The RWTH Aachen University Supervised Machine Translation Systems for WMT 2018

Julian Schamper, Jan Rosendahl, Parnia Bahar,
Yunsu Kim, Arne Nix and Hermann Ney
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
<surname>@i6.informatik.rwth-aachen.de

Abstract

This paper describes the statistical machine translation systems developed at RWTH Aachen University for the German→English, English→Turkish and Chinese→English translation tasks of the *EMNLP 2018 Third Conference on Machine Translation* (WMT 2018). We use ensembles of neural machine translation systems based on the Transformer architecture. Our main focus is on the German→English task where we scored first with respect to all automatic metrics provided by the organizers. We identify data selection, fine-tuning, batch size and model dimension as important hyperparameters. In total we improve by 6.8% BLEU over our last year's submission and by 4.8% BLEU over the winning system of the 2017 German→English task. In English→Turkish task, we show 3.6% BLEU improvement over the last year's winning system. We further report results on the Chinese→English task where we improve 2.2% BLEU on average over our baseline systems but stay behind the 2018 winning systems.

1 Introduction

In this paper we describe the supervised statistical machine translation (SMT) systems developed by RWTH Aachen University for the news translation task of the *EMNLP 2018 Third Conference on Machine Translation* (WMT 2018). We use ensembles of neural machine translation systems to participate in the German→English, English→Turkish and Chinese→English tasks of the WMT 2018 evaluation campaign.

For this year's WMT we switch towards the Transformer architecture (Vaswani et al., 2017) implemented in Sockeye (Hieber et al., 2017). We experiment with different selections from the training data and various model configurations.

This paper is organized as follows: In Section 2 we describe our data preprocessing. Our translation software and baseline setups are explained in Section 3. The results of the experiments for the various language pairs are summarized in Section 4.

2 Preprocessing

For all our experiments on German, English and Turkish we utilize a simple preprocessing pipeline which consists of minor text normalization steps (e.g. removal of some special UTF-8 characters) followed by tokenization from Moses (Koehn et al., 2007) and frequent casing from the Jane toolkit (Vilar et al., 2010). The Chinese side is segmented using the `Jieba4` segmenter [1] except for the `Books 1-10` and `data2011` data sets which were already segmented as mentioned in (Sennrich et al., 2017).

We apply byte-pair encoding (BPE) to segment words into subword units for all language pairs (Sennrich et al., 2016b). Our BPE models are trained jointly for the source and the target language with the exception of the Chinese→English task. For every language pair we use the parallel data to train the BPE operations, excluding any synthetic data and the ParaCrawl corpus of the German→English task. To reduce the number of rare events we apply a vocabulary threshold of 50 as described in (Sennrich et al., 2017) in all our German→English systems. We end up with vocabulary sizes of 45k and 34k for German and English respectively if 50k joint merge operations are used.

3 MT Systems

All systems submitted by RWTH Aachen are based on the Transformer architecture imple-

[1] https://github.com/fxsjy/jieba

Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 496–503

https://doi.org/10.18653/v1/W18-64053

mented in the Sockeye sequence-to-sequence framework for Neural Machine Translation. Sockeye is built on the Python API of MXNet (Chen et al., 2015).

In the Transformer architecture both encoder and decoder consist of stacked layers. A layer in the encoder consists of two sub-layers: a multi-head self-attention layer followed by a feed forward layer. The decoder contains an additional multi-head attention layer that connects encoder and decoder. Before and after each of these sub-layers preprocessing respectively postprocessing operations are applied. In our setup layer normalization (Ba et al., 2016) is applied as preprocessing operation while the postprocessing operation is chosen to be dropout (Srivastava et al., 2014) followed by a residual connection (He et al., 2016).[2] For our experiments we use 6 layers in both encoder and decoder and vary the size of their internal dimension. We set the number of heads in the multi-head attention to 8 and apply label smoothing (Pereyra et al., 2017) of 0.1 throughout training.

We train our models using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001 (for En→Tr and Zh→En) respectively 0.0003 (for De→En). A warmup period with constant or increasing learning rate was not used. We employ an annealing scheme that scales down the learning rate if no improvement in perplexity on the development set is seen for several consecutive evaluation checkpoints. During training we apply dropout ranging from 0.1 to 0.3. All batch sizes are specified on the token level and are chosen to be as big as the memory of the GPUs allows. In case of the utilization of multiple GPUs we use synchronized training, i.e. we increase the effective batch size, which seems to have better convergence properties (Popel and Bojar, 2018).

The weights of embedding matrices and the projection layer prior to the softmax layer are not shared in our architecture and for all translation runs a beam size of 12 is used.

4 Experimental Evaluation

In this section we present our results on the three tasks we participated in, with the primary focus on building a strong system for the German→English system.

[2]Note that this is by now also the default behavior of the Tensor2Tensor implementation https://github.com/tensorflow/tensor2tensor, differing from the original paper.

Data Resource	# lines
WMT 2018 (standard parallel data)	5.9M
ParaCrawl (filtered 50%)	18.2M
NewsCrawl 2015 (UEDIN WMT16)	4.2M
NewsCrawl 2017 (random 50%)	13.4M
newstest2008–2014 (for fine-tuning)	19.1k

Table 1: Number of parallel sentence pairs or monolingual sentences of our different training data resources (De→En). Our strongest systems from Table 3 use all these resources.

For evaluation we use `mteval-v13a` from the Moses toolkit (Koehn et al., 2007) and `TERCom`[3] to score our systems on the BLEU (Papineni et al., 2002) respectively TER (Snover et al., 2006) measures. In addition we report CTER scores[4] (Wang et al.). All reported scores are given in percentage and the specific options of the tools are set to be consistent with the calculations of the organizers.

4.1 German→English

In most experiments for the German→English task we use a subset of the data resources listed in Table 1. All models use the Transformer architecture as described in Section 3. Our baseline model is very similar to the "base" Transformer of the original paper (Vaswani et al., 2017), e.g. $d_{\text{model}} = 512$ and $d_{\text{ff}} = 2048$, however we do not use weight-tying.

Throughout our experiments we analyze various aspects of our experimental setup (e.g. several data conditions or the model size). We evaluate our models every 20k iterations and select the best checkpoint based on BLEU calculated on our development set `newstest2015` afterwards. To handle all different variations in a well organized way, we use the workflow manager Sisyphus (Peter et al., 2018).

In Table 2 we carefully analyze different data conditions. We can see that the Transformer model with 20k BPE merging operation already beats our last year's final submission by 1.4% BLEU. The Transformer model was trained using the standard parallel WMT 2018 data sets (namely Europarl, CommonCrawl, NewsCommentary and Rapid, in total 5.9M sentence pairs) as well as the 4.2M sen-

[3]http://www.cs.umd.edu/~snover/tercom/
[4]https://github.com/rwth-i6/CharacTER

	Systems	BPE	newstest2015 (dev) BLEU	TER	cTER
1	RWTH WMT 2017		32.0	-	-
2	Transformer	20k	33.4	52.7	48.4
3	+ ParaCrawl	20k	30.3	57.8	51.9
4	Transformer	50k	33.9	52.5	47.9
5	+ ParaCrawl	50k	31.3	56.5	50.7
6	+ Oversample	50k	31.6	56.3	50.5
7	+ Filtering	50k	34.7	51.7	46.9

Table 2: Baseline results and analysis of data conditions (De→En). Our baseline starts of with the standard WMT 2018 training data excluding ParaCrawl but including already backtranslated NewsCrawl 2015. "Filtering" refers to filtering ParaCrawl only (50% LM driven).

tence pairs of synthetic data created in (Sennrich et al., 2016a). Last year's submission is an ensemble of several carefully crafted models using an RNN-encoder and decoder which was trained on the same data plus 6.9M additional synthetic sentences (Peter et al., 2017).

We try 20k and 50k merging operations for BPE and find that 50k performs better by 0.5% to 1.0% BLEU. Hence, we use this for all further experiments. As Table 2 shows, just adding the new ParaCrawl corpus to the existing data hurts the performance by up to 3.1% BLEU.

To counter this effect we oversample CommonCrawl, Europarl and NewsCommentary with a factor of two. Rapid and the synthetic news data are not oversampled. As we can observe in Row 6 of Table 2 this gives a minor improvement, but is not enough to counter the negative effects from adding ParaCrawl. Therefore we train a 3-gram language model on the monolingual English NewsCrawl2007-2017 data sets using KenLM (Heafield, 2011) to rank the corpus and select the best 50% of sentence pairs. Together with oversampling this yields an improvement of 3.4% BLEU over the naive concatenation of all training data and 0.8% BLEU over the corresponding system that does not use ParaCrawl at all.

Using the best data configuration described we start to use multiple GPUs instead of one and increase the model size. Each GPU handles a share of the data and the update steps are synchronized, such that the effective batch size is increased. As before we choose the batch size on word level in such a way that the memory of all GPUs is fully used. Note that due to time constraints and the size of the models the reported results are taken from models which did not yet fully converge. Each model in Table 3 is trained using 4 GPUs for close to 8 days.

First we double the dimension of the model to $d_{\text{model}} = 1024$. As can be seen from Table 3, together with the increased batch size, this yields a major improvement of 1.2% BLEU on `newstest2015`.

Using a basic English→German system we backtranslate 26.9M sentences from the NewsCrawl 2017 monolingual corpus. This system uses the same transformer configuration as used for the baseline De→En system and is trained on the standard parallel WMT 2018 dataset (5.9M sentence pairs). It achieves 28.7% BLEU and 38.9% BLEU on `newstest2015` and `newstest2018` respectively. After experimenting with several thresholds we added half of the backtranslated data (randomly selected) to our training data which gave us 0.5% BLEU extra on the development set. Even though the system is trained on 17.6M synthetic news sentences from NewsCrawl 2015 (4.2M) and NewsCrawl 2017 (13.4M), fine-tuning on old test sets (`newstest2008` to `newstest2014`) improves it by 0.6% BLEU on `newstest2015` and 1.0% BLEU on `newstest2017`. We set the checkpoint frequency down to 50 updates only and select the best out of 14 fine-tuned checkpoints (selected on `newstest2015`). Overall we find it beneficial to match the learning conditions which are present for the checkpoint which is fine-tuned: Especially important seems to be the usage of a similar learning rate in contrast to using the comparably high initial learning rate (0.0003).

Adding an extra layer to encoder and decoder did not change the performance of the system significantly. However the model was helpful in the final ensemble. Similarly increasing the dimension of the hidden size of the feed forward layers to 4096 and setting the number of attention heads to 16 barely changed the performance of the system. It turns out to be helpful if we double the batch size of the model. Because the GPUs available to us can not handle bigger batches we increased the effective batch size further by accumulating gradient updates before applying them. The resulting system shown in Table 3 Row 7 is the best single system provided by RWTH Aachen for

	Systems	newstest2015 (dev) BLEU	TER	cTER	newstest2017 BLEU	TER	cTER	newstest2018 BLEU	TER	cTER
1	Baseline (Table 2 Row 7)	34.7	51.7	46.9	36.4	50.7	47.1	44.5	41.4	39.6
2	+ d_{model} =1k + 4GPUs	35.9	50.9	46.2	37.6	49.6	45.9	46.1	40.4	38.5
3	+ NewsCrawl17 (50%)	36.4	50.1	45.7	38.3	49.2	46.3	46.8	39.6	38.2
4	+ fine-tuning	37.0	49.5	45.5	39.3	48.1	45.2	47.5	38.9	37.7
5	+ 7th layer	37.2	49.6	45.3	39.3	48.0	45.0	47.5	39.0	37.8
6	+ d_{ff} =4k + 16 heads	37.0	49.8	45.6	39.0	48.4	45.1	47.5	38.8	37.6
7	+ GradAcc=2	37.1	49.2	45.3	39.5	48.0	45.0	47.6	38.8	37.6
8	Ensemble [4,5,7]	37.5	49.1	44.9	39.9	47.6	44.6	48.4	38.1	36.9

Table 3: Main results for the German→English task (Rows 7 and 8 show the submitted system). Note that using multiple GPUs does not only result in a higher data throughput but also multiplies the effective batch size and therefore affects the convergence. However if only one GPU is available the results could be still reproduced by using just gradient accumulation.

Systems	nt2015 BLEU	nt2017 BLEU	TER	cTER
Winner 2017	-	35.1	52.4	48.9
RWTH 2017	32.0	33.1	54.2	-
RWTH 2018	37.5	39.9	47.6	44.6

Table 4: Comparison with last years' submissions on `newstest2015+2017` (De→En). The winning system of 2017 was submitted by UEDIN. Missing scores are due to inconsistent calculations or unavailability.

the German→English task.

Because checkpoint averaging helped in the past we tried several versions based on last or best checkpoints of different distances but no version turned out to be helpful in our case.

Finally model ensembling brought performance up to 37.5% BLEU and 39.9% BLEU on `newstest2015` and `newstest2017`. Overall we achieved an improvement of 2.8% and 3.5% BLEU over our baseline.

Table 4 shows that we improved our system by 6.2% BLEU on average on `newstest2015+2017` since previous year and by 4.8% BLEU on `newstest2017` over the winning system of 2017 (Sennrich et al., 2017).

4.2 English→Turkish

The English→Turkish task is in a low-resource setting where the given parallel data consists of only around 200k sentences. We therefore apply dropout to various parts of our Transformer model: attention/activation of each layer, pre/post-processing between the layers, and also embedding—with a dropout probability of 0.3. This gives a strong regularization and yields 2.6% BLEU improvement compared to the baseline in `newstest2018` (Row 2 of Table 5).

Although the English and Turkish languages are from different linguistic roots, we find that the performance is better by 4.5% BLEU in `newstest2018` when sharing their vocabularies by tying the embedding matrices (Row 3 of Table 5). They are also tied with the transpose of the output layer projection as done in (Vaswani et al., 2017). We accordingly use BPE tokens jointly learned for both languages (20k merge operations). Since the training signals are weak from the given data, we argue that this kind of parameter sharing helps to avoid overfitting and copy proper nouns correctly.

Checkpoint frequency is set to 4k. Other model parameters and training hyperparameters are the same as described in Section 3.

Table 5 also shows results with back-translated data from Turkish News Crawl 2017 (Row 4, +3.8% BLEU in `newstest2018`). Using more than 1M sentences of back-translations does not help, which might be due to the low quality of back-translations generated with a weak model (trained only with 200k parallel sentences). Note that we oversample the given parallel data to make the ratio of the parallel/synthetic data 1:1. An ensemble of this setup with four different random seeds shows a slight improvement up to 0.2% BLEU (Row 4 vs. 6).

Finally, we fine-tune the models with `newstest2016+2017` sets to adapt to the news domain. We set the learning rate ten times lower (0.00001) and the checkpoint frequency to 100. Dropout rate is reduced to 0.1 for a fast adaptation. This provides an additional boost of

	Systems	newsdev2016 (dev) BLEU	TER	cTER	newstest2017 BLEU	TER	cTER	newstest2018 BLEU	TER	cTER
1	Baseline	6.8	93.0	71.4	7.4	91.6	76.7	7.1	92.1	73.0
2	+ dropout 0.3	9.3	84.9	67.0	10.3	83.4	73.4	9.7	84.5	68.0
3	+ weight tying	14.2	76.9	59.7	15.8	74.5	61.6	14.2	77.3	62.1
4	+ BT 1M sents	16.7	72.3	56.3	20.0	68.0	56.8	17.0	72.3	58.5
5	+ fine-tuning	17.6	71.2	56.6	25.0	62.1	53.7	17.7	71.3	58.5
6	Ensemble 4x [4]	16.7	71.8	56.1	20.1	67.7	56.7	17.2	72.0	58.3
7	Ensemble 4x [5]	17.7	70.6	55.8	25.1	62.1	53.1	18.0	71.0	58.0

Table 5: English→Turkish results. Row 6 is the submitted system.

0.7% BLEU for the single model (Row 4 vs. 5) and 0.8% BLEU for the ensemble (Row 6 vs. 7) in `newstest2018`.

4.3 Chinese→English

We use all available parallel data totaling 24.7M sentence pairs with 620M English and 547M Chinese words and follow the preprocessing described in Section 2. We then learn BPE with 50k merge operations on each side separately. `newsdev2017` and `newstest2017` containing 2002 and 2001 sentences are used as our development and test sets respectively. We also report results on `newstest2018` with 3981 samples. We remove sentences longer than 80 subwords. We save and evaluate the checkpoints according to the BLEU score on the development set every 10k iterations.

In order to augment our training data, we back-translate the NewsCrawl2017 monolingual corpus consisting of approximately 25M samples using a En→Zh NMT system resulting in a total of 49.5 sentence pairs for training. The En→Zh NMT model is based on the RNN with attention encoder-decoder architecture (Bahdanau et al., 2014) implemented in Returnn[5] (Zeyer et al., 2018). The network is similar to (Bahar et al., 2017) with 4-layer of bidirectional encoders using long-short term memory cells (LSTM) (Hochreiter and Schmidhuber, 1997). We apply a layer-wise pre-training scheme that leads to both better convergence and faster training speed during the initial pre-train epochs (Zeyer et al., 2018). We start using only the first layer in the encoder of the model and add new layers during the training progress. We apply a learning rate scheduling scheme, where we lower the learning rate if the perplexity on the development set does not improve anymore.

For Zh→En, we run different Transformer configurations which differ slightly from the model described in Section 3. Our aim is to investigate the effect of various hyperparameters especially the model size, the number of layers and the number of heads. According to the total number of parameters, we call these models as below:

- **Transformer base**: a 6-layer multi-head attention (8 heads) consisting of 512 nodes followed by a feed forward layer equipped with 1024 nodes both in the encoder and the decoder. The total number of parameters is 121M. Training is done using mini-batches of 3000.

- **Transformer medium**: a 4-layer multi-head attention (8 heads) consisting of 1024 nodes followed by a feed forward layer equipped with 4096 nodes both in the encoder and the decoder. The total number of parameters is 271M. Training is done using mini-batches of 2000.

- **Transformer large**: a 6-layer multi-head attention (16 heads) consisting of 1024 nodes followed by a feed forward layer equipped with 4096 nodes both in the encoder and the decoder. The total number of parameters is 330M. Training is done using mini-batches of 6500 on 4 GPUs.

The results are shown in Table 6. Note that all models are trained using bilingual plus synthetic data. Comparing the Transformer base and medium architectures shows that model size is more important for strong performance than the number of layers. Adding more layers with big

[5] https://github.com/rwth-i6/returnn

	Systems	newsdev2017 (dev) BLEU	TER	cTER	newstest2017 BLEU	TER	cTER	newstest2018 BLEU	cTER
1	Base	23.3	66.8	62.1	23.9	67.2	61.5	24.1	63.8
2	Medium	24.5	65.5	61.0	24.8	65.8	60.9	25.4	63.2
3	Large	24.6	65.2	60.7	25.3	65.6	60.5	26.0	62.8
4	Ensemble (linear) [1,2,3]	25.4	64.5	59.9	25.9	64.9	59.9	26.7	61.8
5	Ensemble (log-linear) [1,2,3]	25.4	64.4	60.1	26.1	64.8	59.4	26.4	62.0
6†	Ensemble (linear) 4 checkpoints of [3]	24.4	65.4	60.9	25.6	65.2	60.1	26.7	62.1

Table 6: Results measured in BLEU [%], TER [%] and cTER [%] for Chinese→English. The TER computation on newstest2018 fails. † indicates the submitted system which is the ensemble of 4 non-converged checkpoints of the large Transformer.

model size and increasing the batch size up to 6500 provides an additional boost of 0.4% BLEU, 0.3% TER and 0.4% cTER on average on all sets (see Row 2 and 3). Furthermore, we try an ensemble of best checkpoints based on BLEU either using various models or using different snapshots of the large Transformer. We use both linear and log-linear ensembling which does not make a difference in terms of BLEU as shown in the Table. Log-linear ensembling is slightly better in terms of TER and is a little bit worse in terms of cTER. We also combine the 4 best checkpoints of the large Transformer shown in Row 6 of Table 6.

5 Conclusion

This paper describes the RWTH Aachen University's submission to the WMT 2018 shared news translation task. For German→English our experiments start with a strong baseline which already beats our submission to WMT 2017 by 1.4% BLEU on newstest2015. Our final submission is an ensemble of three Transformer models which beats our and the strongest submission of last year by 6.8% BLEU respectively 4.8% BLEU on newstest2017. It is ranked first on newstest2018 by all automatic metrics for this year's news translation task[6]. We suspect that the strength of our systems is especially grounded in the usage of the recently established Transformer architecture, the usage of filtered ParaCrawl in addition to careful experiments on data conditions, the usage of rather big models and large batch sizes, and effective fine-tuning on old test sets.

In English→Turkish task, we show that proper regularization (high dropout rate, weight tying) is crucial for the low-resource setting, yielding a total of up to +7.4% BLEU. Our best system is using 1M sentences synthetic data generated with back-translation (+2.8% BLEU), fine-tuned with test sets of previous year's tasks (+0.7% BLEU), and ensembled over four different training runs (+0.3% BLEU), leading to 18.0% BLEU in newstest2018. Note that its cTER is better or comparable to the top-ranked system submissions[7]. In newstest2017, our system, even if it is not fine-tuned, outperforms the last year's winning system by +3.6% BLEU.

For our Chinese→English system multiple GPU training that allows for larger models and an increased batch size results in the best preforming single system. A linear ensemble of different Transformer configurations provides 0.7% BLEU, 0.6% TER and 0.8% cTER on average on top of the single best model.

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 the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG.
The work reflects only the authors' views and none of the funding agencies is responsible for

[6] http://matrix.statmt.org/matrix/systems_list/1880

[7] http://matrix.statmt.org/matrix/systems_list/1891

any use that may be made of the information it contains.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*. Version 1.

Parnia Bahar, Jan Rosendahl, Nick Rossenbach, and Hermann Ney. 2017. The RWTH Aachen machine translation systems for IWSLT 2017. In *14th International Workshop on Spoken Language Translation*, pages 29–34.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*. Version 1.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690*. Version 2.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Version 9.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Jan-Thorsten Peter, Eugen Beck, and Hermann Ney. 2018. Sisyphus, a workflow manager designed for machine translation and automatic speech recognition. In *2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Jan-Thorsten Peter, Andreas Guta, Tamer Alkhouli, Parnia Bahar, Jan Rosendahl, Nick Rossenbach, Miguel Graa, and Ney Hermann. 2017. The RWTH Aachen University English-German and German-English machine translation system for WMT 2017. In *EMNLP 2017 Second Conference on Machine Translation*, Copenhagen, Denmark.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 389–399.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, volume 2: Shared Task Papers, pages 368–373, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. 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)*, volume 1, 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 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

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 5998–6008.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. 2010. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 262–270. Association for Computational Linguistics.

Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. CharacTer: Translation Edit Rate on Character Level. In *The 54th Annual Meeting of the Association for Computational Linguistics - proceedings of the First Conference on Machine Translation (WMT) : August 7-12, 2016, Berlin, Germany : ACL 2016. - Volume 2, Shared Task Papers*, pages 505–510, Stroudsburg, PA. Association for Computational Linguistics.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 128–133.

6 Appendix

For our very first experiments with Sockeye a configuration[8] from the Sockeye git repository provided a good starting point.

For our strongest single De→En model (Table 3, Row 7) we ended up with the following options:

```
--num-layers 6:6
--encoder transformer
--decoder transformer
--num-embed 1024:1024
--transformer-model-size 1024
--transformer-feed-forward
   -num-hidden 4096
--transformer-attention-heads 16
--transformer-positional
   -embedding-type fixed
--transformer-preprocess n
--transformer-postprocess dr
--embed-dropout 0:0
--transformer-dropout-prepost 0.1
--transformer-dropout-act 0.1
--transformer-dropout
   -attention 0.1
--label-smoothing 0.1
--learning-rate-reduce-num
   -not-improved 3
--checkpoint-frequency 20000
--batch-type word
--batch-size 5000
--device-ids -4
--grad-accumulation 2     # see*
```

* Note that the `--grad-accumulation` option is introduced by us and is not provided by the official Sockeye version. It refers to the accumulation of gradients, described in Section 4.1, which increases the effective batch-size: In the provided config the effective batch size is 10000.

For our vocabulary sizes (45k and 34k for German and English) the listed configuration results in a Transformer network with 291M trainable parameters.

[8] https://github.com/awslabs/sockeye/blob/arxiv_1217/arxiv/code/transformer/sockeye/train-transformer.sh