Alibaba's Neural Machine Translation Systems for WMT18

Yongchao Deng*[*]* Shanbo Cheng[*] Jun Lu[*] Kai Song[*] Jingang Wang[*] Shenglan Wu[*] Liang Yao[*] Guchun Zhang[*] Haibo Zhang[*] Pei Zhang[*] Changfeng Zhu[*] Boxing Chen

Machine Intelligence Technology Lab, Alibaba Group
{yongchao.dyc,shanbo.csb,jeolu.luj,songkai.sk,jingang.wjg,shenglan.wsl, yaoliang.yl,guchun.zgc,zhanhui.zhb,xiaoyi.zp,changfeng.zcf, boxing.cbx}@alibaba-inc.com

Abstract

This paper describes the submission systems of Alibaba for WMT18 shared news translation task. We participated in 5 translation directions including English $\leftrightarrow$ Russian, English $\leftrightarrow$ Turkish in both directions and English $\rightarrow$ Chinese. Our systems are based on Google's Transformer model architecture, into which we integrated the most recent features from the academic research. We also employed most techniques that have been proven effective during the past WMT years, such as BPE, back translation, data selection, model ensembling and reranking, at industrial scale. For some morphologically-rich languages, we also incorporated linguistic knowledge into our neural network. For the translation tasks in which we have participated, our resulting systems achieved the best case sensitive BLEU score in all 5 directions. Notably, our English $\rightarrow$ Russian system outperformed the second reranked system by 5 BLEU score.

1 Introduction

We participated in the WMT18 shared news translation task in 3 different language pairs: English $\leftrightarrow$ Russian, English $\leftrightarrow$ Turkish and English $\rightarrow$ Chinese. English $\leftrightarrow$ Russian is a traditional WMT language pair possessing a large amount of bilingual training and development data. And especially this year, 16 million new translation units are available for the training. However for some more recent language pairs, the situation of bilingual resources is less promising: English $\leftrightarrow$ Turkish language pair only has 200 K bitexts and for English $\rightarrow$ Chinese, the amount of bilingual resources remains the same as last year. In the following sections of this article, We will see that the availability of bilingual resources can differentiate the performance of the final system. More precisely, more bilingual data means greater ability to interact and absorb target side monolingual knowledge through the process of back translation, as well as its ability to retrieve the pertinent in-domain data during the data selection process.

We share a very similar model architecture and training flow for different languages directions. Our models are based on the Google's Transformer architecture (Vaswani et al., 2017). In order to improve our single system's performance, we experiment with some latest research findings such as transformer with relative position attention (Shaw et al., 2018), weighted transformer (Ahmed et al., 2017) and neural suffix prediction for Russian (Song et al., 2018) which will be developed in the next section. We will also see that different well-known multi-system based techniques such as model ensembling and model reranking can still improve the performance of a very strong single system, even though we have to push further the limit in term of the number of models to employ as well as the methods to combine them together.

The paper is structured as follows: Section 2 will describe the novelties of our model architecture compared to the Google's standard Transformer framework, then we present a detailed overview of our system in Section 3, before giving the experimental settings and main results across languages in Section 4. Finally, Section 5 will draw a brief conclusion of our work for WMT18.

2 Model Features

We describe in this section three different architecture enhancements that we do to the standard Transformer architecture, two of them come from the latest research work on Transformer, the third one is from our internal research group. They all, to a certain extent, help improve the baseline model, but the improvement is not consistent

[*] Equal contribution

across all languages and it becomes progressively weaker, diluted in the combination of other techniques.

2.1 Transformer with Relative Position Attention

We use relative position representation in self-attention mechanism (Shaw et al., 2018) of both the encoder and decoder side for all systems. Originally, the Transformer only uses the absolute position information in the word embedding layer, lacking of position information in higher layers. Incorporating explicit relative position information in self-attention enables its propagation to the higher layers. And in contrast to the absolute position, it's invariant to the sentence length. We compared the translation results between whether using this feature or not, and found that with the relative position features, the model performs better in reordering. We also implement the relative position representation with fast decoding. Experiments showed that it lead to faster convergence and better performance.

2.2 Weighted Transformer

Motivation: The Transformer Model proposed by Vaswani et al. (2017), uses a self-attention mechanism to avoid recurrence and convolution in previously proposed models. The heads in the multi-head attention are independent of each other, Ahmed et al. (2017) improved this with a new mechanism, namely multi-branch attention. The latter adds a group of dynamic learned parameters to distinguish the importance of the heads.

Our Implementation: We implement the weighted transformer, with extra small improvements compared to the original implementation. We introduce the weighting mechanism to both encoder and the bottom layer of multi-head attention in decoder which does not accept encoder output states. The reason we do not add in the upper layer of multi-head attention is that it causes about 3 times slower of training speed.

2.3 Neural Suffix Prediction for Russian

For English to Russian task, we implement Song et al. (2018) 's work, namely neural suffix prediction, in our baseline system. Song et al. (2018) 's work takes a two-step approach for the decoder. Russian word sequence is split into stem sequence and suffix sequence. During the decoding time, stem is first generated at each decoding step, before suffix is predicted. Due to limited resource, we didn't strictly evaluate the actual improvement of this method, compared with the baseline Transformer architecture. We directly use it as our baseline system. For the following part of this paper, our English to Russian model is with neural suffix prediction by default. We use Byte Pair Encoding (BPE) (Sennrich et al., 2015) to get subword sequence of English side. For Russian side, BPE is applied on the stem sequence.

3 System Overview

3.1 Large-scale Back Translation

Adding synthetic data through the process of back translation (Sennrich et al., 2016) has become the paradigm when building state-of-the-art NMT systems. especially when a large amount of target-side in-domain data is available. For low-resource languages, the use of back-translated monolingual data is crucial as the target side lexicon coverage is often insufficient, it is the case for English $\leftrightarrow$ Turkish, with only 0.2M bilingual sentence pairs and Turkish being a very morphologically-rich language.

Considering the abundant volume of the monolingual data provided by the organizers and the costful process of back translation, we need to select among the entire monolingual data those of quality and being close to our domain of interest. We use the methods described in the data selection section (Subsection 3.2) to select this in-domain data from the large monolingual data.

Then comes the question of how many back translated data should be used. Our experiments showed that it's difficult to have an universal recipe for all languages across all tasks, we had to experimentally tune the amount of synthetic data to use according to the specific task, even for the two directions within the same language pair (See Table 1 for more details).

For different translation tasks, we use synthetic data ranging from 5 million to 70 million in combination with the provided parallel corpus to train the NMT system, resulting in an increase of +3 to +7 BLEU point over our baseline systems.

In order to understand the effectiveness of the large-scale back translation, we give a simple analysis using the example of English $\rightarrow$ Russian. A big Russian language model using 96 million monolingual data (All-96M-LM) is trained for this

	authentic	synthetic (critical)	synthetic (upper bound tested)
EN→ RU	8M	8M	24M
RU→ EN	30M	70M	85M
EN→ TR	0.2M	6M	14M
TR→ EN	0.2M	10M	15M
EN→ ZH	7.2M	3.98M	10M

Table 1: Synthetic data usage. **Authentic**: the amount of authentic parallel data after cleaning; **Synthetic (critical):** the maximal amount of synthetic data added to the parallel data with improvement; **Synthetic (upper bound tested)**: the maximal amount of synthetic data tested

	baseline translation (BLEU 36.66)	**translation with BT** (BLEU 38.94)	**reference translation**
All-96M-LM	206.35	203.07	197.95
NoUN-80M-LM	204.72	199.83	194.83

Table 2: Perplexity analysis of the effect of back translation with English → Russian examples

purpose, then 3 different translations of the newstest2017 test set are evaluated in term of perplexity over this language model, the results are shown in the Table 2. We can see that the translation produced by a model using back translation has a lower perplexity than the one without using it, and an increase of +2.3 BLEU is observed accordingly. This means that with the extra target side in-domain data, the model can learn to produce more fluent translation.

Similar observations can be obtained using a different language model (NoUN-80M-LM), we can notice that without the UN data, the same translations have lower perplexity, as the UN domain is a different domain than the news one, that's also in line with the BLEU score increase when training without the UN corpora in the English → Russian experiment results (See Subsection 4.2).

3.2 Fine-tuning with In-domain Data

Fine-tuning is a common method for domain adaption in NMT, which has proven effective for boosting the translation quality in a specific domain. Following Luong and Manning (2015), we first train a model on a large out-of-domain corpus and then continue a few epochs only on a small in-domain corpus. In our work, we try two different approaches to select the small in-domain corpus, namely, n-grams and binary classification.

N-grams: In order to acquire high-quality in-domain data, we exploit the algorithm detailed in Duh et al. (2013); Axelrod et al. (2011), which aims at selecting sentence pairs from large out-domain corpus that are similar to the target domain. In our experiment, the parallel bi-texts and monolingual back-translation corpus are used as out-domain corpus O. While all available newstest sets are regarded as in-domain corpus I. We first train tri-gram language models over the source and target side of the in-domain corpus, respectively (H_{I-src} and H_{I-tgt}). Then, build tri-gram language models of similar size over the random sample from the out-domain corpus (H_{O-src} and H_{O-tgt}). Based on this, each sentence pair s from O is scored by the bilingual cross-entropy difference $[H_{I-src(s)} - H_{O-src(s)}] + [H_{I-tgt(s)} - H_{O-tgt(s)}]$. Finally, we sort all sentence pairs and select top n (n = 100K) sentences pair with the lowest scores to fine-tune the parameters of neural network.

Binary Classification: Finding the sentence pairs that are similar to the in-domain corpora can also be viewed as a text categorization problem, albeit there are only two categories here, that is, in-domain (1) and out-domain (0).

With the development of word embedding (Mikolov et al., 2013), we are now able to convert textual content into numerical representation that bears much more information than the traditional ngram-based models can, such as positional, semantic and syntactical information. In most sentences, there are parts that carry strong domain information and are very useful in determining whether a particular sentence is in-domain or out-domain, while other parts are much more general and thus less useful. To extract such key domain information from a sentence, we can use convolutional neural network (CNN) with a softmax classifier sitting in the top layer.

We follow the footstep of Chen and Huang (2016) where the Semi-Supervised CNN (SS-CNN) domain adaption method was proposed. We use our own cloud-based word2vec to train word embeddings of 300 dimensions, using all available WMT18 bilingual and monolingual corpora for the constrained translation tasks and all the corpora that we have access to for the unconstrained tasks. Similar to Chen and Huang (2016), we also make full use of conText (Johnson and Zhang, 2015) as the CNN-based text classifier, which features a stack of two independent CNNs. The inputs to the first network, which is a simple convolution layer, are bag-of-words one-hot vectors, concatenated one-hot vectors, bag-of-words word embedding vectors and concatenated word embedding vectors, respectively, which results in four output regions correspondingly. The regions are then fed to the actual CNN classifier altogether that consists of one convolution layer, one nonlinear layer, one max pooling layer as well as a softmax. Without the loss of generality, we refer the full stack as one CNN classifier. For bilingual corpora, we train two classifiers, one for each language. Each classifier is trained with pre-trained word embeddings of each sentence and the corresponding label (1 for in-domain or 0 for out-domain). During inference, the classifiers will score each new sentence pair, resulting in four scores. That is, for each language, we will have one score for the in-domain possibility and the other for out-domain. Then, we replace the entropy scores of the scoring equation used in the ngram-based approach with these four possibility scores, to work out the final score for the sentence pair.

While training the CNN classifiers, we first sample a general domain corpora with the same number of sentence pairs in the in-domain set to be used as the out-domain set. For SSCNN, an in-domain set of a few thousand sentence pairs is sufficient to find high quality in-domain sub-corpora from the general corpora. Then, we label all in-domain pairs with 1 and all sampled pairs as 0. Next, for each language, we pass all labelled sentences to a CNN classifier, where the first network scans the input with the window size of 5 and the stride size at 1 with a zero padding of 4. For the second network, we employ 500 neurons with ReLU as the activation for the nonlinear layer. The loss function we use is mean square error and the training progresses using SGD with momentum.

Hyper-specialisation: While the two methods described previously in this section allow us to acquire data that are close to our development set, however, only suboptimal performance is expected on the final test set, as we don't have the reference translation to perform the bilingual data selection for the final test set. Inspired by the idea of hyper-specialisation (Deng et al., 2017), we produced multiple hypotheses of the test set using our best single and ensemble models, and used them as the target side translations. By integrating the real source text and target side translation pairs of the test set as in-domain seed into the data selection process, we makes the latter aware of the test set information, thereby enables it to retrieve better in-domain bi-texts for this specific test set. Subsequently, these synthetic bi-texts can serve as train data as they are in-domain parallel data of good quality, the idea is to imitate the effect of model ensemble, but at the data level.

Finally, we replace Adam (Kingma and Ba, 2014) optimizer with SGD and use the learning rate decay, then we continue training the current best model for a few more iterations on the mixture of synthetic bi-texts and top n (n=100K) selected bilingual texts.

3.3 Greedy Model Selection based Ensembling

Model ensembling is a widely used technique to boost the performance of a MT system, which consists in combining the prediction of multiple models at each decode step. However, we have observed that if the single models were strong enough, very tiny improvement could be drawn from a simple combination of the top N models. Also combining brutally an increasing number of models could easily go over the resource limit even with very powerful multi-gpu machines.

In order to overcome this limit, we adopted an approach named Greedy Model Selection based Ensembling (GMSE) that we will describe in this section.

GMSE Algorithm: The algorithm takes as input a sorted list of N strong single models $\mathscr{L}_{\text{cand}} = \{\mathscr{M}_{0 \leq i \leq N}\}$ with N could possibly up to several hundreds, the order is typically defined by the performance on the development set. The algorithm starts with a "keep" list $\mathscr{C}_{\text{current}}$ which initially only contains the model $\mathscr{M}_0$. At each iteration, a model candidate $\mathscr{M}_{\text{i}}$, is shifted from the input $\mathscr{L}_{\text{cand}}$ and concatenated temporarily to the current "keep" list, all these models are then put through a standard model ensemble process. If the current iteration ends up with a better BLEU score, the candidate model $\mathscr{M}_{\text{i}}$ is added to the "keep" list $\mathscr{C}_{\text{current}}$. Otherwise, it is add to a "redemption" list $\mathscr{R}$. and still has a weak chance to be "redeemed" for the future iterations. One model from the "redemption" list can only be redeemed once, after which it is withdrawn definitely from the candidates. At the beginning of each iteration, a candidate model $\mathscr{M}_{\text{i}}$ could be either drawn from the beginning of the $\mathscr{L}_{\text{cand}}$ with a probability P, the end of the $\mathscr{L}_{\text{cand}}$ with a probability $\text{P}_{\text{reverse}}$, or the "redemption" list $\mathscr{R}$ with a probability P_{redeem}, we used [P, $\text{P}_{\text{reverse}}$, P_{redeem}] = [0.8, 0.1,0.1] for our experiments. The algorithm ends when the input list $\mathscr{L}_{\text{cand}}$ is empty or a certain number of stalls (10) is reached. See algorithm 1 for the pseudo-code.

Algorithm 1 GMSE algorithm

```
Input:
    The sorted list of N single models ordered by performance on dev set: $\mathscr{L}_{\text{cand}} = \{\mathscr{M}_{0 \le i \le N}\}$;
    Number of stalls before stopping the algorithm: K;
Output:
    The best combination when stopping criterion is reached: $\mathscr{C}_{\text{best}}$;
 1: Initialization:
 2:     $\mathscr{C}_{\text{current}} = \{\mathscr{M}_0\}$
 3:     $\mathscr{C}_{\text{best}} = \{\mathscr{M}_0\}$
 4:     $\mathscr{R} = \{\}$
 5:     $S_{\text{best}} = S_{\text{M}_0}$
 6:     $k = 0$
 7: while $k < K$ and $\mathscr{L}_{\text{cand}}$ is not empty: do
 8:     if CONDITION($\mathscr{R}$, $\text{P}_{\text{redeem}}$) = True: then
 9:         $\mathscr{M}_{\text{cand}}$ = shift $\mathscr{R}$
10:     else if CONDITION($\text{P}_{\text{reverse}}$) = True: then
11:         $\mathscr{M}_{\text{cand}}$ = pop $\mathscr{L}_{\text{cand}}$
12:     else
13:         $\mathscr{M}_{\text{cand}}$ = shift $\mathscr{L}_{\text{cand}}$
14:     end if
15:     $\mathscr{C}_{\text{current}} = \mathscr{C}_{\text{current}} \cup \{\text{M}_{\text{cand}}\}$
16:     $S_{\text{current}} = ensemble(\text{C}_{\text{current}})$
17:     if $S_{\text{current}} > S_{\text{best}}$ then
18:         $\mathscr{C}_{\text{best}} = \mathscr{C}_{\text{current}}$
19:         $S_{\text{best}} = S_{\text{current}}$
20:     else
21:         $\mathscr{C}_{\text{current}} = \mathscr{C}_{\text{best}}$
22:         $\mathscr{R} = \{\mathscr{M}_{\text{cand}}\} \cup \text{R}$
23:     end if
24: end while
```

As mentioned at the beginning of the section, the effect of model ensemble is diminished with strong single models, especially with fine-tuned models. In order to boost the performance, we trained independently a large number of models using different model features for transformer models as described in the Section 2 , different hyperparameters, different versions of training data and different model types, resulting in a search space which is sufficiently large and with high diversity. The greedy nature of the GMSE algorithm makes the search feasible in a relatively acceptable time limit. On the development set, this algorithm can consistently improve more +1 BLEU point over the best single model across all the language directions in which we have participated. This increase drops to only around +0.3 - +0.5 on test set.

3.4 Greedy Feature Selection based Reranking

We describe the greedy feature selection based reranking (GFSR) we used in WMT 2018 in this section. N-best reranking in machine translation is a common-used technology, which can improve translation quality by picking better translations from n-best list to replace the one with the highest MT model score.

GFSR Framework: We adopted the widely used an open-source implementation in moses (Koehn et al., 2007) of K-batched MIRA algorithm (Cherry and Foster, 2012) to rerank the nbest list. Unlike most common reranking architectures, we select the features greedily from a large feature pool, in which there are about 50+ different feature types.

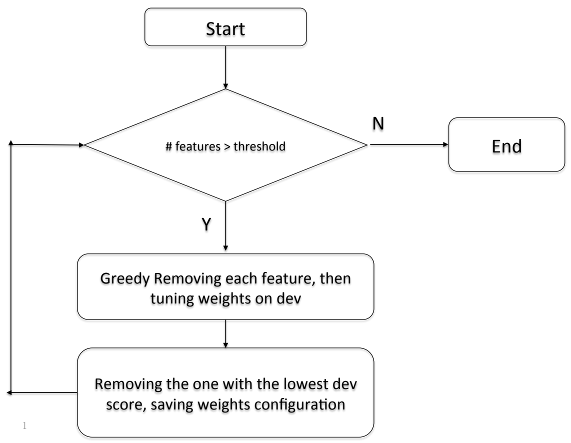
Start
# features > threshold
N
End
Y
Greedy Removing each feature, then tuning weights on dev
Removing the one with the lowest dev score, saving weights configuration

Figure 1: Framework of GFSR

As described in Figure 1, firstly, reranking the nbest list with all n features in the feature pool. Secondly, for all features, ignoring each one of them from the feature pool in a loop, and using the other $n - 1$ features to rerank on the dev data. Then, the feature that can get largest BLEU score improvement by ignoring it is removed from the

Category	Features
NMT Model Features	Main model score
	Left2Right sodel score (Liu et al., 2016)
	Target2source model score (Sennrich et al., 2016)
Language Model Features	Multiple ngram language models
Count Features	Word count
	Char count
	Word count ratio
	Char count ratio
Word-alignment-based Features	Word posterior probability (Ueffing and Ney, 2007)
	Sentence-level translation probability
Expected Scores	Consensus score (Expected BLEU) (DeNero et al., 2009)
	Expected ChrF (Popović, 2015)
	Expected Qmean (Chen et al., 2012)

Table 3: Features (feature templates) for reranking.

feature pools. The loop stops when the number of features is smaller than a threshold.

Features: We used about 50+ features in our reranking module, including NMT model features, count-based features, word-alignment-based features, expected scores features, etc. The feature types are described in Table 3. Some feature types such as NMT model features and Language Model features may have multiple instances.

Reinforced Nbest Generation: In order to have large beam size K = 100+ without introducing too many noises, we use multiple strong ensemble systems to generate a joint Nbest list. The idea is to have a higher upper-bound for the beam without the side-effect of having a lower lower-bound, Thereby, the reranker can focus on only good candidates.

3.5 Postprocessing

To recase (or recapitalize) the MT output, SMT-based recasers are trained on the Target side corpus with Moses toolkit[1]. In these models, language model plays an important role. As a result, large & domain related LMs are built. We also use a few simple uppercase rules, for example province & city names and the words beginning of a sentence are capitalized.

4 Experiments and Results

Preliminary experiments showed that the model features described in the section 2 yielded similar improvements reported in the original papers, or on par with the standard Transformer. For all of our baseline systems, we integrated these features into our model architecture, except the neural suffix prediction which is only used for the English $\rightarrow$ Russian system.

All of our experiments employ 6 encoder and decoder self-attention layers, both embedding and hidden size have a dimension of 512, 8 heads for the self-attention. We use FFN layer with 2048 cells and Swish (Ramachandran et al., 2017) as activation function. Warmup step is set to 16000 with a learning rate equals to 0.0003. We use label smoothing with a confidence score 0.9 and all the dropout probabilities are set to 0.1. All baseline systems are trained with 4 to 8 GPUs using synchronous-SGD with moving average mechanism where the average is taken in time and in space (Zhang et al., 2015).

We use BLEU as evaluation metric (Papineni et al., 2002). For English $\leftrightarrow$ Russian and English $\leftrightarrow$ Turkish, all reported scores are calculated over tokenized texts except for the 2018 submission which is end2end BLEU. For English $\rightarrow$ Chinese, all reported scores are end2end BLEU score using the SACREBLEU toolkit[2] (Post, 2018).

4.1 English $\rightarrow$ Chinese

For the English $\rightarrow$ Chinese system, we use all the available parallel data to train our English $\rightarrow$ Chinese system. The parallel corpus is firstly filtered using the same pipeline as for the other language pairs. As we find many sub-fragments belonging to the same translation units in the parallel data, we do an additional ngram-check based fuzzy filtering to get rid of these noisy pairs. We use an in-house tokenizer for both English and Chinese tokenization. After the preprocessing, we train BPE models with 60000 merge operations for both sides respectively.

To employ the monolingual Chinese corpus, we first build a ZH $\rightarrow$ EN Transformer system with all the available parallel data. We select the good quality in-domain corpus from the XMU monolingual corpus[3] to produce our synthetic data. The corpus contains a total number of $5,959,849$ sentences after the selection and a rule-based filtering. We set beam size as 12 and $alpha$ as 0.6 during batch-decoding. The generated synthetic data is augmented into our parallel training data to build our EN $\rightarrow$ ZH Transformer system. We extended the use of monolingual data to other sources, but it didn't result in better performance.

We follow the methods described in Subsection 3.2 for data selection. A series of models

[1]http://www.statmt.org/moses/

[2]https://github.com/awslabs/sockeye/tree/master/contrib/sacrebleu

[3]http://nlp.nju.edu.cn/cwmt-wmt/

System	newsdev2017	newstest2017
baseline	35.47	35.29
+ corpus cleaning	36.02	36.64
+ back translation	39.15	40.04
+ finetuning	40.06	40.68
+ ensemble	40.57	41.18
+ reranking	40.89	41.60
WMT18 submission	**43.37**	

Table 4: EN → ZH BLEU results on *newsdev2017* and *newstest2017*

can be obtained according to the methods and the amount of data used for fine-tuning. We adopt the GMSE approach for ensemble, the final best combination contains 7 models. Our reranker contains more than 70 features, including 14 Chinese language models, 8 Target-to-Source models, 4 Right-to-Left models. We use *newsdev2017* as the development set and *newstest2017* as the validation set during model training. The results of our system are reported in Table 4.1.

4.2 English ↔ Russian

For English ↔ Russian, we use the following resources from the WMT parallel data: News Commentary v13, CommonCrawl, ParaCrawl corpus, Yandex Corpus, UN Parallel CorpusV1.0 and Wiki Headlines. We perform data quality assessment, language identification, and excessive BPE segmentation filtering, resulting in a 28 million high-quality bilingual data. We train bidirectional systems using this high-quality bilingual data. We use 50000 BPE operations and the vocabulary size is set to 50000. For the English → Russian system, we found that it's beneficial to not make use of the UN corpora.

We selected in-domain monolingual data using the development sets 2012-2017 as seed data from the News Crawl corpora. We back-translated 24 million Russian and 70 million English sentences into the respective source side language using the the best single model trained on the high-quality bilingual data.

4.3 English ↔ Turkish

All parallel training data released are used in our TR ↔ EN systems, and it is about 207K sentences. We use an in-house tokenizer for both English and Turkish tokenization. A joint BPE model is applied in both directions, which is learned from mixed corpus of EN and TR with 16000 merge operations. As the parallel data amount is small, we use a shared vocabulary for both EN and TR, and we tie all embeddings of source, target and output layer following Press and Wolf (2017).

System	newstest2016	newstest2017
baseline		31.62
+ corpus cleaning		34.71
+ w/o UN	31.99	36.15
+ back translation	34.24	38.94
+ finetuning	34.96	40.37
+ ensemble	35.98	41.06
+ reranking	36.41	41.77
WMT18 submission	**34.8**	

Table 5: EN → RU BLEU results on *newstest2016* and *newstest2017*

System	newstest2016	newstest2017
baseline	29.98	33.56
+ corpus cleaning	30.82	36.33
+ back translation	33.90	39.84
+ finetuning	34.72	40.76
+ ensemble	35.76	41.34
+ reranking	36.23	41.97
WMT18 submission	**34.9**	

Table 6: RU → EN BLEU results on *newstest2016* and *newstest2017*

The back translation is particularly effective for EN ↔ TR as the amount of parallel data is very limited. For EN → TR, about 6 million sentences are selected from the newscrawl2016, 2017 and common crawl data, which is scored and sorted by domain similarity with newstest2016 test-set and authentic parallel data. Then, the 6 million sentences are translated into English by a TR ↔ EN model trained by the authentic parallel corpus. The domain relevance and the amount of data are important when using back-translation. The TR → EN follows the same procedure to get synthetic data, except the used monolingual data sources include news2014-2017 and news_comment, and the final amount of effective monolingual sentences is 10 million.

Unlike the back translation process, the fine-tuning is less effective as the amount of authentic parallel data is very limited. However, our data selection methods can still yield about +0.5 BLEU over strong underneath models.

System	newstest2016	newstest2017
baseline	14.28	14.97
+ joint-bpe	15.83	16.13
+ corpus-cleaning	16.31	16.80
+ back translation	22.92	23.87
+ finetuning	23.57	24.20
+ ensemble	24.63	24.96
+ reranking	25.23	25.76
WMT18 submission	**20.0**	

Table 7: EN $\rightarrow$ TR BLEU results on *newstest2016* and *newstest2017*

System	newstest2016	newstest2017
baseline	17.90	18.41
+ joint-bpe	18.33	18.72
+ corpus-cleaning	19.10	19.61
+ back translation	26.41	26.98
+ finetuning	27.21	27.52
+ ensemble	28.12	28.04
+ reranking	28.51	28.20
WMT18 submission	**28.0**	

Table 8: TR $\rightarrow$ EN BLEU results on *newstest2016* and *newstest2017*

5 Conclusion

This paper describes Alibaba's neural machine translation systems for the WMT18 shared news translation task. For all translation directions, we adopted the same strategies, which consist of building numerous strong single systems over which we employed reinforced multi-system based mechanisms to get the best out of all these single systems. We investigated the two mainstream methods to build a strong single system, one is based on incremental improvements of neural machine translation model architecture and the other is to have more data and make a better use of these data, and we found that the latter is more effective, at least in the cases where the former is not "revolutionary" enough. Finally, for all translation directions in which we have participated, we achieved the best results in term of case sensitive BLEU score, setting the new state-of-the-art performance.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *CoRR*, abs/1711.02132.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*, pages 355–362. Association for Computational Linguistics.

Boxing Chen and Fei Huang. 2016. Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data. In *Proceedings of The 2016 SIGNLL Conference on Computational Natural Language Learning*, pages 314–323. Association for Computational Linguistics.

Boxing Chen, Roland Kuhn, and Samuel Larkin. 2012. Port: a precision-order-recall mt evaluation metric for tuning. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 930–939. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436. Association for Computational Linguistics.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 567–575. Association for Computational Linguistics.

Yongchao Deng, Jungi Kim, Guillaume Klein, Catherine Kobus, Natalia Segal, Christophe Servan, Bo Wang, Dakun Zhang, Josep Maria Crego, and Jean Senellart. 2017. SYSTRAN purely neural MT engines for WMT2017. *CoRR*, abs/1709.03814.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 678–683.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 919–927. Curran Associates, Inc.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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.

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. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *EACL*.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions. *CoRR*, abs/1710.05941.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *CoRR*, abs/1803.02155.

Kai Song, Yue Zhang, Min Zhang, and Weihua Luo. 2018. Improved english to russian translation by neural suffix prediction. *CoRR*, abs/1801.03615.

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

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.

Sixin Zhang, Anna E Choromanska, and Yann LeCun. 2015. Deep learning with elastic averaging sgd. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 685–693. Curran Associates, Inc.