Recycling a Pre-trained BERT Encoder for Neural Machine Translation

Kenji Imamura and **Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
{kenji.imamura,eiichiro.sumita}@nict.go.jp

Abstract

In this paper, a pre-trained Bidirectional Encoder Representations from Transformers (BERT) model is applied to Transformer-based neural machine translation (NMT).

In contrast to monolingual tasks, the number of unlearned model parameters in an NMT decoder is as huge as the number of learned parameters in the BERT model. To train all the models appropriately, we employ two-stage optimization, which first trains only the unlearned parameters by freezing the BERT model, and then fine-tunes all the sub-models.

In our experiments, stable two-stage optimization was achieved, in contrast the BLEU scores of direct fine-tuning were extremely low. Consequently, the BLEU scores of the proposed method were better than those of the Transformer base model and the same model without pre-training. Additionally, we confirmed that NMT with the BERT encoder is more effective in low-resource settings.

1 Introduction

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is a language representation model trained in advance on a very large monolingual dataset. We adapt this model to our own tasks after fine-tuning (Freitag and Al-Onaizan, 2016; Servan et al., 2016) using task-specific data. Systems using BERT have achieved high accuracy in various tasks, such as the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) and the reading comprehension benchmark using the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2018). However, most tasks using BERT are monolingual because it was originally developed for natural language understanding.

Recently, models in which the ideas of BERT are extended to multiple languages have been proposed (Lample and Conneau, 2019). These models, which are pre-trained using multilingual data, are called cross-lingual language models (XLMs). We can construct a machine translation system using two XLM models as the encoder and decoder.

In this paper, we apply a pre-trained BERT encoder to neural machine translation (NMT) based on the Transformer (Vaswani et al., 2017). Specifically, the encoder of the Transformer NMT is replaced with the BERT encoder. Generally, systems using BERT, including machine translation systems based on an XLM, are fine-tuned using task-specific data. However, stable training is difficult using simple fine-tuning because the number of unlearned parameters is huge in our system. Therefore, we employ two-stage training, which first trains only unlearned parameters and then applies fine-tuning.

Our experimental results demonstrated that two-stage optimization stabilized training, whereas direct fine-tuning made the BLEU scores quite low. As a result, the BLEU scores improved in comparison with the scores of the Transformer base model and models with the same structure but without pre-training. Our results indicate that we can reuse neural networks trained for one purpose (natural language understanding, in this case) for another purpose (machine translation, in this case) if we use two-stage optimization. From this viewpoint, this paper presents an example of network recycling.

The remainder of this paper is organized as follows. In Section 2, an overview of the BERT related models is provided. In Section 3, our proposal, that is, NMT using the BERT model and its training method, is described. In Section 4, the proposed method is evaluated through experiments. In Section 5, we discuss back-translation and model recycling, including related work. Finally, we conclude the paper in Section 6.

2 Pre-trained Language Models

2.1 BERT

The form of the BERT model (Devlin et al., 2019) is the same as that of a Transformer encoder, in which the input is a word sequence and the output consists of representations that correspond to the input words. The input contexts are encoded by multi-head self-attention mechanisms.

BERT models are distributed with pre-training. The distributed models have a depth of 12 or 24 layers, which is deeper than the Transformer base model (six layers). Users (or system developers) construct various systems by adding a small network to adapt BERT to their own tasks and fine-tune the system using task-specific data. For example, when the BERT model is used for a document classification task, a classifier is constructed by adding a generation layer for classification (which consists of linear and softmax sublayers) to the BERT model. Similarly, when a named entity recognizer is constructed, generation layers that convert word representations to named entity tags are added, and the entire model is fine-tuned. The numbers of additional parameters in these models are much smaller than the number of parameters in the BERT model.

The BERT model is pre-trained to perform two tasks: masked language modeling and next-sentence prediction. Both tasks train the model to improve its language prediction performance.

The masked language model is trained to restore the original word sequence from noisy input. In this task, some words are replaced with special tokens, `[MASK]`, or other words. For instance, if the original sentence is "`my dog is hairy`" and "`my dog is [MASK]`" is given as the input word sequence, then the system predicts that the original word for `[MASK]` was `hairy`. During prediction, both forward and backward contexts in a sentence are used.

In the next-sentence prediction task, the system learns whether two given sentences are consecutive. To implement binary classification using the Transformer, a special token `[CLS]` is placed at the head of an input sentence, and classification is performed from the representation of `[CLS]`. Additionally, `[SEP]` is used as a sentence separator.

BERT has achieved high accuracy on various tasks, such as the GLUE benchmark (Wang et al., 2019) and the reading comprehension benchmark using SQuAD (Rajpurkar et al., 2018). However, the above tasks are monolingual.

2.2 XLMs

XLMs, in which the ideas of BERT are extended to multiple languages (Lample and Conneau, 2019) have been proposed. Although the form of the XLM model is also Transformer, it is trained from multilingual corpora. It also learns bilingual correspondences in a Transformer model by inputting a connected bilingual sentence.

Machine translation can be realized using XLMs by regarding two pre-trained models as an encoder and decoder. NMT using BERT described in this paper is fundamentally the same as XLM-based NMT. However, our aim is to connect different systems, and we regard our approach as model recycling (Ramachandran et al., 2016) using the BERT encoder and Transformer decoder.

Most pre-trained systems, including XLM-based machine translation, are trained only using fine-tuning (Devlin et al., 2019; Lample and Conneau, 2019). However, if the number of unlearned parameters is huge compared with the number of pre-trained parameters, then the pre-trained parameter values will be destroyed due to a phenomenon called *catastrophic forgetting* (Goodfellow et al., 2013), and consequently, training will diverge. We must suppress this problem to stably train the model.

3 NMT with BERT

In this section, we describe our proposal: NMT using the BERT encoder.

3.1 Model

The NMT system in this paper is an encoder-decoder based on the Transformer. The structure is shown in Figure 1. Because the BERT model is also the Transformer encoder, we adopt it as the encoder for NMT without modification. The outputs from the BERT encoder, which are representations of source words, are input to the context attention mechanism in the Transformer decoder to generate a translation. Note that we call the encoder of the conventional NMT the Transformer encoder to distinguish it from the BERT encoder. The number of layers in the Transformer decoder is fixed to six throughout the paper.

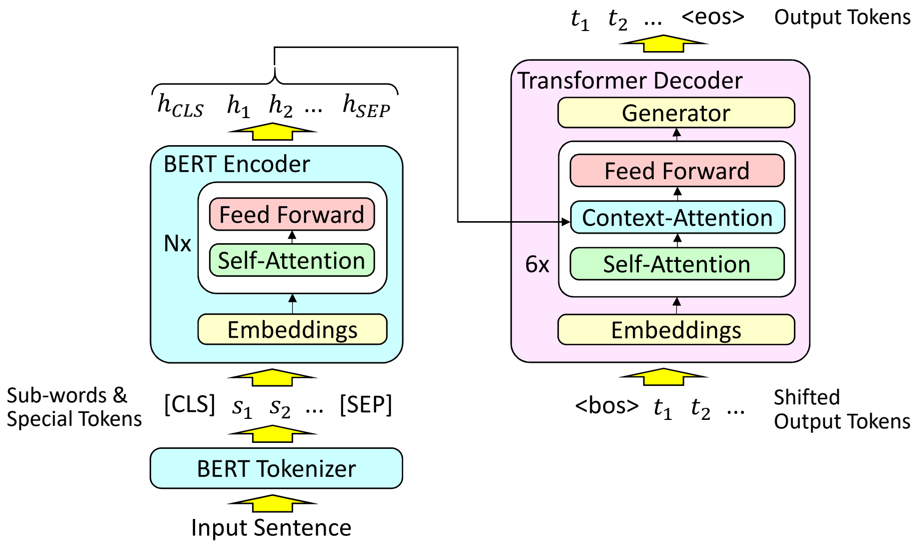

Figure 1: Structure of NMT using a BERT encoder.

Because the BERT encoder is pre-trained using a specific tokenizer, we must use the BERT tokenizer when we apply it to machine translation. The BERT tokenizer includes sub-word segmentation based on WordPiece (Wu et al., 2016). Additionally, the input to the encoder contains `[CLS]` and `[SEP]` tokens at the head and tail, respectively.

Although the Transformer decoder is used here, we assume that a recurrent neural network decoder (Sutskever et al., 2014; Bahdanau et al., 2014) can also be applied,.

3.2 Training

When we construct a system using pre-trained BERT models, we generally add a task-specific network and fine-tune the system using task-specific data. Because the additional network is small (i.e., the number of additional parameters is small), the models can be learned by fine-tuning.

In this paper, the additional network is the decoder, whose number of parameters is huge. Indeed, the number of model parameters of the BERT encoder (pre-trained) and Transformer decoder (unlearned) used in Section 4 were 110M and 80M, respectively. We cannot train such a large number of unlearned parameters only by fine-tuning. Thus, we employ two-stage optimization in this paper, that is decoder training and fine-tuning.

3.2.1 Decoder Training

In this stage, only the decoder parameters are updated while the encoder parameters are frozen.

Specifically, the model is trained using parallel corpora, as in conventional NMTs. During training, however, backpropagation is only applied to the decoder and is dropped before the encoder. This means that we do not need to compute the gradients and maintain the input/output values in the BERT encoder, and we can train the model using GPUs with a relatively small memory even though the model size is huge. Dropout is only applied to the Transformer decoder, that is, it is not applied to the BERT encoder.

Note that decoder training continues until the loss of the development set is minimized. We discuss the efficient number of epochs for the decoder training in Section 4.3 when we combine it with fine-tuning.

3.2.2 Fine-Tuning

In the fine-tuning stage, model parameters in the BERT encoder are also updated to optimize the entire model.

Although the model has already been trained until the loss of the development set is minimized during decoder training, both the encoder and decoder are further optimized in an end-to-end manner by updating the encoder parameters. Dropout is applied to both the BERT encoder and Transformer decoder. In the fine-tuning stage, backpropagation is applied to all layers, and a large amount of memory is consumed.

Task	Set	# Sentences	# Tokens (En)
WMT-2014 En-De	train	4,468,840	116M
	newstest2013	3,000	66K
	newstest2014	2,737	63K
	newstest2015	2,169	68K
IWSLT-2015 En-Vi	train	133,317	2.7M
	tst2012	1,553	34K
	tst2013	1,268	34K

Table 1: Sizes of the experimental datasets.

4 Experiments

4.1 Experimental Settings

Data: In this paper, we used data for shared tasks that were pre-processed by the Stanford NLP Group.[1] The sizes of the datasets are shown in Table 1.

The first task is the news translation task of English-to-German (En-De) from the Workshop on Statistical Machine Translation (WMT-2014) (Bojar et al., 2014). Four sets in total have been published. In this paper, we used `newstest2013` as the development set, and the other two sets (`newstest2014` and `2015`) were used as test sets.

Moreover, to contrast the results with those of a low-resource setting, we also used the English-to-Vietnamese (En-Vi) corpus used in the International Workshop on Spoken Language Translation (IWSLT-2015) (Cettolo et al., 2015). We used `tst2012` as the development set and `tst2013` as the test set.

The target sentences of the above data were further segmented into 16K sub-words using byte-pair encoding (Sennrich et al., 2016b). For the source sentences, we segmented 30K sub-words using the BERT tokenizer if the encoder model was BERT. If the encoder was the conventional Transformer, we segmented the source sentences into 16K sub-words using our byte-pair encoder.

BERT Model: We used a pre-trained model published in the BERT GitHub repository[2] called the BERT base model, whose features are uncased inputs, 12 layers, 768 hidden units, and 12 heads.[3] The number of parameters is about 110M. This model was trained using BookCorpus (Zhu et al., 2015) and English Wikipedia (3.3G words in total).

Note that we converted the published model into a model that is compatible with the PyTorch library using a tool[4] because it was trained for the TensorFlow library.

Type	Option	Value
Decoder training	Batch size	Approx. 500 sents.
	Optimizer	Adam $\beta_1 = 0.9, \beta_2 = 0.99$
	Learning rate	4.0×10^{-4}
	Warm-up	5 epochs
	Cool-down	Inverse square root
	Label smoothing	0.1
	Dropout	0.15
	Loss function	Label-smoothed cross entropy
	Initializer	Xavier uniform
Fine-tuning	Identical to decoder training except for the learning rate and warm-up	
Test	Beam size	10
	Length penalty	1.0 (fixed)

Table 2: Hyperparameter settings.

Translation System: We used fairseq[5] as the basic translator. It is an NMT system constructed on the PyTorch library and includes Transformer models. We replaced its encoder with the BERT model and used the fairseq decoder without modification.

The decoder used here was six-layer Transformer. We set the numbers of hidden units and heads to be the same as those of the encoder (i.e., 768 units and 12 heads) to incorporate encoder outputs into the decoder using the context attention mechanism.

Hyperparameters: Table 2 summarizes the hyperparameter settings. The hyperparameters for fine-tuning were almost the same as those of decoder training, except for the learning rate (LR) and warm-up.

Evaluation: We used BLEU (Papineni et al., 2002) as the evaluation metric. The MultEval tool (Clark et al., 2011)[6] was used for statistical testing, which is based on the bootstrap resampling method. The significance level was 5% ($p < 0.05$).

[1]`https://nlp.stanford.edu/projects/nmt/`

[2]`https://github.com/google-research/bert`

[3]`https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip`

[4]`https://github.com/huggingface/pytorch-pretrained-BERT`

[5]`https://github.com/pytorch/fairseq`

[6]`https://github.com/jhclark/multeval`

	System	LR	Dev. PPL ↓	BLEU ↑ 2013	2014	2015	Remark
Baselines	Transformer base	4.0×10^{-4}	4.23	26.29	27.22	29.48	Stat. test baseline
	Transformer BERT size	4.0×10^{-4}	4.04	26.15	27.09	29.32	
NMT with BERT	Direct fine-tuning	8×10^{-5}	4.28	0.13 (-)	0.10 (-)	0.12 (-)	# Epochs = 33
		4.0×10^{-4}	4.09	0.48 (-)	0.42 (-)	0.54 (-)	# Epochs = 29
	Proposed: Decoder training only	4.0×10^{-4}	4.76	24.13 (-)	23.62 (-)	25.74 (-)	# Epochs = 65
	+ Fine-tuning	4×10^{-5}	3.93	**27.14** (+)	28.27 (+)	30.68 (+)	# Epochs = 21
		8×10^{-5}	**3.92**	27.05 (+)	**28.90** (+)	**30.89** (+)	# Epochs = 9
		1.2×10^{-4}	3.93	27.03 (+)	28.50 (+)	30.51 (+)	# Epochs = 11
		1.6×10^{-4}	3.94	26.64	28.59 (+)	30.51 (+)	# Epochs = 11
		2.0×10^{-4}	3.95	26.89 (+)	28.67 (+)	30.24 (+)	# Epochs = 12
		4.0×10^{-4}	N/A because training did not converge				

Table 3: Results of the system comparison.
The bold values indicate the best results. The (+) and (-) symbols indicate that the results significantly improved or degraded ($p < 0.05$) with respect to the Transformer base model, respectively.

4.2 System Comparison

In this section, we compare systems using WMT-2014 data.

We compare the proposed methods with Vaswani et al. (2017)'s Transformer base model (6 layers, 512 hidden units, and 8 heads) as the baseline. For reference, we also use another Transformer NMT called Transformer BERT size, in which the model structure of the encoder agrees with that of the BERT encoder (i.e., 12 layers, 768 hidden units, and 12 heads). This system is compared to determine whether the model capacity influences translation quality.

The decoders of the proposed systems were first trained until the loss of the development set was minimized (`newstest2013`), and then the systems were fine-tuned. The learning rates during fine-tuning were changed from 1/10 to 1 times those of the decoder training. The warm-up periods were also changed to be proportional to the learning rates. The results are shown in Table 3. In the table, Dev. PPL indicates the perplexity of the development set, and the years of the BLEU scores denote the results of `newstest2013`, `2014`, and `2015`, respectively.

First, we compare the baselines. Transformer BERT size yielded a better model than that of Transformer base because its perplexity was lower. However, the BLEU scores were slightly degraded (but not significantly). Increasing the number of parameters did not lead to better translation quality.

Next, we compare the NMT system with BERT and the Transformer base. When the entire model was directly fine-tuned, training converged but the BLEU scores dramatically decreased even if we used different learning rates. This implies that the models were broken.[7] Unlike monolingual tasks using the BERT model, it was difficult to directly apply fine-tuning for NMT using the pre-trained BERT encoder.

By contrast, in the decoder training-only model, namely the model immediately after decoder training, training was successfully finished. However, the development perplexity was higher and the BLEU scores were lower than those of the baseline. The entire model was not learned completely because of the data mismatch, which was that the pre-training and our training data were different.

After fine-tuning, by contrast, the perplexity decreased and the BLEU score increased. Compared with those of Transformer base, the BLEU scores significantly improved in almost all cases. Because the development perplexity decreased with respect to Transformer BERT size, we can say that these improvements resulted from learning performance rather than the number of model parameters.

We changed the learning rates from 4×10^{-5} to 4.0×10^{-4} in the fine-tuning. When the learning rate was 4.0×10^{-4} (the decoder training learning rate), we could not tune the model because there was no convergence. For the other learning rates, there were no large differences in per-

[7]Indeed, very long translations were generated due to frequent repetition in the case of the low learning rate (8×10^{-5}). We suppose that the decoder was not learned sufficiently. In the case of the high learning rate (4.0×10^{-4}), long and completely wrong translations were generated. We suppose that the encoder was broken.

Decoder training		Fine-tuning		BLEU ↑			
# Epochs	Dev. PPL ↓	# Epochs	Dev. PPL ↓	2013	2014	2105	Remark
0	—	33	4.28	0.13 (-)	0.10 (-)	0.12 (-)	Direct fine-tuning
1	33.05	34	4.23	26.67	28.77	30.16 (-)	
2	11.47	20	4.20	26.80	28.58	30.82	
3	8.12	18	4.10	**27.41**	28.93	30.78	
5	6.69	18	4.02	27.20	28.43 (-)	30.80	
10	5.50	15	3.96	27.33	28.59	30.42	
20	5.08	18	**3.91**	27.00	28.87	**30.92**	
30	4.89	10	3.92	27.18	28.39 (-)	30.70	
40	4.86	12	**3.91**	27.01	**29.04**	30.70	
50	4.81	11	**3.91**	27.02	28.65	30.77	
65	4.76	9	3.92	27.05	28.90	30.89	Decoder training converged Baseline for statistical testing

Table 4: Changes in the perplexity of the development set (Dev. PPL) and BLEU scores with respect to the number of decoder training epochs.

plexity or BLEU scores. However, we confirmed that there were some differences in convergence time (# Epochs), and fine-tuning converged in nine epochs when the learning rate was $8{\times}10^{-5}$. Therefore, we fixed the learning rate for fine-tuning to 8×10^{-5} in our subsequent experiments.

4.3 Number of Epochs for Decoder Training

The proposed method first performs decoder training until convergence, and then applies fine-tuning. However, this approach may not be optimal because decoder training is slow. For instance, it took 65 epochs for decoder training in the experiment in the previous section. In this section, we investigate whether decoder training can be made more efficient by stopping it before convergence.

Table 4 shows the results after fine-tuning when decoder training was stopped after various numbers of epochs. Fine-tuning was conducted until convergence in all cases. The table shows the changes in development perplexity and the BLEU scores, whose baselines are shown on the bottom line. Note that zero epochs of decoder training (the top line) mean that fine-tuning was directly applied without decoder training (the same as in Table 3).

As the number of epochs of decoder training decreased, fine-tuning required more epochs and the final perplexity increased. However, the BLEU scores were almost stable. Indeed, only three scores significantly decreased with respect to the baseline.

Because we assume that the optimal settings of decoder training depend on the data and hyperparameters, we cannot provide explicit criteria values in this paper. At the very least, our experimental results show the following conclusions.

- To shorten the total training time, it is best to perform decoder training for three epochs ($3 + 18 = 21$ epochs in total).
- To obtain the best model (i.e., the model with the minimum development perplexity), 20 epochs are sufficient for decoder training (which yields a Dev. PPL of 3.91).

4.4 Experiment on a Low-resource Language Pair

In this section, the effect of the BERT encoder on a low-resource setting is explored using IWSLT-2015 En-Vi data (133K sentences). All experimental settings, except for the corpus, were the same as those in Section 4.1. The results are shown in Table 5.

In the low-resource setting, the development perplexity decreased in comparison with the baseline when applying the BERT encoder and performing decoder training only. However, the BLEU score degraded, as in the large data setting (Section 4.2).

By contrast, when fine-tuning was applied, both the perplexity and BLEU scores largely improved with respect to the baseline. The BLEU score of `tst2013` improved by +3.45. Considering the score of `newstest2015` in the experiment in Table 3 was +1.41 for the same settings, these results show that the BERT encoder is more effective for improving translation quality in a low-resource setting.

5 Discussion and Related Work

5.1 Contrast with Back-Translation

Back-translation (Sennrich et al., 2016a) is a technique to improve translation quality using mono-

	System	LR	Dev. PPL ↓	BLEU ↑ 2012	BLEU ↑ 2013
Baseline	Transformer base	4.0×10^{-4}	11.54	24.03	26.12
NMT with BERT	Proposed: Decoder training only	4.0×10^{-4}	11.45	21.77 (-)	23.23 (-)
	+ Fine-tuning	8×10^{-5}	**8.98**	**26.77** (+)	**29.57** (+)

Table 5: Results of the IWSLT-2015 data.

lingual corpora. It translates monolingual corpora of the target language into the source language, generates pseudo-parallel sentences, and trains the source-to-target translator from a mixture of pseudo- and manually created parallel corpora. Because BERT encoders are trained using source monolingual corpora, they complement each other.

However, there are differences between BERT encoders and back-translation from the viewpoint of parallel corpus sizes. BERT encoders themselves do not need parallel corpora for training. They can be applied to low-resource language pairs for which large parallel corpora are difficult to obtain. However, huge monolingual corpora are necessary to train BERT encoders. Therefore, they are suitable for translation from resource-rich languages (e.g., English) to low-resource languages.

By contrast, back-translation requires a certain size of parallel corpora to translate back from the target to the source languages. This is because back-translated results are not confident if the parallel corpora are small (Edunov et al., 2018). Therefore, back-translation is suitable for translating middle-resource language pairs.

Note that unsupervised machine translation can be realized using the XLM described in Section 2.2 by connecting two autoencoders as an encoder-decoder. Those autoencoders are trained to encode source-target-source and target-source-target using monolingual corpora. Therefore, this approach can be regarded as including back-translation. Because back-translation was originally developed to enhance decoders, it is reasonable to incorporate it into pre-training.

5.2 NMT Using Source Monolingual Corpora

There are other methods that improve translation quality using source monolingual corpora.

Zhang and Zong (2016) proposed a multi-task learning method that learns a sentence reordering model from source language corpora and a translation model from parallel corpora. Cheng et al. (2016) proposed semi-supervised NMT that simultaneously trains a translation model and two autoencoders using parallel and source/target monolingual corpora. Both methods must use parallel and monolingual corpora during training.

Our method explicitly distinguishes training stages: 1) pre-training of a BERT encoder using monolingual corpora, 2) training of a decoder using parallel corpora, and 3) fine-tuning the entire model. This means that monolingual corpora are only necessary in the pre-training stage, and we can focus on this stage to obtain the advantages of large corpora.

5.3 Pre-Training versus Recycling

The BERT model used in this paper was designed and trained for natural language understanding, and machine translation is an unintended purpose. Therefore, we use the word "recycle."

We assume that pre-training and recycling are distinguished by the number of unlearned parameters. The numbers of model parameters in this paper were 110M for the BERT encoder and 80M for the Transformer decoder. We could not optimize them using fine-tuning alone. In this case, it is appropriate to call the model recycling, and two-stage optimization becomes effective. We believe that our study can be regarded as an example of model recycling.

6 Conclusions

In this paper, an NMT system that incorporates a BERT encoder was presented. We applied two-stage optimization, that is, decoder training and fine-tuning, because the number of unlearned parameters was as large as the number of pre-trained model parameters. Consequently, we constructed a higher quality NMT than that trained from given parallel data. Moreover, it was particularly effective in a low-resource setting. We also investigated the appropriate number of epochs for decoder training and confirmed that several to tens of epochs were sufficient.

There are some future directions for this study. First, various pre-trained models have been distributed, such as the BERT large model, multilingual BERT, and XLMs. Exploring the relationship between these models and translation quality is our future work. Applying the pre-trained models to various language pairs, from low- to high-resource language pairs, is also a curious direction. Regarding model recycling, we plan to combine heterogeneous models in future work.

Acknowledgments

This work was supported by the "Research and Development of Enhanced Multilingual and Multipurpose Speech Translation Systems" a program of the Ministry of Internal Affairs and Communications, Japan.

References

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

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Mauro Cettolo, Kam Moejies amd Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany.

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*, pages 176–181, Portland, Oregon, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

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 of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-2016, Volume 1: Long Papers)*, pages 86–96, 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)*, pages 1715–1725, Berlin, Germany.

Christophe Servan, Josep Maria Crego, and Jean Senellart. 2016. Domain specialization: a post-training domain adaptation for neural machine translation. *CoRR*, abs/1612.06141.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 1535–1545, Austin, Texas.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.