The JHU Machine Translation Systems for WMT 2018

Philipp Koehn **Kevin Duh** **Brian Thompson**
Johns Hopkins University
{phi,brian.thompson}@jhu.edu kevinduh@cs.jhu.edu

Abstract

We report on the efforts of the Johns Hopkins University to develop neural machine translation systems for the shared task for news translation organized around the Conference for Machine Translation (WMT) 2018. We developed systems for German–English, English–German, and Russian–English. Our novel contributions are iterative back-translation and fine-tuning on test sets from prior years.

1 Introduction

We carried out two relatively independent efforts on German–English language directions and Russian–English, using the Marian and Sockeye neural machine translation toolkits, respectively.

The German–English systems outperformed last year's best result (37.0 vs. 35.1 (+1.9) for German–English, 29.1 vs. 28.3 (+0.8) for English–German), but fell short against this year's best performing systems (45.3 vs. 48.4 (-3.1) and 43.4 vs. 48.3 (-4.9), respectively)[1]. The best models this year used the Transformer model instead of the recurrent neural networks that our models are based on. Our novel contributions are iterative back-translation and fine-tuning on prior test sets.

For Russian–English, we carried out extensive hyperparameter search, with different numbers of layers, embedding and hidden state sizes, and drop-out settings.

2 German–English and English–German

The systems for the German–English language pairs were developed with the Marian toolkit (Junczys-Dowmunt et al., 2018). We developed models with both shallow and deep architectures, based on recurrent neural networks. We ensembled 4 independent runs and reranked with right-to-left models (output in reverse order). We saw improvements with iterative back-translations and fine tuning on test sets from previous years, as well as use of the Paracrawl corpus (unfiltered).

A big challenge for system development are long training times (a month on a single GTX 1080ti GPU) which limited our ability to exploit the Paracrawl corpus. Because of this, we also started system development almost a year ago, using the training data from last year for the most part. All scores reported in this Section are on `newstest2017` with case-sensitive BLEU.

2.1 Shallow System Development

We started with shallow systems similar to Edinburgh's submission two years ago (Sennrich et al., 2016a). It uses byte pair encoding with a vocabulary of 50,000 (Sennrich et al., 2016c) and back-translation of the `news2016` monolingual corpus (Sennrich et al., 2016b), about twice the size of the original training data.

For each training run, we compare different ways to obtain a single best model.

- Use the single model that performed best on the dev set (newstest2016).
- Use checkpoint ensembling to obtain the 4 or 8 best models, and decode the test set with an ensemble of these models.
- Merge the models obtained by checkpoint ensembling into a single model.

For German–English, we achieved slightly better results with an ensemble of independent models rather than a merged model (about +0.2 BLEU), while for English–German they perform similarly. Ensembling of either kind clearly outperforms the single best model.

We then built ensembles of the resulting systems for the 4 independent runs. This gives gains

[1] Scores reported at `http://matrix.statmt.org/`

Shallow German–English

	single	ensemble		merged	
		4	8	4	8
run 1	31.8	32.3	32.5	32.2	32.3
run 2	32.4	32.6	32.8	32.5	32.7
run 3	32.8	32.8	32.7	32.5	32.5
run 4	32.2	32.9	32.9	32.7	32.7
ensemble	33.2			33.2	33.3
r2l rerank				33.7	

Deep German–English

	single	ensemble		merged	
		4	8	4	8
run 1	34.5	34.6	34.4	35.1	35.1
run 2	34.2	34.3	34.2	34.3	34.3
run 3	34.5	34.3	34.3	34.5	34.5
run 4	34.0	34.3	34.3	34.9	34.6
ensemble	34.9			35.6	35.6
r2l rerank				35.7	

Shallow English–German

	single	ensemble		merged	
		4	8	4	8
run 1	25.9	26.2	26.2	26.1	26.1
run 2	25.5	26.1	26.1	25.9	25.9
run 3	25.6	25.6	25.7	25.6	25.7
run 4	25.3	25.7	25.8	25.8	25.8
ensemble	26.4			26.4	26.5
r2l rerank				27.3	

Deep English–German

	single	ensemble		merged	
		4	8	4	8
run 1	27.7	28.0	28.1	27.8	27.9
run 2	27.7	27.8	27.7	27.8	27.7
run 3		28.0	27.8	27.9	27.8
run 4	26.1	27.5	27.8	27.8	27.9
ensemble	28.3			28.3	28.3
r2l rerank				28.9	

Table 1: Shallow and deep systems for German-English with Marian. 4 independent training runs, with checkpoint ensemble, and merging the checkpoint ensemble into a single model (averaging parameters). Ensemble of the runs, with right-to-left reranking (4 independent right-to-left runs).

of about +0.5 over the merged checkpoint ensembles. Notable, the ensemble over the single systems yields essentially the same quality.

The final improvement is right-to-left reranking (Liu et al., 2016) where we built also 4 independent systems on the data sets with the output word order reversed. This gave improvements of +0.5 for German–English and +0.9 for English–German. For detailed results, see Table 1.

2.2 Deep System Development

System development for deep models is essentially the same as for shallow models. We used the same data sets, also carried out 4 independent runs for each language direction, carried out checkpoint ensembling for each run, combined the resulting models in a ensemble and performed reranking with right-to-left models.

The models are similar to Edinburgh's submission from last year (Sennrich et al., 2017). They use 4 alternating encoder layers and 4 decoder layers, LSTM cells, dropout, layer normalization, tied embeddings, and Adam optimization.

Detailed results are also in Table 1. Merging the checkpoint models worked better for German–English, and about the same for English–German, compared to decoding with the multiple models. Ensembling the 4 independent runs yielded solid gains (about +0.5), but reranking helped substantially only for English–German (+0.6).

2.3 Iterative Backtranslation

The back-translated data was generated with a single shallow system trained on the parallel data. Since we obtained much better performance by using this back-translated data, employed deep model architecture and ensembled independent runs, we have now a much better system to back-translate data.

Note that this second round of backtranslation uses monolingual data in both languages. Starting with a German–English system (trained on parallel data), we translate monolingual German news text. We then use this synthetic parallel corpus to build a English–German system (in addition to the provided parallel data). We now use this English–German system to translate monolingual English text, yielding again a synthetic parallel corpus to be used in the final system.

We carried out the same system development

Iterative Deep German–English

	single	ensemble		merged	
		4	8	4	8
run 1	35.5	35.6	35.6	35.6	35.6
run 2	35.3	35.6	35.6	35.7	35.6
run 3	35.6	35.6	35.5	35.6	35.7
run 4	35.1	35.5	35.4	35.6	35.7
ensemble	36.1			36.1	36.1
r2l rerank				36.5	

Iterative Deep English–German

	single	ensemble		merged	
		4	8	4	8
run 1	28.5	28.5	28.5	28.5	28.5
run 2	28.1	28.2	28.3	28.3	28.3
run 3	27.8	28.1	28.3	28.3	28.4
run 4	28.6	28.5	28.6	28.7	28.4
ensemble	29.1			29.0	28.9
r2l rerank				29.4	

Table 2: System development with deep models using iterative back-translation.

German–English

	back	final
no back-translation	-	29.6
10k iterations	10.6	29.6 (+0.0)
100k iterations	21.0	31.1 (+1.5)
convergence	23.7	32.5 (+2.9)
re-back-translation	27.9	33.6 (+4.0)
+ deep ensemble		36.1 (+6.2)

English–German

	back	final
no back-translation	-	23.7
10k iterations	14.5	23.7 (+0.0)
100k iterations	26.2	25.2 (+1.5)
convergence	29.1	25.9 (+2.2)
re-back-translation	34.8	27.0 (+3.3)
+ deep ensemble		29.0 (+5.3)

Table 3: Impact of the quality of the back-translation system on the final system performace. Note that the back-translation systems run in the opposite direction and are not comparable to the numbers in the same row. The *deep ensemble* scores reported here match results in Table 2.

German–English

Setup	BLEU
iterative back runs 1–4	35.6–35.7
run with Paracrawl	36.3
ensemble	36.6
+ fine-tuned	37.0

Table 4: Final refinements: a model trained with the unfiltered Paracrawl corpus, an ensemble of the 4 iterative back-translation models, plus the

as for the shallow and deep models. See Table 2 for details. Table 3 shows how the quality of the back-translation impacts the final system's performance. This table is also reported in our paper on iterative back-translation (Hoang et al., 2018).

2.4 Use of Paracrawl Corpus

For German–English only, we added the Paracrawl corpus without any filtering to the training data used up to this point (parallel data plus iterative back-translated monolingual data). We only completed one training run, and obtained 36.3 BLEU (ensembled 4 checkpoints) as opposed to 35.6–35.7 for models without Paracrawl.

We trained this model for more than 2 months on a single GTX1080ti GPU. The best dev score (newstest2016) after 2 weeks was 43.0, after 4 weeks 43.3, after 6 weeks 43.7, and after 10 weeks 43.8. So, it seems to be necessary to train such a model for at least a month and a half.

Adding this model to the ensemble gives a score of 36.6. Weighting the Paracrawl model as much as all the 4 models without gives slightly higher score than equal weights (36.5 for equal weights).

2.5 Fine Tuning on Prior Test Sets

Finally, we fine tuned the model of one of the four iterative backtranslation runs towards the test sets from previous years. We trained for 3 epochs with a learning rate of 0.0003. Adding the resulting model to the ensemble gives an additional gain of +0.4, resulting in a final score of 37.0.

3 Russian–English

3.1 Data

We use the provided bitext for training neural machine translation systems (NMT) in a constrained setting. The bitext is first pre-processed via Joshua's[2] normalize.pl, followed by tokenize.pl and lowercase.pl. The training data is additionally filtered to sentences less than 80 tokens, result-

[2] http://joshua.incubator.apache.org/

ing in 37M sentence pairs (777M English tokens, 725M Russian tokens). We use newstest2016 (2998 sentence pairs) as the development set for early-stopping during NMT training. For continued training experiments, we further used a concatenation of newstests from 2012 to 2016 (14822 sentence pairs). We did not exploit any additional monolingual data, either by itself or via back-translation. After tokenization and lowercasing, the training data consists of 4.6M Russian and 3.7M English vocabulary types. We ran BPE[3]independently for each language, with 50K merge operations each.

All results in this section are reported on newstest2017 (3001 sentence pairs), which is treated as the initial test set. Unless otherwise specified, we report BLEU scores from multi-bleu.perl directly computed on lower-cased tokenized English reference.

3.2 Setup

In this task, we use Sockeye version 1.18.1[4] (Hieber et al., 2018) as our NMT engine. We explored a three-step approach to model building:

1. Hyperparameter search: First, we trained multiple NMT models using different hyperparameter settings (e.g. #layers, embedding size) on the 37M-sentence training bitext.
2. Continued training: Second, we attempted to improve the independent models in Step 1 via continued training on the newstest2012-2016 data, which more closely matches the test set in terms of domain.
3. Ensembles: Finally, we took the best models in Step 2 and performed ensemble decoding.

All our NMT systems above are sequence-to-sequence models using LSTM units. For training, we use the ADAM optimizer, with training set perplexity as the objective. The initial learning rate is set to 0.0003, and reduces by a factor of 0.5 after 3 checkpoints without improvement of development perplexity (”plateau-reduce” scheduler). The checkpoint is computed at a frequency of every 10k batches; with a batch size of 128 sentences, this corresponds to 1280k sentences, or 1/29th of the training data. After 8 checkpoints without improvements, the training is deemed to have converged. Most training runs converged between 30 to 100 checkpoints, which corresponds to 1 to 3 epochs over the training data. We then use the checkpoint with the best validation perplexity as the chosen model for each run. For decoding, we use beam search with the default beam size of 5.

3.3 Hyperparameter Search

We searched over four types of hyperparameters:

- The number of stacked LSTM layers in the encoder and decoder: **layer**=$\{1, 2, 3\}$
- The dimension of the word embeddings in source and target: **embed**=$\{500, 1000\}$
- The number of hidden units in each LSTM: **embed**=$\{500, 1000\}$
- The dropout rate for the embedding layer: **dropout**=$\{0.1, 0.3\}$

The goal is to quantify how sensitive the results are to hyperparameter settings, and to find the best model for submission. We train systems for a sample of 9 different hyperparameter settings from the $3 \times 2 \times 2 \times 2 = 24$ total combinations, and summarize their results in Table 5. For convenience of exposition, we label these models with id a-i.

Observation 1: We observe there is a *large variance* of test-bleu scores among models a-i, ranging from 31.1 for model a (best) to 27.3 for model i (worst). This suggests that hyperparameter search is very important for building strongly performing systems, even for settings that are not too different.

For example, compare the smallest model (e), which has 80M trainable weights, to the second smallest model (a), which has 85M trainable weights: the only difference between the two is one extra layer and 5M extra weights, yet the test-bleu changes from 29.5 (e) to 31.1 (a). Similarly, compare model c (137M weights) to model g (141M weights): they differ only in one extra layer, yet test-bleu varies as much as 30.1 (c) to 27.9 (g). The largest model (f), which has 200M trainable weights, ranks in the middle in terms of test-bleu among the 9 models.

While it may be tempting to extract ”suggested hyperparameter settings” from Table 5, we recommend a more robust strategy is to perform hyperparameter search to the extent possible.

Observation 2: We find that perplexity correlates well with bleu when ranking models in hyperparameter search. To a large extent, models a-d, which have the best training perplexities

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

[4] https://github.com/awslabs/sockeye

	Hyperparameter Setting				Results				
id	**layer**	**embed**	**hidden**	**dropout**	**test-bleu**	**step**	**train-ppl**	**dev-ppl**	**dev-bleu**
a	2	500	500	.1	31.1	91	5.20	9.13	27.7
b	2	1000	500	.3	30.3	61	5.53	9.48	26.8
c	2	1000	500	.1	30.1	58	5.37	9.39	27.3
d	1	1000	1000	.3	29.9	57	5.37	8.98	27.2
e	1	500	500	.1	29.5	72	5.68	10.36	26.2
f	3	1000	1000	.1	28.2	28	6.37	10.35	25.8
g	3	1000	500	.1	27.9	32	5.99	11.15	25.2
h	1	1000	500	.1	27.4	36	6.28	11.92	24.7
i	1	1000	500	.3	27.3	34	6.67	12.25	24.4
a’	2	500	500	.1	29.1	110	5.26	8.86	27.1
e’	1	500	500	.1	28.0	109	5.78	10.1	25.3

Table 5: Hyperparameter search results. The model with **id**=a is a sequence-to-sequence model with 2 **layer** of LSTMs in both the encoder and decoder, 500-dimensional source and target word embeddings (**embed**), 500 **hidden** units in each of the LSTM, and 0.1 **dropout** rate at the embedding layer. This model achieved 31.1 BLEU (**test-bleu**) on the test set (`newstest2017`) and comes from 91th **step** (or checkpoint) of the training run, which achieved a training set perplexity (**train-ppl**) of 5.20, a development set perplexity (**dev-ppl**) of 9.13, and a development set BLEU score (**dev-bleu**) of 27.7. All models with id a-j are trained on the BPE bitext with 50k merge operations in Russian and 50k merge operations in English, and are ranked in this table in terms of test-bleu. The last two rows represent additional experiments with model a’ and e’, which is similar to model a and e but are trained on BPE bitext with *30k* merge operations in Russian and 50k merge operations in English.

(**train-ppl**) and development perplexities (**dev-ppl**), also achieve the best BLEU scores (**dev-bleu**, **test-bleu**). This suggests that for hyperparameter search purposes, optimizing and validating based on perplexity is a sufficiently good surrogate for BLEU, which is expensive to compute.

Observation 3: The last two rows of Table 5 experiments with a different number of BPE operations for the source side (Russian). Models a’ and e’ are similar to models a and e, except that they use 30k merge operations rather than the 50k we used in all other experiments. The goal is to test the impact of subword units in hyperparameter search. The **train-ppl** and **dev-ppl** of these 30k models are better than or on-par with the 50k counterparts, but the BLEU scores appear to be worse. It is somewhat difficult to conclude with only these two datapoints, but we think that perhaps hyperparameter search with different subword units need to be conducted separately. Even on the same dataset, hyperparameters that work well in one version of the BPE data may not necessarily work well in another version of BPE.

Observation 4: It appears that the better models (a-e) seem to have trained longer; their final checkpoints are chosen at a relative high number of steps. For example, model (a) comes from checkpoint 91, which corresponds to 3 epochs over a 37M sentence dataset. In Figure 1, we plot the development BLEU for each of the training runs over time. Our models train for a maximum of 5 days; this is when the learning rate becomes minuscule and the training process determines convergence. We observe that BLEU continuously improves (while at a slower pace), even towards the end of the training process. This suggests that it might be possible to extract further BLEU gains by adjusting the learning rate and convergence criteria, encouraging the training to continue longer.

3.4 Continued Training

The training bitext comes from multiple domains, while the focus of the test set is news. One may treat this problem as domain adaptation. Here, we experiment with continued training[5] (Luong and Manning, 2015). The idea is:

Phase 1: Train a model until convergence on the multi-domain training bitext, as done in Sec 3.3.

Phase 2: Use the model weights from Phase 1

[5] Also called fine-tuning by some works.

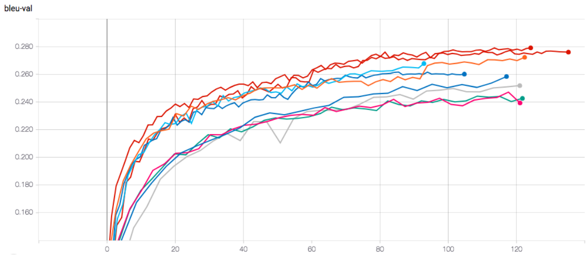

Figure 1: The 9 curves represent the change in BLEU scores when training each of models in Table 5. (y-axis: **dev-bleu**, x-axis: time in hours) Note that BLEU improves rapidly in the first 20 hours. The rate of improvement slows down but the improvement does not stop: BLEU continually improves even at hour 100 (4 days of training).

id	base	3 epochs	Δ	9 epochs	Δ
a	31.1	31.7	0.6	28.4	-2.7
b	30.3	31.5	1.2	27.8	-2.5
c	30.1	31.3	1.2	27.3	-2.8
d	29.9	31.3	1.4	27.8	-2.1
e	29.5	30.3	0.8	27.8	-1.7
g	27.9	29.2	1.3	25.9	-2.0
h	27.4	29.2	1.8	25.4	-2.0
i	27.3	29.2	1.9	26.1	-1.2

Table 6: Continued Training **test-bleu** on `newstest2017`. Base is the baseline number for each model from Table 5. The BLEU scores for continued training after 3 or 9 epochs are shown, along with their difference Δ against base. Continued training with few epochs improve results.

to initialize a new training process on adaptation data. This new training process usually only proceeds for a few steps. This is the Continued Training model, and can be used to decode the test set.

In Phase 2, we use `newstest2012-2016` as the adaptation training data. Part of it overlaps with the dev data (`newstest2016`), so the training procedure may constantly improve both train-ppl and dev-ppl, and never decide to converge. We therefore impose a hard-stop to prevent overfitting.

First, we experimented with stopping continued training after 3 epochs over the adaptation data. This corresponds to 350 batch updates (batch size is 128 sentences). ADAM is used as the optimizer, and the learning rate is fixed at a constant 0.0003.

The **test-bleu** scores are shown in Table 6. We observe that continued training is very effective, improving BLEU scores for all models by 0.6 to 1.9 points. For example, model (a) improves from 31.1 to 31.7, and model (b) improves from 30.3 to 31.5 on the `newstest2017` test set. However, if we train on adaptation data for too long, the results degrade. When continued training runs for 9 epochs (1150 batch updates), model (a) degrades to 28.4 The degradation is consistent for all models. This suggests that learning rate and amount of batch updates are important hyperparameters.

ensemble	test-bleu	Δ
a+b+c+d+e+f (6)	33.63	1.93
a+b+c+d+e (5)	33.57	1.87
a+b+c+d (4)	33.13	1.43
a+b+c (3)	33.13	1.43
a+b (2)	32.89	1.19

Table 7: Ensemble decoding **test-bleu** on `newstest2017`. We use the models from Table 6. The difference Δ is gain with respect to the best single model (a), with BLEU 31.7.

3.5 Ensembles

Finally, we performed ensemble decoding with the best continued training models obtained in the previous step. Table 7 shows a 6-model ensemble improves 1.93 BLEU over the best single model (a). This reaffirms the effectiveness of ensembles.

3.6 Final Russian–English Results

We submitted the 6-model ensemble in Table 7 as our final system in the official evaluation. As shown before, this model achieved 33.63 BLEU via multi-beu.perl on a tokenized and lowercased version of `newstest2017`. We also computed the official NIST-BLEU with the detokenized versions: it achieves 0.3195 (cased) and 0.3309 (lowercased) on `newstest2017`.

This result is only slightly improves upon our 2017 Moses SMT submission (Ding et al., 2017), which achieves 0.3129 (cased) and 0.3246 (lowercased) NIST-BLEU. We were interested in exploring the effectiveness of NMT under constrained data conditions (e.g. without backtranslation on large monolingual data) and standard sequence-to-sequence setups (e.g. withough reranking with left-to-right features or SMT/NMT hybrids). We imagine that these enhancements are needed if further gains are to be desired; unfortunately we may need to pay the cost of forgoing the simplicity of standard sequence-to-sequence NMT models.

References

Shuoyang Ding, Huda Khayrallah, Philipp Koehn, Matt Post, Gaurav Kumar, and Kevin Duh. 2017. The jhu machine translation systems for wmt 2017. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 276–282, Copenhagen, Denmark. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2018. The sockeye neural machine translation toolkit at amta 2018. In *Annual Meeting of the Association for Machine Translation in the Americas (AMTA)*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. 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, San Diego, California. Association for Computational Linguistics.

Minh-Thang Luong and Christopher Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 76–79.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural MT systems for wmt17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. 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)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. 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. Association for Computational Linguistics.