-to-end training of the attention model as well as an efficient and simple beam search. To study the effect of the latent model, we qualitatively analyze the alignment behavior of the different approaches. Our experiments on three ASR tasks show promising results in WER with more focused alignments in comparison to the attention models.

Index Terms— End-to-end speech recognition, Latent models, direct HMM, Attention, Transformer, LSTM

1. INTRODUCTION & RELATED WORKS
In automatic speech recognition (ASR), label sequence posterior distribution should cover the alignment information between input and output sequences either implicitly or explicitly. Deep neural network (DNN) hybrid hidden Markov model (HMM) [1,2] and its variants such as connectionist temporal classification (CTC) [3], recurrent neural transducer [4] and recurrent neural aligner [5] tackle the underlying alignment problem using explicit stochastic latent variables. Similarly, in direct HMM [6] and segmental methods [7] alignments are handled using latent variables by assuming a direct alignment of labels to observations, instead of aligning each input frame to a label.

Contrary to the above approaches, sequence-to-sequence models such as 2D recurrent [8] and attention models cover the alignment information implicitly [9],[12]. The latter applies a soft attention mechanism where the (implicit) alignment is not a fully stochastic process. In contrast to the HMM-based method, where a transparent process defines the correspondence between inputs and outputs, it is often challenging for the fuzzy attention mechanism to extract a comprehensible alignment. A weighted sum of the input states can be seen as a kind of partial decision on the local average observation (encoder states) where the input does not directly influence the final output prediction. Hence, it is often hard to determine the contribution of the alignment on the output. An explicit latent variable can be used to define an alignment sequence for such models. A latent alignment can have no dependence on its history (zero-order) [13],[15] with an extension of the prior into an explicit posterior distribution [16], or it can depend on its predecessor (first-order) [7],[17],[20]. Another implicit way is including dependence on previous attention weights while computing the current ones [21],[22].

In this work, we incorporate the attention model into the direct HMM formulation similar to [14],[17]. We study a latent attention model that has a direct dependency on the input/temporal positions. Since marginalization is exponential in the alignment dependency order of the model, we only explore a zero-order assumption. The marginalization over the latent variable becomes simple and efficient for zero-order models and can be easily applied in both training and decoding. In this case, no dynamic programming and no search for the alignment path is required, thus a simple beam search decoder is used. The model results in more focused alignments, leading to better explainability. We also compare this approach to the attention mechanism from the statistical and modeling perspective.

2. BACKGROUND
In ASR, given an input sequence $x_1^T$ of length $T$, with $x_t$ being an observation at time $t$, the posterior probability of a label sequence (e.g. words, characters, etc.) $w_1^N$, with unknown length $N$, is defined using the chain rule:

$$p(w_1^N | x_1^T) = \prod_{n=1}^{N} p(w_n | w_{n-1}^{n-1}, x_1^T)$$

This conditional posterior covers the alignments between input and output sequences either implicitly or explicitly. In recent end-to-end sequence-to-sequence models, including both the long short-term memory (LSTM)-based [23] attention [9] and the Transformer [24], the dependence on $x_1^T$ is modeled using an attention mechanism, where a weighted sum of the input representations is used as an intermediate step inside the network layer. The attention mechanism gives a distribution
over the input positions that is used to compute a weighted average of the observations. This distribution can be considered as an implicit probabilistic notion of alignment.

\[ p(w_1^N | x_T^T) = \prod_{n=1}^N p(w_n | w_1^{n-1}, c_n(w_1^{n-1}, h_1^{T'}(x_T^T))) \]  

(1)

where \( h_1^{T'} \) is the encoder’s output with downsampling \((T' < T)\), and \( c_n \) is the output of an attention approach at label position \( n \), summarizing the alignment information implicitly.

\[ c_n(w_1^{n-1}, h_1^{T'}(x_T^T)) = \sum_{t=1}^{T'} \alpha_{n,t}(w_1^{n-1}, h_1^{T'}(x_T^T))h_t \]

with \( \sum_{t=1}^{T'} \alpha_{n,t}(w_1^{n-1}, h_1^{T'}(x_T^T)) = 1 \). Here, \( \alpha_{n,t} \) is the normalized attention weight that relates the label position \( n \) and input position \( t \). As expressed in these equations, there is no direct dependence on input positions, and the alignment distribution does not directly influence the label sequence posterior. Since the number of input frames may vary strongly between labels, the summarization of the input representations into a single vector might be not suitable.

3. LATENT ATTENTION MODEL

Similar to the direct HMM model [6], we introduce alignments as a sequence of stochastic latent variables \( t_1^n \) that establishes a mapping from position \( n \) on the label sequence to position \( t \) on the input sequence, i.e., \( n \to t_n \), and marginalize out all possible alignments in Eq. 2 such that \( t_n \) aligns the output position \( n \) to the input position \( t \). In the conventional HMM, the summation is usually replaced by a maximization, especially during search (Viterbi approximation).

\[ p(w_1^N | x_T^T) = \sum_{t_1^N} p(w_1^N, t_1^N | x_T^T) \]

(2)

\[ = \sum_{t_1^n} \prod_{n=1}^N p(w_n, t_n | w_1^{n-1}, t_1^{n-1}, x_T^T) \]  

\[ = \sum_{t_1^n} \prod_{n=1}^N p(w_n | w_1^{n-1}, t_1^{n-1}, x_T^T) \cdot p(t_n | w_1^{n-1}, t_1^{n-1}, x_T^T) \]  

(3)  

transition model  

\[ = \prod_{n=1}^N \sum_{t_n=1}^{T'} \sum_{t_1^n} p(t_n | w_1^{n-1}, t_1^{n-1}, x_T^T) \cdot p(w_n | w_1^{n-1}, t_n, x_T^T) \]

(4)  

emission model  

\[ \approx \prod_{n=1}^N \sum_{t_n \in \text{topK}} p(t_n | w_1^{n-1}, x_T^T) \cdot p(w_n | w_1^{n-1}, t_n, x_T^T) \]

(5)  

polynomial complexity, \( N \times K \)

Assuming the zero-order Markov assumption to simplify the dependencies of the alignment sequences, we decompose the posterior into two parts: transition and emission, obtaining Eq. 3. Performing the sum over all alignments would lead to a combinatorial problem. Due to the zero-order dependence, one can swap the sum and product in Eq. 4 by the distributive property that leads to polynomial complexity. Here, we note the difference between Eq. 4 and Eq. 1 of the attention model.

4. NEURAL PARAMETERIZATION

To parameterize the individual components of the latent model, we use both the LSTM attention [9] and the Transformer model [24]. Both setups are mainly based on [25].

4.1. LSTM Modeling

We use a stacked bidirectional LSTM (BLSTM) equipped with max-pooling in time to compute a sequence of encoder states \( h_1^{T'} \). In the attention baseline, at each step, the decoder LSTM generates an output label conditioned on the previous label, the last decoder state \( s_{n-1} \) and the context vector \( c_n \).

\[ p(w_n | w_1^{n-1}, x_T^T) = \text{softmax}(w_{n-1}, s_{n-1}, c_n) \]

(6)

In order to build our latent attention model, we need to parameterize the transition and emission models.

**Transition model** To form the transition model, we use the attention weights itself, \( \alpha_{n,t} \), as our distribution.

\[ p(t_n | w_1^{n-1}, x_T^T) = \text{Att}(h_{t_n}, s_{n-1}) \]

(7)

**Emission model** Here, we combine the decoder state with the encoder states, as well as the last output token at each time step. This can also be seen as a hard attention model.

\[ p(w_n | w_1^{n-1}, t_n, x_T^T) = \text{softmax}(w_{n-1}, s_{n-1}, h_{t_n}) \]

(8)

Once we compute the transition and emission probabilities, we sum over \( t_n \) positions as stated in Eq. 5 with topK approximation. We again note the difference between equations 6 and 7 and direct dependency on temporal positions instead of a summary of all inputs as a single vector, \( c_n \).

4.2. Transformer Modeling

We also parameterize our latent attention model using the Transformer architecture, where both the encoder and decoder are composed of stacked self-attentive layers. The decoder contains an extra multihead attention layer, incorporating the encoder and the decoder states to get the attention weights. Several attention heads are used to attend to the input positions. At decoder layer \( l_d \), each attention head \( m \), computes a normalized distribution over the input positions, \( \alpha_{m,n}^{(l_d)}(t_n | n) \). These weights are used to compute the context vectors. The concatenation of these context vectors is fed through a feed-forward layer to form the decoder states \( s_{n-1}^{(l_d)} \). The representation from the last decoder layer is then fed into a softmax layer providing the final distribution over the vocabulary. Following the Transformer architecture, we build our latent attention model as:
Table 1: LibriSpeech 1000h results. AM: acoustic model that can be joint model as the attention. LM: language model.

<table>
<thead>
<tr>
<th>Method</th>
<th>AM</th>
<th>LM</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
<td>clean</td>
</tr>
<tr>
<td>HMM</td>
<td>LSTM</td>
<td>Trans.</td>
<td>1.9</td>
</tr>
<tr>
<td>attention</td>
<td>LSTM</td>
<td>none</td>
<td>3.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>LSTM</td>
<td>Trans.</td>
<td>2.5</td>
</tr>
<tr>
<td>Trans.</td>
<td>LSTM</td>
<td>Trans.</td>
<td>2.2</td>
</tr>
<tr>
<td>Trans.</td>
<td>Trans.</td>
<td>Trans.</td>
<td>2.5</td>
</tr>
</tbody>
</table>

| This work       |       |       |        |        |       |        |
| attention       | LSTM  | none   | 4.1    | 11.2   | 4.2   | 11.8   |
| latent attention| LSTM  |       | 5.3    | 12.8   | 5.3   | 13.7   |
| attention       | Trans. |       | 4.3    | 11.0   | 4.5   | 11.5   |
| latent attention| Trans. |       | 4.6    | 11.9   | 4.6   | 12.1   |

Transition model To form the transition model in Eq. 4, we average over the attention heads in the last layer of the decoder at each time step. We use the last layer as it showed better performance on preliminary experiments, in comparison to averaging over all layers.

\[ p(t_n|w_n^{t-1}, x_T^T) = \frac{1}{M} \sum_{m=1}^{M} \alpha_m^{L_d}(t_n|n) \]

\[ \alpha_m^{L_d}(t_n|n) = \text{DotAtt}(h_{t_n}, s_{L_d-1}^n) \]

where DotAtt is the dot-product attention. M is the number of attention heads and \( L_d \) is the number of decoder layers.

Emission model Here, we combine the encoder states only with the last decoder state of the top layer (not all layers), as well as the last output word at each time step. Therefore,

\[ p(w_n|w_n^{t-1}, t_n, x_T^T) = \text{softmax}(w_n^{t-1}, s_{L_d-1}^n, h_{t_n}) \]

The latent attention model has an identical number of parameters as its base architecture, for both the LSTM and the Transformer, however, in theory, it has more statistical capacity than the attention-based models. The attention model is a deterministic interpolation of deterministic features (encoder states), whereas the latent attention model is a mixture model.

With the attention model, the network outputs exactly one distribution over the output vocabulary per time step, whilst our model outputs \( T \) (or \( K \) when using the optimization) distributions, which are mixed probabilistically, yielding a multimodal marginal distribution. We equip a deterministic soft function with a probabilistic model, in which alignments have direct effects on the final label sequence posterior and it worths exploring whether a lexicalized alignment model can assign more appropriate scores.

5. EXPERIMENTS

We have carried out our experiments on 1000h LibriSpeech, 300h of Switchboard telephone and TED-LIUM release 2 (200h) for all tasks, we use byte-pair-encoding with a size of 10k, 1k and 1k respectively. Both our LSTM and Transformer baselines are mainly taken from [25].

For the LSTM-based setups, we apply a 6-layer of BLSTM encoder of 1024 nodes and a 1-layer unidirectional LSTM decoder with a size of 1024 equipped with a single-head additive attention. For the Transformer-based setups, both the encoder and decoder of the Transformer models consist of 12 layers, with the internal hidden size of 512 and 8 heads. The feed-forward hidden dimension is chosen to be 2048 for the LibriSpeech and 1024 for the Switchboard and TED-LIUM. We utilize 2 stacked BLSTM with interleaved max-pooling with a time reduction factor of 6.

All our models (LSTM, Transformer, and latent attention) use a variant of SpecAugment as our data augmentation as well as a variant of layerwise pre-training strategy similar to [34]. The CTC as an additional loss is used on top of the encoder during training. The models are trained end-to-end using the Adam optimizer with warm-up technique, and a dropout of 0.1% in all tasks, all of our end-to-end models perform worse compared to the hybrid HMM. A comparison to the recent other works can be found in the tables as well. As listed in Table 1, our Transformer performs as good as our LSTM baseline, while in the other tasks it is behind due to observed overfitting. The Transformer latent attention model is worse than the Transformer baseline by 0.3% and 0.9% WER on average on the clean and other sets respectively. The difference is larger on the LSTM-based models.

For Switchboard as shown in Table 2, the LSTM latent attention model slightly underperforms the LSTM baseline, by 0.4% absolute WER on average, and the Transformer latent model is behind its analogous baseline. We also compare our latent model with the segmental model that is conceptually close to our model. Both LSTM and Transformer-based

1https://github.com/rwth-ifs/returnn-experiments
latent models outperform the segmental approach. We have also done our experiments on the TED-LIUM-v2 dataset including longer sequences, which can be more challenging. The results can be seen in Table 3. The similar behavior to the other tasks has been observed. The latent models both based on LSTM and Transformer parametrizations do not yet reach the performance of our baselines, therefore more work has to be done to examine possible improvements. We have observed that the latent models have a better frame error rate (FER) indicating that they provide better alignments and contribute more on the final label sequence posterior, hence make a better decision of the output label.

5.2. Attention Analysis
To support our previous justification, we qualitatively examine the alignment behavior and compare the attention distribution \( p(\tau_n|w_1^{n-1}, x_T^T) \) of the models. Fig. 1 shows a forced alignment example on one utterance from the LibriSpeech dev-other. As seen, the attention layer used in the latent attention model has highly concentrated monotonic alignments compared to the attention models. We have often observed this characteristic over all sets for many utterances independent of sequence length. We believe the difference in the distributions might be attributed to the way that the alignment is explicitly used in the latent attention model, and thus predicts the output label differently. For the Transformer-based models many attention heads did not perform any different operations and always attend to the same input position throughout an entire sequence. Here, we have taken the best attention heads of all Transformer-based models.

5.3. Effect of K
We also plot the effect of K applied in the topK approximation versus the corresponding WER on dev-other set using the Transformer latent attention model in Fig. 2. As it is shown, the WER goes down from 12.1% to 11.9% by changing K from 2 to 6 respectively, and then it saturates without any further improvements. This implies that small values of K are good enough for training. Given a trained model using \( K = 6 \), we also check if larger values of K improves the performance during inference and confirm that applying the same K value in both training and inference meets the model’s requirements. Using \( K = 1 \) leads to faster decoding with WER of 12.3%.

Table 3: TED-LIUM-v2 results. 1: single Tesla P100 GPU.

<table>
<thead>
<tr>
<th>Method</th>
<th>AM</th>
<th>LM</th>
<th>WER [%]</th>
<th>Train speed [char/sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>HMM attention</td>
<td>LSTM</td>
<td>RNN</td>
<td>7.1</td>
<td>7.7</td>
</tr>
<tr>
<td>LSTM</td>
<td>none</td>
<td>14.6</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>Trans.</td>
<td>none</td>
<td>15.3</td>
<td>16.7</td>
<td></td>
</tr>
<tr>
<td>Trans.</td>
<td>LSTM</td>
<td>9.3</td>
<td>8.1</td>
<td></td>
</tr>
<tr>
<td>Trans.</td>
<td>Trans.</td>
<td>10.3</td>
<td>8.8</td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>attention</td>
<td>LSTM</td>
<td>none</td>
<td>13.4</td>
<td>10.5</td>
</tr>
<tr>
<td>latent attention</td>
<td>LSTM</td>
<td></td>
<td>13.8</td>
<td>11.8</td>
</tr>
<tr>
<td>attention</td>
<td>Trans.</td>
<td></td>
<td>14.7</td>
<td>12.5</td>
</tr>
<tr>
<td>latent attention</td>
<td>Trans.</td>
<td></td>
<td>15.3</td>
<td>13.5</td>
</tr>
</tbody>
</table>

Fig. 1: An attention example on LibriSpeech dev-other. The best of all models. 1:average over all heads of 4th (best) layer. 2:average over all heads of 12th (best) layer.

Fig. 2: WER vs. different values of K during training.

5.4. Complexity
To compute the emission probability, we employ a softmax over the vocabulary \( V \), \( T \) times (or \( K \) times using the approximation). In theory, the time complexity of the latent model is \( O(N \times K \times V) \) compared to \( O(N \times V) \) for the attention models. Thus, the latent model can become very slow for a large vocabulary, which is not the case for most of the end-to-end NN-based ASR systems in which the label sequences consist of subwords or characters. In practice, using GPU and optimized matrix operations, the computational cost is significantly amortized by parallelization (see Table 3).

6. CONCLUSION
We have presented new results for a zero-order latent attention based direct HMM model using the LSTM-based and Transformer attention models. The method keeps the explicit alignment as part of the stochastic model and combines a straightforward training procedure as well as an efficient and simple beam search. It has not yet reached parity on all tasks, however, it provides more focused alignments. These are the first experiments using the latent model and we expect better results with more tuning. More work needs to be done and we believe the investigation of the attention weights and further exploration on higher-order dependence w.r.t the alignments are key aspects that can guide future development.

7. ACKNOWLEDGEMENTS
This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 694537, project “SEQCLAS”), and from a Google Focused Award. Experiments were partially performed with computing resources granted by RWTH Aachen University under project nova0003. The work reflects only the authors’ views and none of the funding parties is responsible for any use that may be made of the information it contains.