The RWTH Aachen University English-German and German-English Unsupervised Neural Machine Translation Systems for WMT 2018

Miguel Graça, Yunsu Kim, Julian Schamper, Jiahui Geng and Hermann Ney
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{graca,kim,schamper,jgeng,ney}@i6.informatik.rwth-aachen.de

Abstract

This paper describes the unsupervised neural machine translation (NMT) systems of the RWTH Aachen University developed for the English ↔ German news translation task of the *EMNLP 2018 Third Conference on Machine Translation* (WMT 2018). Our work is based on iterative back-translation using a shared encoder-decoder NMT model. We extensively compare different vocabulary types, word embedding initialization schemes and optimization methods for our model. We also investigate gating and weight normalization for the word embedding layer.

1 Introduction

Unsupervised NMT was recently investigated in (Artetxe et al., 2017; Lample et al., 2017, 2018) and has shown promising results in language pairs like German to English. For the WMT 2018 unsupervised learning track, we combine the concepts proposed in previous research and perform a thorough comparison of the main components of each method. Additionally, we augment the word embedding initialization with weight normalization to improve its integration in the model and with a gating technique to allow the model to learn task specific information.

The main findings of this paper are: (i) the iterative method (Lample et al., 2017) outperforms the online training method (Artetxe et al., 2017), (ii) cross-lingual embedding initialization is required in the online method and (iii) byte-pair encoding (BPE)-based vocabularies (Sennrich et al., 2016) outperform word-based vocabularies in online training.

This paper is organized as follows: Section 2 describes pre- and postprocessing pipelines, corpora selection and vocabularies used in our experiments. Section 3 details the models used in this work together with the embedding augmentation techniques. The experimental evaluation is presented in Sections 4 and 5 and finally we conclude with Section 6.

2 Pre- and Postprocessing

Our preprocessing pipeline consists of a tokenization with a script from the Moses toolkit (Koehn et al., 2007), lower-casing, and the introduction of a number category token which replaces all occurrences of numbers. We use joint BPE in our experiments and apply it at this stage of preprocessing.

After the search procedure, we first monotonically replace number tokens with their original content, and unknown words to the target hypothesis by their order of occurrence in the source sentence. This method is very restrictive, as it fails when, e.g., more unknown tokens are hypothesized than there are in the source sentence due to an unknown token being attended twice. Since, to our knowledge, there are no well-founded methods of pin-pointing which source words are attended during the generation of a target word in the Transformer (Vaswani et al., 2017), we decided for the forementioned method.

As postprocessing, we first convert subwords to words. Lower-cased words are then frequent-cased using the tools provided in the Jane toolkit (Vilar et al., 2010). As a final step, the text is detokenized using the detokenizer from Moses and punctuation is normalized.

2.1 Corpora Selection

We use monolingual News Crawl articles from 2014 to 2017[1] as our training corpora for both German and English languages. $100M$ sentences are sub-sampled for pre-training word embeddings and $5M$ sentences are used for translation model training.

[1] http://www.statmt.org/wmt18/translation-task.html

	German	English
# sentences	5M	5M
Vocabulary	1.3M	577K
OOV rate	6.9% / 8.9%	1.73% / 3.3%
Effective voc.	50K / 46.3K	50K / 31.2K

Table 1: Corpus statistics for the German and English monolingual corpora. OOV word rates and effective vocabulary sizes are given for unshared and shared, respectively displayed, vocabularies limited to the most frequent $50k$ words.

Table 1 shows the corpus statistics for the model training. The reported out-of-vocabulary (OOV) word rates and effective vocabulary sizes are shown for our word-level experiments, which use either top frequent $50k$ words for each language or a shared vocabulary with a total of $50k$ words.

Even though the News Crawl corpora contain mostly clean data, we noticed that common English words are found in the German corpus and vice-versa, which causes an overlap in the effective vocabulary.

2.2 Vocabulary

In this work we consider different kinds of vocabularies for the unsupervised translation systems. Lample et al. (2017) use word-level vocabularies due to the initialization with a word-by-word model. Artetxe et al. (2017) perform experiments on both word- and BPE-level and report that the model has difficulties translating rare sub-word codes. We perform experiments with both BPE and word vocabularies to find the best setting for unsupervised NMT.

For the BPE vocabularies, we consider only the joint variant, performing $20k$ and $50k$ merge operations. The word-level vocabulary is restricted to a $50k$-word shared vocabulary or two seperate $50k$-word German and English vocabularies.

3 Unsupervised Neural Machine Translation

The models proposed in Lample et al. (2017); Artetxe et al. (2017) follow a recurrent attention-based encoder-decoder architecture (Bahdanau et al., 2015). As a follow-up work Lample et al. (2018) make use of the Transformer (Vaswani et al., 2017) architecture, which we also utilize for our systems.

3.1 Model Description

We closely follow the model architecture in (Lample et al., 2017), but with a Transformer encoder-decoder. It is able to translate in both source to target and target to source translation directions via joint training and parameter sharing of components, therefore we denote it further as a shared architecture. In this section, we describe how the model functions for an input source sentence $f_1^J = f_1, ..., f_j, ..., f_J$ and output target sentence $e_1^I = e_1, ..., e_i, ..., e_I$.

The model consists of an self-attentive encoder and decoder, word embeddings and output layers, where the encoder and decoder share parameters in both translation directions. The output layer may additionally be shared when the output vocabularies are also shared between both directions.

Word embeddings: Each word is encoded in a continuous space of dimension D via a lookup table function $E : V \rightarrow \mathbb{R}^D$, where V represents the source or target vocabulary, scaled up by $\sqrt{D}$ as in the original formulation (Vaswani et al., 2017). Fixed positional embeddings $pos : \mathbb{N}_0 \rightarrow \mathbb{R}^D$ (Vaswani et al., 2017), which encode the absolute position j of a word f_j in the source sentence, are added to the word vectors to represent a word embedding:

$$\bar{f}_j = E_f(f_j) \cdot \sqrt{D} + pos(j) \tag{1}$$

Source word embeddings are applied whenever the model reads a source sentence or outputs a source sentence. All of the above hold analogously for the target word embeddings.

Encoder: The input source embeddings are read by a self-attentive encoder module and outputs a sequence of hidden states h_1^J with $h_j \in \mathbb{R}^D$ having the same dimensionality as the input embeddings.

$$h_1^J = H(\bar{f}_1^J; \theta_{enc}) \tag{2}$$

A noise model as described in (Lample et al., 2017) is applied to the encoder inputs.

Decoder: Target word predictions are conditioned on the sequence of previously seen embedded target words $\bar{e}_0^{i-1}$ and the encoder outputs h_1^J. The decoder outputs a single hidden state $s_i \in \mathbb{R}^D$, which is then propagated to an output layer. Note that in our setup encoder and decoder outputs have the same dimensionality.

$$s_i = S(h_1^J, \bar{e}_0^{i-1}; \theta_{dec}) \tag{3}$$

The target sentence is augmented with a sentence start symbol e_0, which is an identifier for the output language. In our setup the decoder is shared between languages.

Output layer: The hidden state s_i is projected to the size of the output vocabulary and normalized with a softmax operation resulting in a probability distribution over target words.

$$p(e_i|e_1^{i-1}, f_1^J) = \text{softmax}(W \cdot s_i + b)_{e_i} \quad (4)$$

As mentioned in Section 2.2, the output layer may or may not be shared depending on the type of vocabularies.

Optimization: The model is trained via cross-entropy on both translation directions. Additionally, we include auto-encoding losses for both languages for a total of four optimization criteria as in both approaches (Artetxe et al., 2017; Lample et al., 2017).

Finally, we include an adversarial loss term (Lample et al., 2017) in a feature study experiment, where the model is trained to fool a separate model that attempts to discriminate the language of the input sentence after the encoder module. Each component of the loss function is equally weighted.

Note that, in contrast to (Artetxe et al., 2017), we do not alternate between loss functions during optimization and instead optimize the summation of them. We noticed the same translation quality when comparing both sum and alternating variants in preliminary experiments.

3.2 Batch Optimization

Proposed by Lample et al. (2017), the batch optimization method trains the model iteratively: the model trained on iteration $n-1$ is used to generate back-translations to train the model at iteration n. The initial model is an unsupervised word-by-word translation model based on cross-lingual word vectors (Conneau et al., 2017).

The workflow of this method for the n-th iteration is as follows:

1. Translate monolingual corpora with the model at iteration $n-1$
2. Train for one epoch on the back-translated and monolingual corpora

Throughout this work, we denote an iteration as the forementioned steps. We restrict ourselves to only one epoch for model training per iteration, but it is also possible to train for a different amount of updates.

3.3 Online Optimization

Leveraging the model's ability to translate in both translation directions, Artetxe et al. (2017); Lample et al. (2018) generate back-translations for each mini-batch using the currently trained parameters. This method is not initialized with a word-by-word translation.

We noticed that with the original implementation training was slow due to the generation of back-translations with a smaller batch size than what fit in our device's memory. Therefore, we implement online optimization by generating 10 mini-batches of back-translations at once. We noticed no loss of translation quality when doing this.

3.4 Gated Word Embeddings

The initialization of the word embeddings with pre-trained word vectors allows the model to start from a much more informative state and exploit information from a larger corpus. Indeed it is a crucial component of the shared architecture, as shown empirically in Section 5.2. As an alternative to just training the initialized vector, we consider a gating mechanism, shown in Equation 5 and introduced in (Yang et al., 2016):

$$\begin{aligned}\bar{f}_j = \Big(& g(f_j) \odot E_{f,pre-train}(f_j) \\ & + (1 - g(f_j)) \odot E_{f,random}(f_j) \Big) \\ & \cdot \sqrt{D} + pos(j)\end{aligned} \quad (5)$$

with the interpolation weights $g(f_j) \in \mathbb{R}^D$ being defined as a feed-forward projection to the word embeddings' dimensionality with a sigmoidal output:

$$\begin{aligned}g(f_j) = \sigma(b + W \cdot \Big[& E_{f,pre-train}(f_j), \\ & E_{f,random}(f_j) \Big])\end{aligned} \quad (6)$$

$\odot$ denotes element-wise multiplication. This allows the model to learn task-specific information and interpolate it with the pre-trained parameters. When using this approach, the pre-trained vectors are not updated during training.

Ding and Duh (2018) perform a simpler approach to combine both kinds of embeddings, in which they concatenate the word vectors and, as in this work, keep the pre-trained embeddings fixed during training.

Our idea is most similar to the concept in (Yang et al., 2018), where the authors also employ a gating mechanism on the embeddings, but combine it with the output of the encoder in order to reinforce a language-independent encoder representation.

3.5 Embedding Weight Normalization

The training criteria for word embeddings does not enforce normalization constraints on the continuous output values and therefore might cause very high or low gradient values in the encoder and decoder parameters, especially at the beginning of training.

Weight normalization (Salimans and Kingma, 2016), as shown in Equation 7, normalizes each word embedding by its L_2-norm and introduces an additional tunable parameter v_{f_j} for each word, that rescales the vector. It is initialized with the value of 1.

$$\bar{f}_j = \frac{v_{f_j} \cdot E_f(f_j) \cdot \sqrt{D}}{||E_f(f_j)||_2} + pos(j) \qquad (7)$$

4 Experimental Setup

All processing steps and experiments were organized with Sisyphus (Peter et al., 2018) [2] as workflow manager.

4.1 Model Hyperparameters

Our models use the Transformer architecture (Vaswani et al., 2017) implemented in Sockeye (Hieber et al., 2017), based on MXNet (Chen et al., 2015). The encoder and decoder both have 4 layers of size 300 with the internal feed-forward operation having 2048 nodes. The multi-head attention mechanism uses 6 heads. For each encoder and decoder layer, 10% dropout (Srivastava et al., 2014) and layer normalization (Ba et al., 2016) are used as preprocessing[3] operations and a residual connection (He et al., 2016) is additionally included in the postprocessing operations.

Monolingual word embeddings have a dimensionality of 300 and are trained as a skip-gram model using FastText (Bojanowski et al., 2017), only for words that have occured at least 10 times. Cross-lingual word embeddings are trained with MUSE (Conneau et al., 2017) for 10 epochs with the adversarial setting and 10 steps of the refinement procedure using the learned monolingual embeddings.

Model optimization is performed with the AdaM (Kingma and Ba, 2014) algorithm using a learning rate of 10^{-4} and a momentum parameter $\beta_1 = 0.5$. Training sequences are limited to 50 words or subwords. Parameters are initialized with Glorot initialization (Glorot and Bengio, 2010). The batch method is trained for 5 iterations, $800K$ updates, for a total of 6 days and the online method is trained for roughly the same amount of time for $500K$ updates.

Translation is performed using beam search with beam size 5 and the best hypothesis is the one with the lowest length normalized negative log-probability. Length normalization divides the sentence score by the number of words.

4.2 Evaluation

We constrain our results to the newstest2017 and newstest2018 data sets in the German → English translation direction. BLEU (Papineni et al., 2002), computed with mteval from the Moses toolkit (Koehn et al., 2007), and TER (Snover et al., 2006), computed with TERCom, are used as evaluation metrics. BLEU scores are case-sensitive and TER is scored lower-cased. All presented scores are percentages. For the experiments in Sections 5.3 and 5.4 we additionally test for statistical significance with MultEval (Clark et al., 2011).

Lample et al. (2017) propose a model selection criterion based on round-trip BLEU scores, however we do not notice a correlation of this measure and BLEU between experiments. The more expressive the model is, the better round-trip BLEU scores it will get, whereas BLEU itself does not change. Therefore we choose to validate on newstest2015 on the German → English translation direction for the feature study.

For our final submission, we select optimization method, embedding initialization and vocabulary types based on BLEU on the German → English direction of newstest2017 and select the best hyperparameter settings using the metric from Lample et al. (2017). In this case, we only consider models that have trained exactly 6 iterations.

5 Experimental Results

5.1 Translation Units

We experiment with both words and BPE subwords as initial work (Artetxe et al., 2017; Lample

[2] https://github.com/rwth-i6/sisyphus

[3] Pre- and postprocessing terminology is described in (Hieber et al., 2017).

		newstest2017		newstest2018	
	method	BLEU	TER	BLEU	TER
words	batch	**14.9**	**72.7**	**18.1**	**67.1**
unshared		14.5	73.3	17.2	67.8
words	online	11.9	75.7	14.2	71.0
unshared		10.6	77.7	13.2	73.1
BPE 20k		11.8	77.9	13.6	73.9
BPE 50k		**13.1**	**75.5**	**15.4**	**70.8**

Table 2: Vocabulary comparison between different optimization methods for German → English. All systems are initialized with cross-lingual word embeddings.

et al., 2017) focuses primarily on words and only briefly discuss the effects of sub-word units.

Table 2 shows the effect of different vocabulary sizes and units on both online and batch optimization methods. The best performing experiments are trained with batch optimization and a word-based vocabulary, even though they face an OOV word problem during both training and testing. Furthermore restricting the vocabulary to the top-$50k$ most common words in both vocabularies and sharing an output layer performs up to 0.9% BLEU and 0.7% TER better than using separate top-$50k$ vocabularies (different output layers). The same effect is noticeable with the online optimization method.

The online optimization method is additionally run with joint BPE codes trained with $20k$ and $50k$ merge operations, which improves over the word-based vocabulary by up to 1.2% BLEU and 0.2% TER when using $50k$ operations.

We do not present results with the batch method and BPE-based vocabularies, because the initial word-by-word translation is designed to work on the word-level.

5.2 Embedding Initialization

Initializing word embeddings with pre-trained vectors was a component in both original works (Artetxe et al., 2017; Lample et al., 2017). Two kinds of embeddings are considered, monolingual and cross-lingual, both serving the role of initializing the model with prior knowledge to aid the training of the model. Cross-lingual embeddings further add the property of language abstraction to pre-trained monolingual vectors.

Results on the embedding initialization are reported in Table 3 for both batch and online optimization methods.

		newstest2017		newstest2018	
	method	BLEU	TER	BLEU	TER
random	online	4.9	92.7	4.9	91.7
monolingual		7.5	88.2	8.2	85.7
cross-lingual		**13.1**	**75.5**	**15.4**	**70.8**
+ frozen		12.7	76.3	15.1	71.6
random	batch	14.5	73.6	17.6	68.2
monolingual		14.3	73.3	17.2	68.0
cross-lingual		**14.9**	**72.7**	**18.1**	**67.1**
+ frozen		14.0	75.8	16.9	71.5

Table 3: Embedding initialization comparison between different optimization methods for German → English. Online systems use joint BPE with $50k$ merge operations, whereas batch systems use seperate word-based vocabularies. Word-by-word initialization is only used for the batch optimized system.

First considering the online optimization scenario, both random and monolingual initializations fail to produce proper results. This is due to the differing word distributions for source and target embeddings that are given as an input to the encoder and decoder modules. Once the embeddings are language-independent, the model is able to achieve much better values. This follows the same motivation as the adversarial feature proposed by Lample et al. (2017), where the authors argue that the decoder must be fed with language-independant inputs in order to function effectively. Freezing the embeddings during training is also detrimental to translation quality.

Examining the initialization with the batch optimization method results in similar behaviours for a cross-lingual initialization. Here the initialization has a slight, albeit significant, influence on the translation quality. This is due to the cross-lingual signal already being strongly present in the word-by-word initialization, replacing the prior information that one gets from the word embedding initialization. Random and monolingual initializations perform roughly the same, which shows again the problem with the differing representation distributions. Overall, the cross-lingual initialization performs best for both methods.

Recently, Lample et al. (2018) have noted that it is possible to share embeddings across languages and initialize them with monolingual word vectors. We leave this for future work.

5.3 Embedding Features

Considering the empirical results of the previous section, we focus on improving upon the integra-

	newstest2017		newstest2018	
	BLEU	TER	BLEU	TER
baseline	**14.9**	72.7	18.1	**67.1**
+ frozen emb.	14.0*	75.8*	16.9*	71.5*
+ gating	14.4*	**72.5**	17.6*	67.3
+ emb. WN	14.5*	73.4*	17.5*	68.4*
+ emb. WN	14.7	72.8	18.2	**67.1**

Table 4: Results for different embedding initialization on systems optimized with the online strategy for German → English. The baseline system uses batch optimization, cross-lingual embeddings and shared vocabularies. WN stands for weight normalization. * denotes a p-value of < 0.01 w.r.t. the baseline.

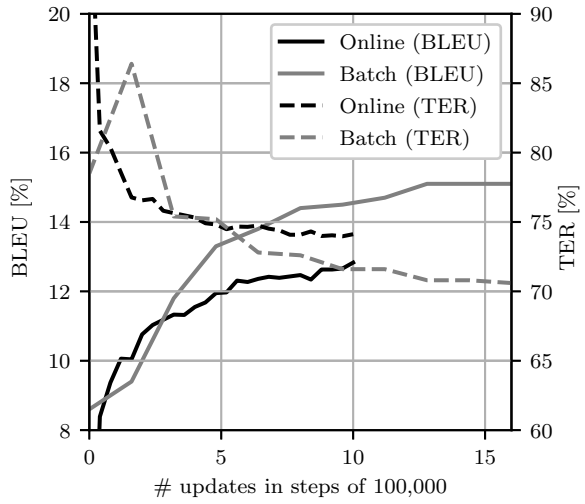

Figure 1: BLEU and TER values on newstest2017 German → English for checkpoint models of online and batch optimization methods. The initial step of the batch method uses the word-by-word translation scores.

tion of pre-trained embeddings on top of the best system, namely a word-based batch optimized system with cross-lingual embeddings and a shared output layer. Experiments are presented in Table 4.

As seen in Table 3, freezing embeddings during training worsens translation quality. One can conclude that the model learns task-specific information via the embedding component.

As a first step, we apply the gating mechanism as presented in Section 3.4 and observe an increase in performance of up to 0.7% BLEU . However, results show that the model performs up to 0.5% BLEU worse than the baseline and achieves roughly the same TER performance.

Afterwards, we apply weight normalization as presented in Section 3.5 on top of both trainable and frozen embeddings. When applied on top of frozen embeddings, the normalization helps, but still lags behind the baseline. Adding it on top of the baseline does not worsen, but also does not improve translation quality.

These experiments allow one to conclude that fine-tuning embeddings to the task at hand performs better than the implemented techniques. Alternative embedding features could be considered, as for example the works mentioned in Section 3.4.

5.4 Training Variations

In this Section, we consider additional experiments that do not fit in a specific category and present them in Table 5.

Firstly, we add an adversarial loss term as in (Lample et al., 2017) on top of a batch optimized model with cross-lingual embeddings and a shared output layer. We report that performance drops by up to 1.2% BLEU and we hypothesize that the feature does not integrate well in the Transformer architecture. Specifically, the encoder outputs of an LSTM (Hochreiter and Schmidhuber, 1997) are bounded between -1 and 1, whereas the Transformer encoder outputs can take on any real value. The effect of the feature was not reproducible in separate experiments with the setup described in the original publication.

Secondly, we separate both output layer and decoder components from the model to obtain a setting similar to the one in (Artetxe et al., 2017). Translation quality drops by up to 0.8% BLEU and 0.9% TER . Note that in Section 5.1, we already saw a drop of roughly the same amount when not sharing the output layer.

We investigate whether noisy input sentences and auto-encoding are necessary at later stages of the training. Hence, these features are disabled after the 3rd iteration. The improvements are not statistical significant, but at the very least the comparison shows that the model does not worsen from focusing solely on the translation task after its initial learning period. This is due to it already being able to generate decent translations after the first few iterations.

Finally, we train the batch and online methods for a larger number of iterations, see Table 6, reaching 19.2% BLEU with the batch method af-

	newstest2017		newstest2018	
	BLEU	TER	BLEU	TER
baseline	14.9	72.7	18.1	67.1
+ adversarial	13.9*	74.2*	16.9*	69.0*
+ unshared decoder	14.3*	73.3*	17.3*	68.0*
+ drop AE & noise	**15.2**	**72.6**	**18.3**	**66.9**

Table 5: Results for training variations on German → English. The baseline system uses batch optimization, cross-lingual embeddings and shared vocabularies. * denotes a p-value of < 0.01 w.r.t. the baseline.

	newstest2018			
	De → En		En → De	
	BLEU	TER	BLEU	TER
online method	15.4	70.8	12.0	79.5
$1M$ updates	**16.8**	**69.3**	**13.2**	**77.7**
batch method	18.1	67.1	14.0	77.0
10th iteration	**19.2**	**64.6**	**15.4**	**74.3**

Table 6: Results for longer training iterations for German ↔ English. The baseline system uses batch optimization, cross-lingual embeddings and shared vocabularies.

ter 10 iterations and 16.8% BLEU with the online method after $1M$ updates on newstest2017. The extended training for the online and batch method trained for 14 and 12 days respectively. Figure 1 shows the training of the models on newstest2017 German → English. The initial TER spike occurs because hypotheses are 13% longer than the ones of the word-by-word system. Considering the results of these experiments, one should look into better optimization algorithm tuning for the online method.

5.5 Final Submission

The model in the final submission, shown in Table 7, consists of a word-based model with separate vocabularies, trained with the batch optimization method, initialized with cross-lingual embeddings, applies embedding weight normalization and is trained with a learning rate of $3 \cdot 10^{-4}$. The ensemble system consists of 4 variations of the single-best model, varying in learning rate values ($3 \cdot 10^{-4} \rightarrow 10^{-4}$), feed-forward projection hidden sizes ($2048 \rightarrow 1024$) and monolingual, instead of cross-lingual, embedding initialization.

For reference, we include our supervised submission system for the German → English constrained task. As expected, there is a large performance gap between both our systems. The most crucial point of improvement in our submission is the amount of data used. We used a small amount of the available data, even smaller than for our supervised submission, since we noted that the models took a long time to converge as portrayed in Figure 1. We suggest to invest efforts into tuning of optimization algorithm hyperparameters and using more data.

6 Conclusion

The RWTH Aachen University has participated in the WMT 2018 German → English and English → German unsupervised news translation tasks. We focus on reproducing related work and infer empirically that the batch optimization method from (Lample et al., 2017) performs best on our constrained setting, i.e. $5M$ sentences for each language. An initialization with cross-lingual word embeddings performs best for both optimization strategies. Sharing vocabularies is important for a shared model architecture. BPE-based vocabularies outperform word-based ones with the online optimization method. The noise model and auto-encoding losses are not needed in later stages of training in batch optimization. Freezing the word embedding layer during training hurts. Simply initializing and training the embeddings performs better than performing weight normalization or applying a gating mechanism on top of a frozen embedding layer.

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

	German → English						English → German					
	newstest2016		**newstest2017**		**newstest2018**		**newstest2016**		**newstest2017**		**newstest2018**	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
Single-best	17.2	68.7	14.5	72.9	18.1	66.9	13.7	77.0	11.2	82.0	14.5	75.8
Ensemble of 4	17.6	68.3	14.9	72.1	18.5	67.0	14.1	76.4	11.5	81.6	15.0	74.7
WMT 2018 Supervised submission	46.0	41.0	39.9	47.6	48.4	38.1	-	-	-	-	-	-

Table 7: Submission systems for the WMT 2018 German ↔ English news translation task.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*. Version 2.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, pages 1–14, San Diego, California, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*, volume 2, pages 176–181.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*. Version 3.

Shuoyang Ding and Kevin Duh. 2018. How do source-side monolingual word embeddings impact neural machine translation? *arXiv preprint arXiv:1806.01515*. Version 2.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

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 Association for Computational Linguistics*, pages 177–180.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043. Version 2.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*. Version 1.

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*, pages 311–318.

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

Tim Salimans and Diederik P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909.

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)*, 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 Association for Machine Translation in the Americas*, volume 200.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The 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 6000–6010.

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.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. *arXiv preprint arXiv:1804.09057*. Version 1.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*. Version 2.