You May Not Need Attention

Ofir Press[♠] **Noah A. Smith**[♠♣]
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]Allen Institute for Artificial Intelligence
{ofirp,nasmith}@cs.washington.edu

Abstract

In NMT, how far can we get without attention and without separate encoding and decoding? To answer that question, we introduce a recurrent neural translation model that does not use attention and does not have a separate encoder and decoder. Our **eager translation model** is low-latency, writing target tokens as soon as it reads the first source token, and uses constant memory during decoding. It performs on par with the standard attention-based model of Bahdanau et al. (2014), and better on long sentences.[1]

1 Introduction

Nearly all actively-researched NMT models have the following properties:

- The decoder uses an attention mechanism over the source sequence representations. (Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017).
- The encoder and decoder are two different modules, and the encoder must finish encoding the source sentence before the decoder starts operating (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015; Kalchbrenner et al., 2016; Vaswani et al., 2017).

Here we investigate how well an NMT model can do without these properties.

To that end, we start with the model of Bahdanau et al. (2014), remove the attention mechanism, and unify the encoder and decoder into a single, *simpler* model that resembles the language model of Zaremba et al. (2014). The result is that our model can "eagerly" begin emitting a translation as soon as it reads the first source word, and can finish translating soon after the last source word is read.

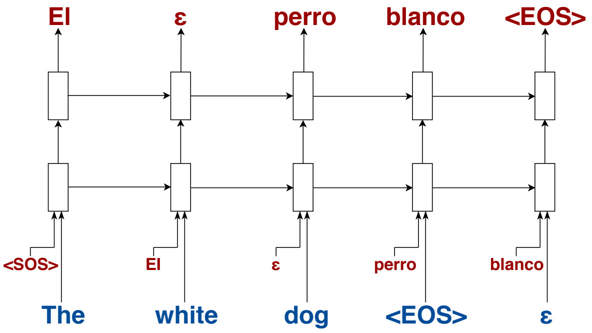

Figure 1: The eager model translating the sentence "The white dog" into Spanish. Source (target) tokens are in blue (red). ε is the padding token, which is removed during postprocessing. The diagram presents an eager translation model with two LSTM layers.

The **eager translation model** uses a constant amount of memory, since it needs to use only one previous hidden state (rather than all previous hidden states) at every timestep. Instead of "cramming a whole sentence into a single vector", our approach crams a prefix of a source sentence (and its resulting translation) into a dynamic memory vector, emitting target tokens immediately and every time another source word is read.

In practice, most of the changes required by our eager translation model affect preprocessing (§2). Experimentally, we show that our model performs on par with the attention-based machine translation model of (Bahdanau et al., 2014). We find that our model outperforms the attention-based model on longer sequences (a known challenge for attention-based models) but is less effective on short sequences (§4).

We expect that, in the future, this kind of low-latency and low-memory translation may be attractive in some application settings.

[1] Our code is available at https://github.com/ofirpress/YouMayNotNeedAttention

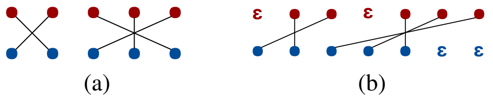

Figure 2: A source (blue) and target (red) sequence with their alignment before (a) and after (b) preprocessing to make the pair "eager feasible."

	FR→EN	EN→FR	DE→EN	EN→DE
ε Proportion	23%	14%	23%	20%

Table 1: Average percentage of ε in the target sentences of the four language directions. Initial padding tokens and padding tokens inserted to make the source and target sequence lengths equal are not counted.

2 Data Preprocessing

The eager translation model requires the training data to be preprocessed in a specific way described below.

We begin by inferring a correspondence between words in a source/target sentence pair $(\boldsymbol{s}, \boldsymbol{t})$ assuming each target word is aligned to (at most) one source word, as in Brown et al. (1993). We describe a source and target sequence as *eager feasible* if, for every aligned pair of words (s_i, t_j), $i \leq j$.

Our model requires each source and target sentence in the training set to have this property. We achieve this by first using the alignments inferred by an off-the-shelf alignment model (fast_align; Dyer et al., 2013), then inserting the minimal number of ε (empty) tokens into the target sentence to achieve the desired property. These ε tokens are used during training and inference, but are removed in a postprocessing step when generating translations.

For example, for the source sentence "El perro blanco" (literally: "The dog white") the correct translation is "The white dog". Assuming the obvious alignment in which the second English word is aligned to the third Spanish word, to make the sequences eager feasible, we modify the target sentence so that it becomes "The ε white dog". A more complex example is given in Fig. 2.

We define the algorithm for making a source and target sequence eager feasible as follows. $\boldsymbol{s} = \langle s_1, \ldots, s_m \rangle$ is the source sentence and $\boldsymbol{t} = \langle t_1, \ldots, t_n \rangle$ is the target sentence. Let $\mathcal{A}$ be a set of ordered pairs (i, j) such that t_j is aligned to s_i.

We go over the target words one by one, from left to right. Suppose the current target word is t, and it currently occupies position j in the target sequence. If t is aligned to a source word s_i such that $i \leq j$, then we move on to the next target word. Otherwise, we insert enough ε tokens in the target sequence, just before t, so that it shifts to position i in the target sequence. This, of course, shifts the target words to its right, as well.

Table 1 shows the average percentage of these εs in the target sequence sets of the four training directions that we trained on.

In order to make the eager model's task simpler, we also experiment with adding $b \in \{0, 1, \ldots, 5\}$ padding ε tokens at the beginning of every target sentence. This gives the model a chance to consume more of the source sentence before it begins translating; at inference time, we force the model to produce b εs before the translation. Our preprocessing algorithm takes these initial padding εs into account (if they are used), with the result that fewer εs need to be inserted *between* target tokens.

After the above transformations, if a training pair of sentences do not have the same length, we insert εs at the end of the shorter sentence to make the lengths equivalent.

3 Model

At each timestep, our model first embeds the current input word (in the source language) and the previously selected output word (in the target language) into dense representations both of dimension E. These vectors are concatenated and the resulting vector is then fed into a multi-layered LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) containing $2E$ units at each layer. The output of the LSTM is transformed into a vector of size E using a fully connected layer. The output of that fully connected layer is transformed into a distribution over the target vocabulary using an output embedding matrix and the softmax function. We tie the input embedding matrix of the source language, the input embedding matrix of the target language, and the output embedding matrix (Press and Wolf, 2017; Inan et al., 2016).

Our model strongly resembles the recurrent language model of Zaremba et al. (2014). As in that model, during training we use teacher forcing (Williams and Zipser, 1989) and the cross-entropy loss. Note that we treat the padding symbol as a token in the target language, and we do not use a special loss or make any modifications to the objective for timesteps in which the target output is the padding token.

Additionally, unlike most other translation models, our model uses a constant amount of memory during inference. It only has to store the previous hidden state in memory, and does not have to store the representations of all previously encoded words. Finally, the decoding complexity of our model is at worse $\mathcal{O}(n+m)$.

3.1 Aligned Batching

After preprocessing, all source-target pairs will have equal lengths. We concatenate all the source sentences into a source string and all the target sentences into a target string, keeping them in the same order. This allows us to train our translation model similarly to how a language model is trained. Specifically, a backpropagation through time (BPTT) hyperparameter is defined. Each element in every batch contains BPTT source tokens and their respective target tokens. The next batch uses the next BPTT tokens, and so on. As in language modeling training, the last hidden state from the $(i-1)$th batch becomes the initial hidden state of the ith batch.

3.2 Decoding

During inference, we use a modified beam search to improve the quality of the outputs. We modify the beam search algorithm as follows:

- Padding limit: We place an upper limit on the number of padding symbols emitted, by forcing the probability of ε to zero after the limit is reached. Initial padding symbols are not counted towards this limit.

- Source padding injection (SPI): During the development of the model we noticed that the decoder assigns a high probability to the end-of-sequence (EOS) token once the EOS token in the source language is read. We found that translation quality improves if we insert ε tokens on the *input* side just before the EOS. If the SPI hyperparameter is set to c, our beam search process will consider anywhere from 0 to c padding ε tokens before the source EOS.

  Source padding injection enables the model to output a sentence that is longer than the input sentence. Without it, the generated sentence length is capped by the source sentence length (and is exactly equal to source sentence length minus the number of generated padding tokens).

4 Experiments

Setup For both EN↔FR and EN↔DE we train on the WMT 2014[2] dataset, use newstest2013 as the validation dataset, and test on newstest2014. We tokenize the sentences and then segment the words using 32,000 BPE operations (Sennrich et al., 2016). Finally, we shuffle the corpora before training.

We use four LSTM layers with 1,000 units for our model, and embeddings of size 500. The model is regularized during training using dropout on both the LSTM and the word embeddings, as done by Merity et al. (2017). We train our model with a batch size of 200, and we backpropagate through time for 60 tokens. We use SGD and start with a learning rate of 20. We check the perplexity on the validation set every 6,500 updates and halve the learning rate if it does not improve. The padding limit, source padding injection value, and beam size that we use for inference on the test set are the ones that perform best on the development set. These values are reported in Table 4 in the supplementary material section.

As a reference model we use the OpenNMT (Klein et al., 2017) implementation of Bahdanau et al. (2014). We use a model that has two LSTM layers in the encoder and two in the decoder, all with 1,000 units, and embeddings of size 500. This resulted in a model containing a similar number of parameters to our model. The optimization algorithm used is SGD, with a starting rate of 1, which is halved every 10,000 steps if there is no improvement in development set perplexity.

Both our model and the reference model are trained until there is no improvement on the development set for 50,000 updates. The reference model took 13 hours to train on a single GPU while our models took around 38 hours. Although the eager model can process approximately three times the amount of source tokens per second as the OpentNMT reference model, training takes longer because the eager model requires more epochs to converge.

We compute BLEU scores on the detokenized outputs using SacreBLEU (Post, 2018).

[2] http://www.statmt.org/wmt14/translation-task.html

		FR→EN	EN→FR	DE→EN	EN→DE
	0	24.42	19.97	20.11	11.50
	1	25.76	24.81	20.81	15.81
Start	2	27.10	25.63	21.45	16.53
εs	3	27.98	**26.98**	21.39	17.36
	4	28.30	26.37	22.00	17.52
	5	**28.47**	25.49	**22.59**	**17.97**
Ref. Model		28.56	27.20	23.01	18.89

Table 2: BLEU performance on the test sets for the reference model and our model with zero to five initial ε padding tokens (as defined in Sec. 2)

	Source Sentence Length	Number of Sentences	Reference Model BLEU	Eager Model BLEU
FR→EN	1–20	864	**26.22**	23.74
	21–40	1312	**29.50**	29.20
	41–60	659	**28.71**	27.77
	61–80	152	27.66	**27.89**
	81+	16	22.10	**27.44**
DE→EN	1–20	963	**22.94**	20.12
	21–40	1275	**23.07**	22.95
	41–60	414	**23.06**	22.53
	61–80	76	23.02	**23.51**
	81+	9	21.24	**24.73**

Table 3: BLEU performance by source sentence length on the FR→EN and DE→EN test sets.

Results The results of the eager model are reported in Table 2. On FR→EN and EN→FR the model is at most 0.8% lower in terms of BLEU than the reference model. On the harder DE→EN and EN→DE tasks, the eager model is at most 4.8% worse than the reference model.

Table 3 breaks down the performance of the best FR→EN and DE→EN model by length, showing that the eager model is worse on shorter sequence but better on longer ones, which are known to be difficult for attention-based approaches (Koehn and Knowles, 2017). Table 5 in the supplementary material section presents a more detailed breakdown of the performance for different source sequence lengths for all four tasks.

5 Related Work

Early models for neural machine translation consisted of a separate encoder and decoder, but without an attention mechanism (Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). Recent state-of-the-art results have been achieved by either recurrent, attention-based neural translation models (Bahdanau et al., 2014; Wu et al., 2016), transformer-based models (Vaswani et al., 2017), or a combination of these methods (Domhan, 2018; Chen et al., 2018). Kalchbrenner et al. (2016) and Gehring et al. (2017) use convolutional networks in both the encoder and decoder. Gehring et al. (2017) use an attention mechanism, while Kalchbrenner et al. (2016) do not.

Raffel et al. (2017) propose a model that uses a monotonic attention mechanism. Yu et al. (2016) propose a neural transduction model where the alignment is a latent variable. Both of these models have much higher training time requirements than attention-based translation and so only show results on small datasets.

Huang et al. (2017) use a reordering layer in their translation model in order to make the input monotonically aligned to its target output. After the reordering step, decoding is done in parallel. The computational complexity of this model is high.

Wang et al. (2018) present a recurrent translation model that does not employ an attention mechanism. Their neural HMM model has a decoding complexity of $\mathcal{O}(m^2 n)$ in our notation from §2.

All of the aforementioned models use a separate encoder and decoder, and do not begin decoding until the entire source sentence is encoded. In addition, all of the recently introduced models (other than Kalchbrenner et al., 2016; Huang et al., 2017; Wang et al., 2018) use an attention mechanism, either just in the decoder or in both decoder and encoder.

Elbayad et al. (2018) unify the encoder and decoder but do not use a recurrent architecture, and use an attention mechanism. Kalchbrenner et al. (2015) and Bahar et al. (2018) do not use a typical encoder-decoder architecture, and instead use a 2D LSTM to translate, but these approaches are much slower than encoder-decoder sequence models.

Grissom II et al. (2014) and Gu et al. (2017) use reinforcement learning to train online translation models; the latter use an attention-based translation model. In parallel with our work, Ma et al.

(2018) employ a mechanism similar to our initial padding tokens to improve the performance of their simultaneous translation model. While our model does start outputting candidate translations as soon as the first input token is fed in, it is not a simultaneous translation model: we use beam search, which makes it possible (and very probable) for the top candidate translation to change when the next input token is consumed. While these simultaneous translation models have low latency, they do not manage to preform as well as non-simultaneous models.

For morphological inflection generation, Aharoni and Goldberg (2017) use an alignment model on the training data in order to know when to train the model to output a symbol and when it should read more of the source sequence, similar to our eager feasibility preprocessing.

6 Conclusion

We introduce a simple translation model that doesn't use attention and resembles a recurrent language model, and show that it preforms on par with a conventional attention model.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Parnia Bahar, Christopher Brix, and Hermann Ney. 2018. Towards two-dimensional sequence to sequence model in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3009–3015. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peter E. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Tobias Domhan. 2018. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648. Association for Computational Linguistics.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2018. Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction. In *CoNLL 2018 - Conference on Computational Natural Language Learning*, pages 1–11, Brussels, Belgium.

Mikel L. Forcada and Ramón P. Ñeco. 1997. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology, International Work-Conference on Artificial and Natural Neural Networks, IWANN '97, Lanzarote, Canary Islands, Spain, June 4-6, 1997, Proceedings*, pages 453–462.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Felix A. Gers, Jrgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *CoRR*, abs/1611.01462.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *CoRR*, abs/1507.01526.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Open-NMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Hao Xiong, Kaibo Liu, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, and Haifeng Wang. 2018. Stacl: Simultaneous translation with integrated anticipation and controllable latency. *arXiv preprint arXiv:1810.08398*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. *CoRR*, abs/1804.08771.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. *arXiv preprint arXiv:1704.00784*.

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 (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, and Hermann Ney. 2018. Neural hidden markov model for machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 377–382. Association for Computational Linguistics.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

A Supplementary Material

		FR→EN	EN→FR	DE→EN	EN→DE
Start εs	0	5, 4, 25	3, 13, 20	4, 7, 35	5, 7, 25
	1	5, 4, 25	4, 12, 20	4, 7, 35	5, 7, 25
	2	5, 4, 25	3, 14, 20	4, 9, 35	5, 9, 25
	3	6, 5, 35	1, 16, 5	4, 9, 35	4, 9, 25
	4	5, 6, 35	1, 15, 5	3, 9, 35	2, 9, 25
	5	4, 6, 25	2, 15, 5	3, 8, 35	1, 13, 5

Table 4: The padding limit, source padding injection value, and beam size used during inference for each model from Table 2.

	Source Sentence Length	Number of Sentences	Reference Model BLEU	Eager Model BLEU
FR→EN	1-10	162	**23.33**	12.43
	11-20	702	**26.54**	25.05
	21-30	713	**29.51**	28.96
	31-40	599	**29.49**	29.23
	41-50	431	**28.72**	27.68
	51-60	228	**28.68**	27.91
	61-70	110	**28.71**	28.22
	71-80	42	25.25	**27.14**
	81+	16	22.10	**27.44**
EN→FR	1-10	251	**26.31**	22.32
	11-20	834	**26.15**	23.74
	21-30	770	**29.33**	27.88
	31-40	589	**28.37**	27.49
	41-50	329	26.28	**26.85**
	51-60	151	26.88	**29.63**
	61-70	51	19.49	**25.46**
	71-80	19	22.30	**31.64**
	81+	9	12.55	**26.48**
DE→EN	1-10	203	**20.26**	12.61
	11-20	760	**23.32**	21.15
	21-30	756	**22.25**	22.09
	31-40	519	**23.93**	23.85
	41-50	258	**23.21**	22.38
	51-60	156	**22.84**	22.73
	61-70	58	23.50	**23.61**
	71-80	18	21.66	**23.24**
	81-+	9	21.24	**24.73**
EN→DE	1-10	211	**21.49**	14.78
	11-20	859	**18.72**	17.15
	21-30	781	**18.11**	17.64
	31-40	492	**19.43**	18.37
	41-50	227	**18.60**	18.00
	51-60	116	**18.18**	18.11
	61-70	38	19.87	**20.05**
	71-80	6	**15.79**	15.62
	81-+	7	**27.10**	26.41

Table 5: BLEU performance by source sentence length on all four language directions.