TencentFmRD Neural Machine Translation for WMT18

Bojie Hu[1], **Ambyer Han**[2], **Shen Huang**[1]
[1] Tencent Research, Beijing, China
[2] Natural Language Processing Lab, Northeastern University, China
{bojiehu, springhuang}@tencent.com, ambyerhan0301@outlook.com

Abstract

This paper describes the Neural Machine Translation (NMT) system of TencentFmRD for Chinese↔English news translation tasks of WMT 2018. Our systems are neural machine translation systems trained with our original system TenTrans. TenTrans is an improved NMT system based on Transformer self-attention mechanism. In addition to the basic settings of Transformer training, TenTrans uses multi-model fusion techniques, multiple features reranking, different segmentation models and joint learning. Finally, we adopt some data selection strategies to fine-tune the trained system and achieve a stable performance improvement. Our Chinese→English system achieved the second best BLEU scores and fourth best cased BLEU scores among all WMT18 submitted systems.

1 Introduction

End-to-end neural machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) based on self-attention mechanism (Vaswani et al., 2017), the Transformer, has become promising paradigm in field of machine translation academia and industry. Experiments show Transformer, which does not rely on any convolutional or recurrent networks, to be superior in translation performance while being more parallelizable and requiring significantly less time to train. The training part of this paper is an improvement on the tensor2tensor[1] open source project based on the Transformer architecture, and the inference part is completely original, and we called this system TenTrans. We participated in two directions of translation tasks: English→Chinese and Chinese→English.

We divide TenTrans system into three parts to introduce in this paper. First, we introduce how to train better translation model, that is, the training phase. Second, we describe how good models can generate better translation candidates, that is, the inference phase. Finally, we describe N-best rescoring phase, which ensures that translation results which are closer to the expression typically produced by users are chosen. Our experimental setup is based on recent promising techniques in NMT, including using Byte Pair Encoding (BPE) (Sennrich et al., 2016b) and mixed word/character segmentation rather than words as modeling units to achieve open-vocabulary translation (Luong and Manning, 2016), using back-translation (Sennrich et al., 2016a) method and joint training (Zhang et al., 2018) applied to make use of monolingual data to enhance training data. And we also improve the performance using an ensemble based on six variants of the same network, which are trained with different parameter settings.

In addition, we design multi-dimensional features for strategic integration to select the best candidate from n-best translation lists. Then we perform minimum error rate training (MERT) (Och, 2003) on validation set to give different features corresponding reasonable weights. And we process named entities, such as person name, location name and organization name into generalization types in order to improve the performance of unknown named entity translation (Wang et al., 2017). Finally, we adopt some data selection strategies (Li et al., 2016) to fine-tune the trained system and achieve a stable performance improvement.

Our Chinese→English system achieved the second best BLEU (Papineni et al., 2002) scores and fourth best cased BLEU scores among all WMT18 submitted systems. The remainder of this paper is organized as follows: Section 2 describes the system architecture of TenTrans. Section 3 states

[1] https://github.com/tensorflow/tensor2tensor

Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 414–421

https://doi.org/10.18653/v1/W18-6413

all experimental techniques used in WMT18 news translation tasks. Sections 4 shows designed features for reranking n-best lists. Section 5 shows experimental settings and results. Finally, we conclude in section 6.

2 System Architecture of TenTrans

In this work, TenTrans has the same overall architecture as the Transformer: that is, it uses stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The encoder and decoder both are composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers, multi-head self-attention mechanism and position-wise connected feed-forward network. We add a residual connection (He et al., 2016) around each of the two sub-layers, followed by layer normalization (Ba et al., 2016). The left part of training phase in Figure 1 describes the structure of the basic sub-layer in the encoder and decoder. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. In this work we employ $multi\_head = 16$ heads, that is, parallel attention layers.

For all our models, we adopt Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$) as optimizer. We use model hidden state dimension 1024, the same as input embedding dimension and output embedding dimension. We linearly increase the learning rate whose initial value is 0.1 in the first $warmup = 6000$ training steps, and then anneal with the same way as Transformer. We use synchronous mini-batch SGD (Dean et al., 2012) training with $batch\_size = 6144$ and data parallelism on 8 NVIDIA Tesla P40 GPUs. We clip the gradient norm to 1.0 (Pascanu et al., 2013). We apply residual dropout (Zaremba et al., 2014; Srivastava et al., 2014) with $P_{rd} = 0.3$ to avoid overfitting. In training, we don't just focus on the word with highest probability score, but let the likelihood calculation be smoother, so applying label smoothing (Szegedy et al., 2016) with $\epsilon_{ls} = 0.1$. All weight parameters are initialized according to uniform distribution in the interval $[-0.08, 0.08]$. We will early stop training (Sennrich et al., 2017) when there is no new maximum value of the validation BLEU for 10 consecutive save-points (saving every 10000 updates) and select the model with the highest BLEU score on the validation set.

We mainly optimize TenTrans system through three parts. First, through the first part of Figure 1, multiple models are trained, and then the data selection method is used to continue to fine tune the system. Then, through the second part of Figure 1, the combination of best multiple models is used to decode the monolingual corpus to generate pseudo-bilingual data, and then the pseudo-bilingual data is proportionally added to the training set to continue the training of the first stage, and these two phases are continuously iterated until convergence. Finally, the third stage, N-best rescoring phase, finds the best translation result among the translation candidates by designed multiple sets of features. In order to learn the corresponding weights of multiple sets of features, the optimization is carried out through minimum error rate training (MERT).

3 Experimental Techniques

This section mainly introduces the techniques used in training and inference phase.

3.1 Multi-model Fusion Technology

For multi-model fusion, we try three strategies:

Checkpoint ensembles (CE), refers to the last N checkpoints saved during a single model training, where N is set to 10. In addition, we add the best 10 models saved during the early stopping training.

Independent parameter ensembles (IPE), refers to firstly training N models with different initialization parameters, and then weighting the average probability distribution of multiple models when softmax layer is calculated. Here we set N to 6, and we make better models have relatively higher weights, and poorer models have relatively lower weights.

Independent model ensembles (IME). An independent model ensemble is a set of models, each of which has been assigned a weight. It is not necessary to perform calculating the probability distribution in the inference process. Our experimental results show that this method performs slightly lower than IPE method, but the advantage is that the decoding speed is the same as the single model decoding.

In this work, we use the checkpoint ensemble method to initially integrate each single model, and then use the independent parameter ensemble method to perform multi-model integration in the

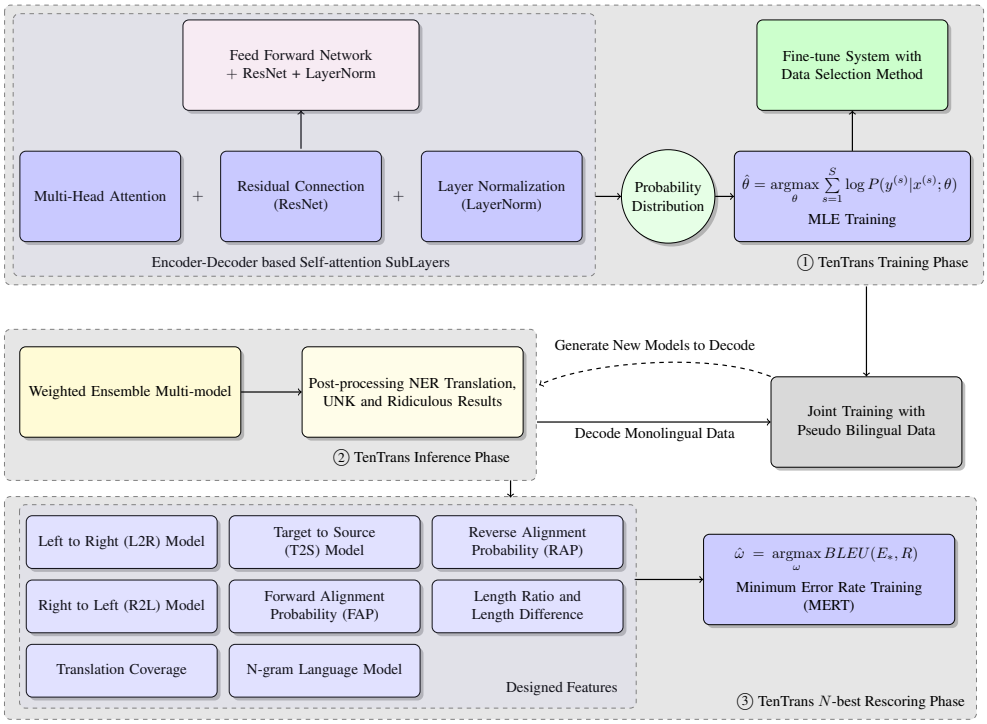

Figure 1: An illustration of system architecture of TenTrans. θ indicates model parameters being trained, and s indicates a training sample containing a source language x and a target language y. ω are the feature weight parameters being tuned by MERT. Ridiculous results mainly refer to translation results that are extremely long or short and clearly inconsistent with the source language.

stage of generating the final result of the system. The independent model ensemble method is used to decode monolingual corpus to generate pseudo-bilingual data during joint training.

3.2 Fine-tune System with Data Selection Method

In mainstream machine translation systems, network parameters are fixed after the training is finished. The same model will be applied to various test sentences. A very important problem with this approach is that it is difficult for a model to self-adapt to different sentences, especially when there is a big difference between the test set field and training set field. To alleviate this problem, (Li et al., 2016) proposed to search similar sentences in the training data using the test sentence as a query, and then fine tune the NMT model using the retrieved training sentences for translating the test sentence. We follow the strategy of (Li et al., 2016). This method firstly learns the general model from the entire training corpus. Then, for each test sentence, we extract a small subset from the training data, consisting of sentence pairs whose source sides are similar to the testing sentence. The subset is used to fine tune the general model and get a specific model for every sentence. To calculate similarity between two sentences, we adopt Levenshtein distance, which calculates the minimum steps for converting a string to another string using insertion, deletion and substitution these operations. We firstly filter the training corpus by only considering those which have common words with the testing sentence, and then compute similarity with the filtered set. In order to speed up the calculation, we use the inverted index method.

3.3 Joint Training

This work uses the monolingual corpus to enhance the training set by joint training. Joint training refers to the use of the corresponding additional target side and source side monolingual data at the source-to-target (S2T) and the target-to-source (T2S) translation model, and jointly optimizing the two translation models through an iterative process. In each iteration, T2S model is used to generate pseudo bilingual data for S2T with target-side monolingual data, and S2T model is used to generate pseudo bilingual data for T2S

Algorithm 1 Joint Training Algorithm in TenTrans System

Input: original bilingual data B, source monolingual data X_m, target monolingual data Y_m

Output: trained S2T models $M^i_{s2t}(i = 1 \cdots 6)$ and T2S models $M^i_{t2s}(i = 1 \cdots 6)$

Train 6 $M^i_{s2t}(i = 1 \cdots 6)$ and 6 $M^i_{t2s}(i = 1 \cdots 6)$ with different parameters

$n \Leftarrow 1$

while Not Converged **do**

  Integrate 6 $M^i_{s2t}(i = 1 \cdots 6)$ to generate M^{ens}_{s2t} with IME method

  Integrate 6 $M^i_{t2s}(i = 1 \cdots 6)$ to generate M^{ens}_{t2s} with IME method

  Use M^{ens}_{t2s} to generate pseudo-training data F_{t2s} by translating Y_m

  Use M^{ens}_{s2t} to generate pseudo-training data F_{s2t} by translating X_m

  New corpus to train $M^i_{s2t}(i = 1 \cdots 6)$, $C_{s2t} \Leftarrow n \times B + F_{t2s}$

  New corpus to train $M^i_{t2s}(i = 1 \cdots 6)$, $C_{t2s} \Leftarrow n \times B + F_{s2t}$

  $n \Leftarrow n + 1$

  Train M^i_{s2t} with $L(\theta_{s2t}) = \sum_{s=1}^{S} \log P(y^{(s)}|x^{(s)}) + \sum_{t=1}^{T} \log P(y^{(t)}|x^{(t)}) P'(x^{(t)}|y^{(t)})$ using C_{s2t} [2]

  Train M^i_{t2s} with $L(\theta_{t2s}) = \sum_{s=1}^{S} \log P(x^{(s)}|y^{(s)}) + \sum_{t=1}^{T} \log P(x^{(t)}|y^{(t)}) P'(y^{(t)}|x^{(t)})$ using C_{t2s} [2]

end while

with source-side monolingual data. This joint optimization approach enables the translation model in both directions to be improved, and generating better pseudo-training data to be added to the training set. Therefore, in the next iteration, it can train better T2S model and S2T model, so on and so forth. The right part of the decoding phase of Figure 1 outlines the iterative process of joint training. In addition, in order to solve the problem that back-translation often generates pseudo data with poor translation quality and thus affects model training, the generated training sentence pairs are weighted so that the negative impact of noisy translations can be minimized in joint training. Original bilingual sentence pairs are all weighted as 1, while the synthetic sentence pairs are weighted as the normalized corresponding model output probability. For the specific practice of joint training in this paper, see Algorithm 1.[2]

3.4 Different Modeling Units

We use BPE[3] with 50K operations in both source side and target side of Chinese→English translation. In English→Chinese translation task, we use BPE with 50K operations in English source side, and use mixed word/character segmentation in Chinese target side. We keep the most frequent 60K Chinese words and split other words into characters. In post-processing step, we simply remove all the spaces.

3.5 NER Generalization Method

To alleviate poor translation performance of named entities, we first use the pre-defined tags to replace named entities in training set to train a tagged NMT system, for example, use $\$number$ for numbers, $\$time$ for time, $\$date$ for date, $\$psn$ for person name, $\$loc$ for location name, $\$org$ for organization name. Then the key to the problem is how to identity these entities and classify them into corresponding types accurately. In order to solve this problem, we classify these entities into two types, one type that can be identified by rules, and the other type that can be identified by classification models. To decide whether an entity is a time, a number, or a date, we use finite automata (FA) (Thatcher and Wright, 1968). Aiming at the names of people, location names, and organization names, we first use biLSTM-CRF[4] (Lample et al., 2016; Huang et al., 2015) to train a Chinese sequence tagging model on "People's Daily 1998" data set and an English sequence tagging model on CoNLL2003 data set, and then identify named entities at the source and target language side of the training set.

[2] Here $P'(x^{(t)}|y^{(t)})$ refers to translation probability of M^{ens}_{t2s} translating monolingual sentence $y^{(t)}$ to generate $x^{(t)}$, $P'(y^{(t)}|x^{(t)})$ refers to translation probability of M^{ens}_{s2t} translating monolingual sentence $x^{(t)}$ to generate $y^{(t)}$, $P(y^{(s)}|x^{(s)})$ denotes translation probability of $x^{(s)} \rightarrow y^{(s)}$ during training S2T model, and $P(x^{(s)}|y^{(s)})$ denotes translation probability of $y^{(s)} \rightarrow x^{(s)}$ during training T2S model.

[3] https://github.com/rsennrich/subword-nmt

[4] https://github.com/guillaumegenthial/sequence_tagging

In the test phase, we first convert these entities in the test sentences into corresponding predefined tags, and then directly using the tagged NMT system to translate the sentences. When a tag is generated at target side, we select the corresponding translation of the word in the source language side that has the highest alignment probability based attention probability with the same as tag type in target side. If the source side does not have the same type of tag, delete the current tag directly. In order to obtain the corresponding translation of each entity vocabulary, we obtain it in the phrase extraction stage in statistical machine translation (SMT) (Koehn, 2009). We extract a phrase pair with one source word number from phrase table, and then use target side of the phrase pair with highest frequency of occurrence as the translation of the word to construct a bilingual translation dictionary. Although this method has not greatly improved the BLEU evaluation metric, it is of great benefit to the readability of the translation result for human. We use UNK to represent out-of-vocabulary (OOV) words, and translate it in the same way as above.

4 Experimental Features

This section focuses on the features designed to help choose translation results which are closer to the way normal user expressions - that is, it focuses on the N-best rescoring phase. Several features designed in this work can be seen in the left part of third phase in Figure 1.

4.1 Right to Left (R2L) Model

Since the current translation models all carry out modeling from left to right, there is a tendency for the prefix part of translation candidates to be of higher quality than the suffix part (Liu et al., 2016). In order to alleviate this problem of translation imbalance, we adopt a right-to-left translation modeling method. Two R2L modeling method are used in this work: the first is that only the target data is inversed, and the second is that both the target data and the source data are inversed. Then, two models, R2L model and R2L-both model were trained. Finally, we also reverse the n-best lists and calculate the likelihood probability of each translation candidate given the source sentence using these two models. Each model mentioned above is integrated by training 6 models with different parameters.

4.2 Target to Source Model

Neural machine translation models often have the phenomena of missing translation, repeated translation, and obvious translation deviation (Tu et al., 2017). In order to alleviate this problem, we use the target-to-source translation system to reconstruct the source-to-target translation results to the source sentence. This approach can make it very difficult to reproduce poor translation results to the source sentence, and the corresponding probability score will be low. Similarly, these models are all integrated by multiple models.

4.3 Alignment Probability

In order to express the degree of mutual translation between the translation candidate and source sentence at the lexical level, the lexical alignment probability feature is adopted. This paper uses two kinds of alignment probabilities, forward alignment probability and backward alignment probability. The forward alignment probability indicates the degree of alignment of source language vocabulary to the target language vocabulary, while the backward alignment probability indicates the degree of alignment of target language vocabulary to the source language vocabulary. We obtain the alignment score by *fast-align* toolkit[5] (Dyer et al., 2013).

4.4 Length Ratio and Length Difference

In order to reflect the length ratio between source sentence and translation candidates, we designed the length ratio feature $R_{len} = Len(source)/Len(candidate)$ and the length difference feature $D_{len} = Len(source) - Len(candidate)$.

4.5 Translation Coverage

To reflect whether words in the source language sentences have been translated, we introduce translation coverage feature. This feature is calculated by adding one to the feature value if the source language words has been translated. We use the *fast-align* toolkit to count the top 50 target words with highest probability of aligning each source language word as the translation set of this source word.

[5]https://github.com/clab/fast_align

System	C2E	E2C
baseline	23.32	33.06
+CE (checkpoint ensemble)	24.06	33.84
+IPE	25.98	35.58
+back-translation	26.49	36.0
+joint training	26.96	36.51
+fine-tune	27.63	37.29
+NER gereralization	27.74	37.43
+reranking (beam size 12)	29.72	39.03
+reranking (beam size 100)	30.13	39.49
submitted system	30.21	39.61

Table 1: Chinese↔English BLEU results on WMT18 validation set. The "C" and "E" denotes Chinese and English respectively.

4.6 N-gram Language Model

For English, the word-level 5-gram language model is trained on the mixing corpus of "News Crawl: articles from 2016" selected by newsdev2018 and English side of the training data. For Chinese, the character-level 5-gram language model is trained on the XMU. This work uses KenLM[6] toolkit (Heafield, 2011) to train n-gram language model.

4.7 Minimum Error Rate Training (MERT)

Obviously, some of the above features may be very powerful, while some of the effects are not particularly obvious. Therefore, we need to give each feature a corresponding weight. Our optimization goal is to find a set of feature weights that make the model score of translation candidates higher and the corresponding BLEU score higher. Therefore, we use minimum error rate training method to learn the feature weights

$$\begin{aligned}\omega^* &= \underset{\omega}{\operatorname{argmin}}\, Error(E_*, R) \\ &= \underset{\omega}{\operatorname{argmin}}\, (1 - BLEU(E_*, R)) \\ &= \underset{\omega}{\operatorname{argmax}}\, BLEU(E_*, R)\end{aligned} \quad (1)$$

where ω^* indicates tuned weights, E_* indicates the best translation candidate for the source language and R represents the corresponding reference translation.

[6]https://github.com/kpu/kenlm

5 Experimental Settings and Results

In all experiments, we report case-sensitive and detokenized BLEU using the NIST BLEU scorer. For Chinese output, we split to characters using the script supplied for WMT18 before running BLEU. In training and decoding phase, the Chinese sentences are segmented using NiuTrans (Xiao et al., 2012) Segmenter. For English sentences, we use the Moses (Koehn et al., 2007) tokenizer[7].

We used all the training data of WMT2018 Chinese↔English Translation tasks, firstly filtering out bilingual sentences with unrecognizable code, large length ratio difference, duplications and wrong language coding, then filtering out bilingual sentences with poor mutual translation rate by using *fast-align* toolkit. After data cleaning, 18.5 million sentence pairs remained. We used beam search with a beam size of 12, length penalty $\alpha = 0.8$ for Chinese→English system and length penalty $\alpha = 1.0$ for English→Chinese system. In order to recover the case information, we use Moses toolkit to train SMT-based recaser on English corpus. In addition, we also use some simple rules to restore the case information of the results. The size of the Chinese vocabulary and English vocabulary is 64k and 50k respectively after BPE operation. Table 1 shows the Chinese↔English translation results on development set of WMT2018. Wherein reranking refers to multi-feature based rescore method mentioned above. The submitted system in Table 1 has slightly better performance than is seen in the previous experiment because we have manually written some rules. As can be seen from the Table 1, when we increase the size of n-best from 12 to 100, the performance is improved by 0.41 BLEU after reranking based on multiple features.

6 Conclusion

In training phase of TenTrans, we report five experimental techniques. In the rescoring phase, we designed multiple features to ensure that candidates which are more likely to be produced by users are as close as possible to the top of n-best lists. Finally, our Chinese→English system achieved the second best BLEU scores among all WMT18 submitted systems.

[7]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

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

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.

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

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.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, San Diego, California, United States*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

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.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. 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 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

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. *arXiv preprint arXiv:1708.00726*.

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 (Volume 1: Long Papers)*.

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*.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

James W. Thatcher and Jesse B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103.

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

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017. Sogou neural machine translation systems for wmt17. In *Proceedings of the Second Conference on Machine Translation*, pages 410–415.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: an open source toolkit for phrase-based and syntax-based machine translation. In *Proceedings of the ACL 2012 System Demonstrations*, pages 19–24. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018. Joint training for neural machine translation models with monolingual data. *arXiv preprint*, arXiv:1803.00353.