

Neural Hidden Markov Model for Machine Translation

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department

RWTH Aachen University, 52056 Aachen, Germany

<surname>@i6.informatik.rwth-aachen.de

Abstract

This work aims to investigate alternative neural machine translation (NMT) approaches and thus proposes a neural hidden Markov model (HMM) consisting of neural network-based alignment and lexicon models. The neural models make use of encoder and decoder components, but drop the attention component. The training is end-to-end and the standalone decoder is able to provide comparable performance with the state-of-the-art attention-based models on three different translation tasks.

1 Introduction

Attention-based neural translation models (Bahdanau et al., 2015; Luong et al., 2015) attend to specific positions on the source side to generate translation. Using the attention component provides significant improvements over the pure encoder-decoder sequence-to-sequence approach (Sutskever et al., 2014) that uses no such attention mechanism. In this work, we aim to compare the performance of attention-based models to another baseline, namely, neural hidden Markov models.

The neural HMM has been successfully applied in the literature on top of conventional phrase-based systems (Wang et al., 2017). In this work, our purpose is to explore its application in standalone decoding, i.e. the model is used to generate and score candidates without assistance from a phrase-based system. Because translation is done standalone using only neural models, we still refer to this as NMT. In addition, while Wang et al. (2017) applied feedforward networks to model alignment and translation, the recurrent structures proposed in this work surpass the feedforward variants by up to 1.3% in BLEU.

By comparing neural HMM and attention-based NMT, we shed light on the role of the attention component. To this end, we use an alignment-based model that has a recurrent bidirectional encoder and a recurrent decoder, but use no attention component. We replace the attention mechanism by a first-order HMM alignment model. Attention levels are deterministic normalized similarity scores part of the architecture design of an otherwise fully supervised classifier. HMM-style alignments on the other hand are discrete random variables and (unlike attention levels) must be marginalized. Once alignments are marginalized, which is tractable for a first-order HMM, parameters can be estimated to attain a local optimum of log-likelihood of observations as usual.

2 Motivation

In attention-based approaches, the alignment distribution is used to select the positions in the source sentence that the decoder attends to during translation. Thus the alignment model can be considered as an implicit part of the translation model. On the other hand, separating the alignment model from the lexicon model has its own advantages: First of all, this leads to more flexibility in modeling and training: The models can not only be trained separately, but they can also have different model types, such as neural models, count-based models, etc. Second, the separation avoids propagating errors from one model to another. In attention-based systems, the translation score is based on the alignment distribution, in which errors can be propagated from the alignment part to the translation part. Third, probabilistic treatment to alignments in NMT typically implies an extended degree of interpretability (e.g. one can inspect posteriors) and control over the model (e.g. one can impose priors over alignments

and lexical distributions).

3 Neural Hidden Markov Model

Given a source sentence $f_1^J = f_1 \dots f_j \dots f_J$ and a target sentence $e_1^I = e_1 \dots e_i \dots e_I$, where $j = b_i$ is the source position aligned to the target position i , we model translation using an alignment model and a lexicon model:

$$p(e_1^I | f_1^J) = \sum_{b_1^I} p(e_1^I, b_1^I | f_1^J) \quad (1)$$

$$:= \sum_{b_1^I} \prod_{i=1}^I \underbrace{p(e_i | b_i^I, e_0^{i-1}, f_1^J)}_{\text{lexicon model}} \cdot \underbrace{p(b_i | b_1^{i-1}, e_0^{i-1}, f_1^J)}_{\text{alignment model}} \quad (2)$$

Instead of predicting the absolute source position b_i , we use an alignment model $p(\Delta_i | b_1^{i-1}, e_0^{i-1}, f_1^J)$ that predicts the jump $\Delta_i = b_i - b_{i-1}$.

Wang et al. (2017) applied feedforward neural networks for modeling the lexicon and alignment probabilities. In this work, we would like to model these distributions using recurrent neural networks (RNN). RNNs have been shown to outperform feedforward variants in language and translation modeling. This is mainly due to that RNN can handle arbitrary input lengths and thus include unbounded context information. Unfortunately, the recurrent hidden layer cannot be easily applied for the neural hidden Markov model, since it will significantly complicate the computation of forward-backward messages when running Baum-Welch. Nevertheless, we can apply long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) structure for source and target words embedding. With this technique we can take the essence of LSTM RNN and do not break any sequential generative model assumptions.

Our models are close in structure to the model proposed in Luong et al. (2015), where we have a component that encodes the source sentence, and another that encodes the target sentence. As shown in Figure 1, we use a source side bidirectional LSTM embedding $h_j = \vec{h}_j + \overleftarrow{h}_j$, where $\vec{h}_j = \text{LSTM}(W, f_j, \vec{h}_{j-1})$ and $\overleftarrow{h}_j = \text{LSTM}(V, f_j, \overleftarrow{h}_{j+1})$, as well as a target side LSTM embedding $s_{i-1} = \text{LSTM}(U, e_{i-1}, s_{i-2})$. h_j , \vec{h}_j , \overleftarrow{h}_j and s_{i-1} , s_{i-2} are vectors, W , V and U are weight matrices. Before the non-linear hidden layers, there is a projection layer which

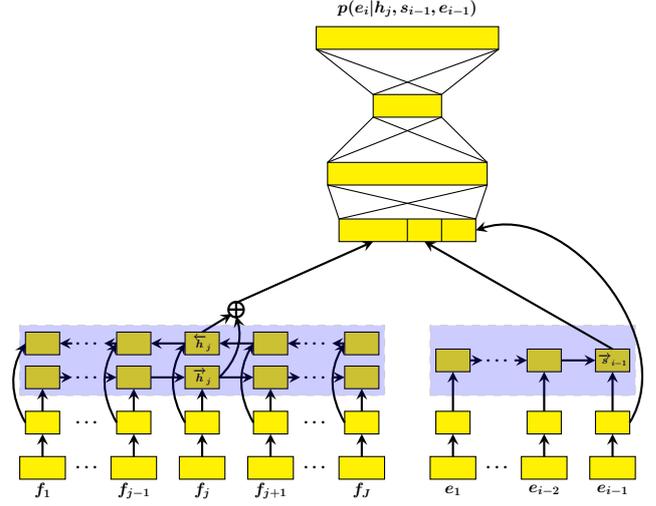


Figure 1: The architecture of our neural networks with LSTM RNN on source and target side.

concatenates h_j , s_{i-1} and e_{i-1} . Then the neural network-based lexicon model is given by

$$p(e_i | b_1^i, e_0^{i-1}, f_1^J) := p(e_i | h_j, s_{i-1}, e_{i-1}) \quad (3)$$

and the neural network-based alignment model

$$p(b_i | b_1^{i-1}, e_0^{i-1}, f_1^J) := p(\Delta_i | h_{j'}, s_{i-1}, e_{i-1}) \quad (4)$$

where $j' = b_{i-1}$.

The training criterion is the logarithm of sentence posterior probabilities over training sentence pairs (F_r, E_r) , $r = 1, \dots, R$:

$$\arg \max_{\theta} \left\{ \sum_r \log p_{\theta}(E_r | F_r) \right\} \quad (5)$$

The derivative for a single sentence pair $(F, E) = (f_1^J, e_1^I)$ is:

$$\begin{aligned} & \frac{\partial}{\partial \theta} \log p_{\theta}(E | F) \\ &= \sum_{j', j} \sum_i p_i(j', j | f_1^J, e_1^I; \theta) \\ & \quad \cdot \frac{\partial}{\partial \theta} \log p(j, e_i | j', e_0^{i-1}, f_1^J; \theta) \end{aligned} \quad (6)$$

with HMM posterior weights $p_i(j', j | f_1^J, e_1^I; \theta)$, which can be computed using the forward-backward algorithm.

The entire training procedure can be summarized as *backpropagation in an EM framework*:

1. compute:
 - the posterior HMM weights
 - the local gradients (backpropagation)
2. update neural network weights

4 Decoding

In the decoding stage we still calculate the sum over alignments and apply a target-synchronous beam search for the target string.

The auxiliary quantity for each unknown partial string e_0^i is specified as $Q(i, j; e_0^i)$. During search, the partial hypothesis is extended from e_0^{i-1} to e_0^i :

$$Q(i, j; e_0^i) = \sum_{j'} [p(j, e_i | j', e_0^{i-1}, f_1^J) \cdot Q(i-1, j'; e_0^{i-1})] \quad (7)$$

The decoder is shown in Algorithm 1. In the innermost loop (line 11-13), alignments are hypothesized and used to calculate the auxiliary quantity $Q(i, j; e_0^i)$. Then for each source position j , the lexical distribution over the full target vocabulary is computed (line 14). The distributions are accumulated ($Q(i; e_0^i) = \sum_j Q(i, j; e_0^i)$, line 16), then sorted (line 18) and the best candidate translations ($\arg \max_{e_i} Q(i; e_0^i)$) lying within the beam are used to expand the partial hypotheses (line 19-23). `cache` is a two-dimensional list of size $J \times |V_{\text{src}}|$ (source vocabulary size), which is used to cache the current quantities.

Whenever a partial hypothesis in the beam ends with the sentence end symbol (`<EOF>`), the counter will be increased by 1 (line 26-28). The translation is terminated if the counter reaches the beam size or hypothesis sentence length reaches three times the source sentence length (line 6). If a hypothesis stops but its score is worse than other hypotheses, it is eliminated from the beam, but it still contests non-terminated hypotheses. During comparison the scores are normalized by hypothesis sentence length. Note that we have no explicit coverage constraints. This means that a source position can be revisited many times, whereby creating one-to-many alignment cases. This also allows unaligned source words.

In the neural HMM decoder, word alignments are estimated and scored according to the distribution calculated by the neural network alignment model, leading alignment decisions to become part of the beam search. The search space consists of both alignment and translation decisions. In contrast, the search space in attention-based decoding consists only of translation decisions.

The decoding complexity is $\mathcal{O}(J^2 \cdot I)$ (J = source sentence length, I = target sentence length)

Algorithm 1 Neural HMM Decoder

```

1: function TRANSLATE( $f_1^J$ , beam_size)
2:   count = 0
3:    $i = 1$ 
4:   hyps =  $\{e_0\}$ 
5:   new_hyps =  $\{\}$ 
6:   while count < beam_size and  $i < 3 \cdot J$  do
7:     for hyp in hyps do
8:       sum_dist =  $[0] * |V|_{\text{src}}$ 
9:       for  $j$  from 1 to  $J$  do
10:        sum = 0
11:        for  $j'$  from 1 to  $J$  do
12:          sum = sum + SCORES(hyp,  $j'$ )
               $\cdot p_{\text{align}}(f_{j'}, j - j')$ 
13:        end for
14:        cache[ $j$ ] = sum  $\cdot \text{lex\_dist}(f_j)$ 
15:        #Element wise addition
16:        sum_dist = sum_dist  $\oplus$  cache[ $j$ ]
17:      end for
18:      dist = SORT(sum_dist, beam_size)
19:      for word in dist[:beam_size] do
20:        new_hyp = EXTEND(hyp, word)
21:        SETSCORES(new_hyp, cache)
22:        new_hyps.INSERT(new_hyp)
23:      end for
24:    end for
25:    PRUNE(new_hyps, beam_size)
26:    for <EOF> in new_hyps do
27:      count = count + 1
28:    end for
29:    hyps = new_hyps
30:     $i = i + 1$ 
31:  end while
32:  return GETBEST(hyps)
33: end function

```

compared to $\mathcal{O}(J \cdot I)$ for attention-based models. These are theoretical complexities of decoding on a CPU only considering source and target sentence lengths. In practice, the size of the neural network must also be taken into account, and there are some optimized matrix multiplications for decoding on a GPU. In general, the decoding speed of our model is about 3 times slower than that of a standard attention model (1.07 sentences per second vs. 3.00 sentences per second) on a single GPU. This is still an initial decoder and we did not spend much time on accelerating its decoding yet. The optimization of our decoder would be a promising future work.

5 Experiments

The experiments are conducted on the WMT 2017 German↔English and Chinese→English translation tasks, which consist of 5M and 23M parallel sentence pairs respectively. Translation quality is measured with the case sensitive BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) metric on `newstests 2017`, which contain 3004 (German↔English) and 2001 (Chinese→English) sentence pairs.

For German and English preprocessing, we use the *Moses* tokenizer with hyphen splitting, and perform truecasing with *Moses* scripts (Koehn et al., 2007). For German↔English subword segmentation (Sennrich et al., 2016), we use 20K joint BPE operations. For the Chinese data, we segment it using the *Jieba*¹ segmenter. We then learn a BPE model on the segmented Chinese, also using 20K merge operations. During training, sentences with a length greater than 50 subwords are filtered out.

5.1 Attention-Based System

The attention-based systems are trained with *Sockeye* (Hieber et al., 2017), which implement an attentional encoder-decoder with small modifications to the model in Bahdanau et al. (2015). The encoder and decoder word embeddings are of size 620. The encoder consists of a bidirectional layer with 1000 LSTMs with peephole connections to encode the source side. We use *Adam* (Kingma and Ba, 2015) as optimizer with a learning rate of 0.001, and a batch size of 50. The network is trained with 30% dropout for up to 500K iterations and evaluated every 10K iterations on the development set with BLEU. Decoding is done using beam search with a beam size of 12. In the end the four best models are averaged as described in

¹<https://github.com/fxsjy/jieba>

WMT 2017	# free parameters	German→English		English→German		Chinese→English	
		BLEU[%]	TER[%]	BLEU[%]	TER[%]	BLEU[%]	TER[%]
FFNN-based neural HMM ¹	33M	28.3	51.4	23.4	58.8	19.3	64.8
LSTM-based neural HMM ²	52M	29.6	50.5	24.6	57.0	20.2	63.7
Attention-based neural network ³	77M	29.5	50.8	24.7	57.4	20.2	63.8

Table 1: Experimental results on WMT 2017 German↔English and Chinese→English test sets. All models are trained without synthetic data. Single model is used for decoding.

¹ (Wang et al., 2017) but applied in decoding instead of rescoring

² This work

³ (Bahdanau et al., 2015) with small modifications (Section 5.1)

the beginning of Junczys-Dowmunt et al. (2016).

5.2 Neural Hidden Markov Model

The entire neural hidden Markov model is implemented in *TensorFlow* (Abadi et al., 2016). The feedforward models have three hidden layers of sizes 1000, 1000 and 500 respectively, with a 5-word source window and a 3-gram target history. 200 nodes are used for word embeddings.

The output layer of the neural lexicon model consists of around 25K nodes for all subword units, while the neural alignment model has a small output layer with 201 nodes, which reflects that the aligned position can jump within the scope from -100 to 100.

Apart from the basic projection layer, we also applied LSTM layers for the source and target words embedding. The embedding layers have 350 nodes and the size of the projection layer is 800 (400 + 200 + 200, Figure 1). We use *Adam* as optimizer with a learning rate of 0.001. Neural lexicon and alignment models are trained with 30% dropout and the norm of the gradient is clipped with a threshold 1 (Pascanu et al., 2014). In decoding we use a beam size of 12 and the element-wise average of all weights of the four best models also results in better performance.

5.3 Results

We compare the neural HMM approach (Subsection 5.2) with the state-of-the-art attention-based approach (Subsection 5.1) on different translation tasks. The results are presented in Table 1. Compare to the model presented in Wang et al. (2017), switching to LSTM models has a clear advantage, which improves the FFNN-based system by up to 1.3% BLEU and 1.8% TER. It seems that the HMM model benefits from richer features, such as LSTM states, which are very similar to what an attention mechanism would require. We actually

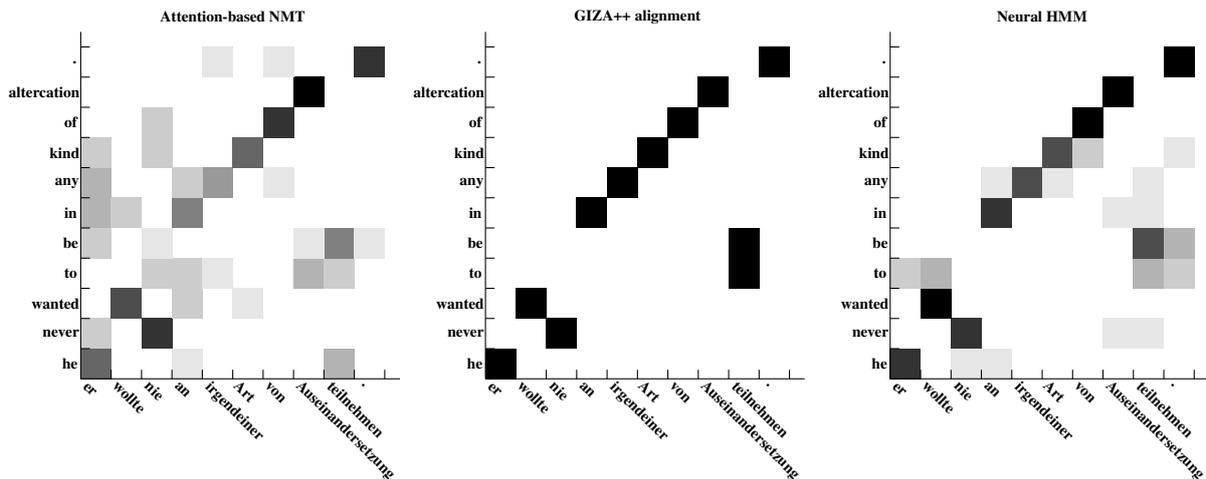


Figure 2: Attention weight and alignment matrices visualized in heat map form. Generated by the attention NMT baseline, GIZA++ and the neural hidden Markov model.

expected it to do with less, the reason being that alignment distributions get refined a posteriori and so they do not have to be as strong a priori. We can also observe that the performance of our approach is comparable with the state-of-the-art attention-based system with $25M$ more parameters on all three tasks.

5.4 Alignment Analysis

We show an example from the German→English `newstest 2017` in Figure 2, along with the attention and alignment matrices. We can observe that the neural network-based HMM could generate a more clear alignment path compared to the attention weights. In this example, it can exactly estimate the alignment positions for words `wanted` and `of`.

6 Discussion

We described a novel formulation for a neural network-based machine translation system, which applied neural networks to the conventional hidden Markov model. The training is end-to-end, the model is monolithic and can be used as a standalone decoder. This results in a more modern and efficient way to use HMM in machine translation and enables neural networks to benefit from HMMs.

Experiments show that replacing attention with alignment does not improve the translation performance of NMT significantly. One possible reason is that alignment may fail to capture relevant contexts as attention does. While alignment aims to identify translation equivalents between two lan-

guages, attention is designed to find relevant context for predicting the next target word. Source words with high attention weights are not necessarily translation equivalents of the target word. Although using alignment does not lead to significant improvements in terms of BLEU over attention, we think alignment-based NMT models are still useful for automatic post editing and developing coverage-based models. These might be interesting future directions to explore.

Acknowledgments



This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement n° 645452 (QT21), and from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme, grant agreement n° 694537 (SEQCLAS). The work reflects only the authors’ views and neither the European Research Council Executive Agency nor the European Commission is responsible for any use that may be made of the information it contains. The GPU cluster used for the experiments was partially funded by Deutsche Forschungsgemeinschaft (DFG) Grant INST 222/1168-1. Tamer Alkhouli was partly funded by the 2016 Google PhD Fellowship for North America, Europe and the Middle East.

References

- Martín Abadi, Ashish Agarwal, and Paul Barham et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR* abs/1603.04467.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, CA, USA.
- Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. [Sockeye: A Toolkit for Neural Machine Translation](https://arxiv.org/abs/1712.05690). *ArXiv e-prints* <https://arxiv.org/abs/1712.05690>.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.
- Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT. In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*. Berlin, Germany, pages 319–325.
- Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*. San Diego, CA, USA.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic, pages 177–180.
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1412–1421.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, PA, USA, pages 311–318.
- Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to Construct Deep Recurrent Neural Networks. In *Proceedings of the Second International Conference on Learning Representations*. Banff, Canada.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 1715–1725.
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas*. Cambridge, MA, USA, pages 223–231.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems 27*. Montréal, Canada, pages 3104–3112.
- Weiyue Wang, Tamer Alkhouli, Derui Zhu, and Hermann Ney. 2017. Hybrid Neural Network Alignment and Lexicon Model in Direct HMM for Statistical Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, pages 125–131.