The JHU Machine Translation Systems for WMT 2017

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
The JHU Machine Translation Systems for WMT 2017

Shuoyang Ding† Huda Khayrallah† Philipp Koehn†
Matt Post‡ Gaurav Kumar† and Kevin Duh‡

†Center for Language and Speech Processing
‡Human Language Technology Center of Excellence

Johns Hopkins University
{dings, huda, phi}@jhu.edu,
{post, gkumar, kevinduh}@cs.jhu.edu

Abstract

This paper describes the Johns Hopkins University submissions to the shared translation task of EMNLP 2017 Second Conference on Machine Translation (WMT 2017). We set up phrase-based, syntax-based and/or neural machine translation systems for all 14 language pairs of this year’s evaluation campaign. We also performed neural rescoring of phrase-based systems for English-Turkish and English-Finnish.

1 Introduction

The JHU 2017 WMT submission consists of phrase-based systems, syntax-based systems and neural machine translation systems. In this paper we discuss features that we integrated into our system submissions. We also discuss lattice rescoring as a form of system combination of phrase-based and neural machine translation systems.

The JHU phrase-based translation systems for our participation in the WMT 2017 shared translation task are based on the open source Moses toolkit (Koehn et al., 2007) and strong baselines of our submission last year (Ding et al., 2016). The JHU neural machine translation systems were built with the Nematus (Sennrich et al., 2016c) and Marian (Junczys-Dowmunt et al., 2016) toolkits. Our lattice rescoring experiments are also based on a combination of these three toolkits.

2 Phrase-Based Model Baselines

Although the focus of research in machine translation has firmly moved onto neural machine translation, we still built traditional phrase-based statistical machine translation systems for all language pairs. These submissions also serve as a baseline of where neural machine translation systems stand with respect to the prior state of the art.

Our systems are very similar to the JHU systems from last year (Ding et al., 2016).

2.1 Configuration

We trained our systems with the following settings: a maximum sentence length of 80, grow-diag-final-and symmetrization of GIZA++ alignments, an interpolated Kneser-Ney smoothed 5-gram language model with KenLM (Heafield, 2011) used at runtime, hierarchical lexicalized reordering (Galley and Manning, 2008), a lexically-driven 5-gram operation sequence model (OSM) (Durrani et al., 2013) with 4 count-based supportive features, sparse domain indicator, phrase length, and count bin features (Blunsom and Osborne, 2008; Chiang et al., 2009), a distortion limit of 6, maximum phrase-length of 5, 100-best translation options, compact phrase table (Junczys-Dowmunt, 2012) minimum Bayes risk decoding (Kumar and Byrne, 2004), cube pruning (Huang and Chiang, 2007), with a stack-size of 1000 during tuning and 5000 during test and the no-reordering-over-punctuation heuristic (Koehn and Haddow, 2009). We optimize feature function weights with k-best MIRA (Cherry and Foster, 2012).

We used POS and morphological tags as additional factors in phrase translation models (Koehn and Hoang, 2007) for the German-English language pairs. We also trained target sequence models on the in-domain subset of the parallel corpus using Kneser-Ney smoothed 7-gram models. We used syntactic preordering (Collins et al., 2005) and compound splitting (Koehn and Knight, 2003) for the German-to-English systems. We did no language-specific processing for other languages.

We included Och cluster language model, with 4 additional language models trained on 50, 200,
Table 1: Tuning set sizes for phrase and syntax-based system

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>German–English</td>
<td>21,243</td>
</tr>
<tr>
<td>Czech–English</td>
<td>21,730</td>
</tr>
<tr>
<td>Finnish–English</td>
<td>2,870</td>
</tr>
<tr>
<td>Latvian–English</td>
<td>984</td>
</tr>
<tr>
<td>Russian-English</td>
<td>11,824</td>
</tr>
<tr>
<td>Turkish–English</td>
<td>1,001</td>
</tr>
<tr>
<td>Chinese–English</td>
<td>1,000</td>
</tr>
</tbody>
</table>

For English data, we used the scripts from Moses (Koehn et al., 2007) to tokenize our data, while for Chinese data we carried out word segmentation with Stanford word segmenter (Chang et al., 2008). We also normalized all the Chinese punctuations to their English counterparts to avoid disagreement across sentences. We parsed the tokenized data with Berkeley Parser (Petrov and Klein, 2007) using the pre-trained grammar provided with the toolkit, followed by right binarization of the parse. Finally, truecasing was performed on all the English texts. Due to the lack of casing system, we did not perform truecasing for any Chinese texts.

We performed word alignment with fast-align (Dyer et al., 2013) due to the huge scale of this year’s training data and grow-diag-final-and heuristic for alignment symmetrization. We used the GHKM rule extractor implemented in Moses to extract SCFG rules from the parallel corpus. We set the maximum number of nodes (except target words) in the rules ($\text{MaxNodes}$) to 30, maximum rule depth ($\text{MaxRuleDepth}$) to 7, and the number of non-part-of-speech, non-leaf constituent labels ($\text{MaxRuleSize}$) to 7. We also used count bin features for the rule scoring as our phrase-based systems (Blunsom and Osborne, 2008)(Chiang et al., 2009). We used the same language model and tuning settings as the phrase-based systems.

While BLEU score was used both for tuning and our development experiments, it is ambiguous when applied for Chinese outputs because Chinese does not have explicit word boundaries. For discriminative training and development tests, we evaluate the Chinese output against the automatically-segmented Chinese reference with multi-bleu.perl scripts in Moses (Koehn et al., 2007).

3.2 Results

Our development results on newsdev2017 are shown in Table 3. Similar to the phrase-based system, the syntax-based system is also outperformed by NMT systems for both translation directions.

4 Neural Machine Translation

We built and submitted neural machine translation systems for both Chinese-English and English-Chinese language pairs. These systems are trained...
with all the CWMT and UN parallel data provided for the evaluation campaign and newsdev2017 as the development set. For the back-translation experiments, we also included some monolingual data from new crawl 2016, which is back-translated with our basic neural machine translation system.

### 4.1 Preprocessing

We started by following the same preprocessing procedures for our syntax-based model baselines except that we didn’t do parsing for our training data for neural machine translation systems. After these procedures, we then applied Byte Pair Encoding (BPE) (Sennrich et al., 2016c) to reduce the vocabulary size in the training data. We set the number of BPE merging operations as 49500. The resulting vocabulary size for Chinese and English training data are 64126 and 35335, respectively.

### 4.2 Training

We trained our basic neural machine translation systems (labeled base in Table 3) with Nematus (Sennrich et al., 2017). We used batch size 80, vocabulary size of 50k, word dimension 500 and hidden dimension 1024. We performed dropout with dropout rate 0.2 for the input bi-directional encoding and the hidden layer, and 0.1 for the source and target word embedding. To avoid gradient explosion, gradient clipping constant 1.0 was used. We chose AdaDelta (Zeiler, 2012) as the optimization algorithm for training and used decay rate $\rho = 0.95$, $\varepsilon = 10^{-6}$.

We performed early stopping according to the validation error on the development set. The validation were carried out every 5000 batch updates. The early stopping was triggered if the validation error does not decrease for more than 10 validation runs, i.e. more than 50k batch updates.

### 4.3 Decoding and Postprocessing

To enable faster decoding for validation, test and back-translation experiments (in Section 4.4), we used the decoder from Marian (Junczys-Dowmunt et al., 2016) toolkit. For all the steps where decoding is involved, we set the beam size of RNN search to 12.

The postprocessing we performed for the final submission starts with merging BPE subwords and detokenization. We then performed de-trucasing for English output, while for Chinese output we re-normalized all the punctuations to their Chinese counterparts. Note that for fair comparison, we used the same evaluation methods for English-Chinese experiments as we did for the English-Chinese syntax-based system, which means we do not detokenize our Chinese output for our development results.

### 4.4 Enhancements: Back-translation, Right-to-left models, Ensembles

To investigate the effectiveness of incorporating monolingual information with back-translation (Sennrich et al., 2016b), we continued training on top of the base system to build another system (labeled back-trans below) that has some exposure to the monolingual data. Due to the time and hardware constraints, we only took a random sample of
Table 3: Chinese-English and English-Chinese System Development Results on newsdev2017 (cased BLEU scores). Bold scores indicate best and submitted systems.

<table>
<thead>
<tr>
<th>Language Pairs</th>
<th>Syntax</th>
<th>base single</th>
<th>base ensemble</th>
<th>back-trans single</th>
<th>back-trans ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese-English</td>
<td>16.22</td>
<td>17.81</td>
<td>18.46</td>
<td>17.52</td>
<td>18.16</td>
</tr>
<tr>
<td>English-Chinese</td>
<td>14.43</td>
<td>17.22</td>
<td>17.95</td>
<td>17.76</td>
<td>18.60</td>
</tr>
</tbody>
</table>

2 million sentences from news crawl 2016 monolingual corpus and 1.5 million sentences from preprocessed CWMT Chinese monolingual corpus from our syntax-based system run and back-translated them with our trained base system. These back-translated pseudo-parallel data were then mixed with an equal amount of random samples from real parallel training data and used as the data for continued training. All the hyperparameters used for the continued training are exactly the same as those in the initial training stage.

Following the effort of (Liu et al., 2016) and (Sennrich et al., 2016a), we also trained right-to-left (r2l) models with a random sample of 4 million sentence pairs for both translation directions of Chinese-English language pairs, in the hope that they could lead to better reordering on the target side. But they were not included in the final submission because they turned out to hurt the performance on development set. We conjecture that our r2l model is too weak compared to both base and back-trans models to yield good reordering hypotheses.

We performed model averaging over the 4-best models for both base and back-trans systems as our combined system. The 4-best models are selected among the model dumps performed every 10k batch updates in training, and we select the models that has the highest BLEU scores on the development set. The model averaging was performed with the `average.py` script in Marian (Junczys-Dowmunt et al., 2016).

4.5 Results

Results of our neural machine translation systems on newsdev2017 are also shown in Table 3. Both of our neural machine translation systems outperform their syntax-based counterparts by 2-4 BLEU points.

The results also indicate that the 4-best averaging ensemble uniformly performs better than single systems. However, the back-translation experiments for Chinese-English system do not improve performance. We hypothesize that the amount of our back-translated data is not sufficient to improve the model. Experiments with full-scale back-translated monolingual data are left for future work.

5 Rescoring

We use neural machine translation (NMT) systems to rescore the output of the phrase-based machine translation (PBMT) systems. We use two methods to do this, 500-best list rescoring, and lattice rescoring. Rescoring was performed on English-Turkish, and English-Finnish translation tasks. We combined the baseline PBMT models from Table 2, with basic NMT systems.

5.1 NMT Systems

We build basic NMT systems for this task. We preprocess the data by tokenizing, truecasing, and applying Byte Pair Encoding (Sennrich et al., 2015) with 49990 merge operations. We trained the NMT systems with Nematus (Sennrich et al., 2017) on the released training corpora. We used the following settings: batch size of 80, vocabulary size of 50000, word dimension 500, and hidden dimension 1000. We performed dropout with a rate of 0.2 for the input bi-directional encoding and the hidden layer, and 0.1 for the source and target word embedding. We used Adam as the optimizer (Kingma and Ba, 2014).

We performed early stopping according to the validation error on the development set. Validation was carried out every 20000 batch updates. The early stopping was triggered if the validation error does not decrease for more than 10 validation runs, if early stopping is not triggered, we run for a maximum of 50 epochs.

We create ensembles by averaging the 3 best validation models with the `average.py` script in Marian (Junczys-Dowmunt et al., 2016).
Table 4: Comparison of PBMT, NMT, NMT-Ensembles, and neural rescoring of PBMT output in the form of N-best lists or lattices (cased BLEU scores)

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>PBMT</th>
<th>NMT</th>
<th>NMT-Ens</th>
<th>N-best</th>
<th>Lattice</th>
<th>N-best</th>
<th>Lattice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>newstest2016</td>
<td></td>
<td></td>
<td>newstest2017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English-Turkish</td>
<td>9.2</td>
<td>8.1</td>
<td>8.5</td>
<td>9.4</td>
<td>9.9</td>
<td>9.4</td>
<td>10.4</td>
</tr>
<tr>
<td>English-Finnish</td>
<td>14.1</td>
<td>12.6</td>
<td>13.6</td>
<td>14.6</td>
<td>15.5</td>
<td>14.3</td>
<td>16.0</td>
</tr>
</tbody>
</table>

Table 5: Grid search on the pruning (.1, .25, .5) and stack parameters (1, 10, 100, 1000) for English-Turkish newstest2016 (cased BLEU)

<table>
<thead>
<tr>
<th>.1</th>
<th>.25</th>
<th>.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.60</td>
<td>9.51</td>
</tr>
<tr>
<td>10</td>
<td>9.82</td>
<td>9.86</td>
</tr>
<tr>
<td>100</td>
<td>9.86</td>
<td>9.90</td>
</tr>
<tr>
<td>1000</td>
<td>9.88</td>
<td>9.92</td>
</tr>
</tbody>
</table>

Table 6: Grid search on the pruning (.1, .25, .5) and stack parameters (1, 10, 100, 1000) for English-Finnish newstest2016 (cased BLEU)

<table>
<thead>
<tr>
<th>.1</th>
<th>.25</th>
<th>.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.85</td>
<td>15.06</td>
</tr>
<tr>
<td>10</td>
<td>14.92</td>
<td>15.30</td>
</tr>
<tr>
<td>100</td>
<td>14.92</td>
<td>15.33</td>
</tr>
<tr>
<td>1000</td>
<td>14.94</td>
<td>15.29</td>
</tr>
</tbody>
</table>

5.2 500-best Rescoring

We rescore 500-best candidate lists by first generating 500-best lists from Moses (Koehn et al., 2007) using the -N-best-list flag. We then use the Nematus (Sennrich et al., 2017) N-best list rescoring to rescore the list using our NMT model.

5.3 Lattice Rescoring

We also rescore PBMT lattices. We generate search graphs from the PBMT system by passing the -output-search-graph parameter to Moses. The search graphs are then converted to the OpenFST format (Allauzen et al., 2007) and operations to remove epsilon arcs, determinize, minimize and topsort are applied. Since the search graphs may be prohibitively large in size, we prune them to a threshold; we tune this threshold.\(^2\)

The core difficulty in lattice rescoring with NMT is that its RNN architecture does not permit efficient recombination of hypotheses on the lattice. Therefore, we apply a stack decoding algorithm (similar to the one used in PBMT) which groups hypotheses by the number of target words (the paper describing this work is under review). Figure 5.3 describes this pipeline.

5.4 Results

We use newstest2016 as a development set, and report the official results from newstest2017.

Tables 5 and 6 show the development set results for pruning thresholds of .1, .25, and .5 and stack sizes of 1, 10, 100, 1000. We chose not to use a stack size of 1000 in our final systems because the improvement in devset BLEU over a stack size of 100 is not large. For our final English-Turkish system, we use a pruning threshold of .25 and a stack size of 100; for our final English-Finnish system we use a pruning threshold of .5 and a stack size of 100.

Table 4 shows development results for the baseline PBMT, NMT systems, as well as the NMT ensembles, 500-best rescoring, and lattice rescoring. We also report test results for the 500-best rescoring, and lattice rescoring. On newstest2016, lattice rescoring outperforms 500-best rescoring by .5-1.1 BLEU, and on newstest2017, lattice rescoring outperforms 500-best rescoring by 1-1.7 BLEU. 500-best rescoring also outperforms PBMT, NMT system, and the NMT ensembles. While these results are not competitive with the best systems on newstest2017 in the evaluation campaign, it is interesting to note that lattice rescoring gave good performance among the models we compared. For future work it is worth re-running the lattice rescoring experiment using stronger baseline PBMT and NMT models.

\(^2\)Pruning removes arcs that do not appear on a lattice path whose score is within than \(t \cdot w\), where \(w\) is the weight of the FSTs shortest path, and \(t\) is the pruning threshold.
6 Conclusion

We submitted phrase-based systems for all 14 language pairs, syntax-based systems for 2 pairs, neural systems for 2 pairs, and two types of rescoring systems for 2 pairs. While many of these systems underperformed neural systems, they provide a strong baseline to compare the new neural systems to the previous state-of-the-art phrase-based systems. The gap between our neural systems and the top performing ones can be partially explained by a lack of large-scale back-translated data, which we plan to include in future work.

References


Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. CoRR abs/1508.07909.


