Optimizing Transformer for Low-Resource Neural Machine Translation

Araabi, A.; Monz, C.

DOI
10.18653/v1/2020.coling-main.304

Publication date
2020

Document Version
Final published version

Published in
The 28th International Conference on Computational Linguistics

License
CC BY

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)
Optimizing Transformer for Low-Resource Neural Machine Translation

Ali Araabi  
Informatics Institute  
University of Amsterdam  
a.araabi@uva.nl

Christof Monz  
Informatics Institute  
University of Amsterdam  
c.monz@uva.nl

Abstract

Language pairs with limited amounts of parallel data, also known as low-resource languages, remain a challenge for neural machine translation. While the Transformer model has achieved significant improvements for many language pairs and has become the de facto mainstream architecture, its capability under low-resource conditions has not been fully investigated yet. Our experiments on different subsets of the IWSLT14 training data show that the effectiveness of Transformer under low-resource conditions is highly dependent on the hyper-parameter settings. Our experiments show that using an optimized Transformer for low-resource conditions improves the translation quality up to 7.3 BLEU points compared to using the Transformer default settings.

1 Introduction

Despite the success of Neural Machine Translation (NMT) (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), for the vast majority of language pairs for which only limited amounts of training data exist (a.k.a. low-resource languages), the performance of NMT systems is relatively poor (Koehn and Knowles, 2017; Gu et al., 2018a). Most approaches focus on exploiting additional data to address this problem (Gulcehre et al., 2015; Sennrich et al., 2016; He et al., 2016; Fadaee et al., 2017). However, Sennrich and Zhang (2019) show that a well-optimized NMT system can perform relatively well under low-resource data conditions. Unfortunately, their results are confined to a recurrent NMT architecture (Sennrich et al., 2017), and it is not clear to what extent these findings also hold for the nowadays much more commonly used Transformer architecture (Vaswani et al., 2017).

Like all NMT models, Transformer requires setting various hyper-parameters but researchers often stick to the default parameters, even when their data conditions differ substantially from the original data conditions used to determine those default values (Gu et al., 2018b; Aharoni et al., 2019).

In this paper, we explore to what extent hyper-parameter optimization, which has been applied successfully to recurrent NMT models for low-resource translation, is also beneficial for the Transformer model. We show that with the appropriate settings, including the number of BPE merge operations, attention heads, and layers up to the degree of dropout and label smoothing, translation performance can be increased substantially, even for data sets with as little as 5k sentence pairs. Our experiments on different corpus sizes, ranging from 5k to 165k sentence pairs, show the importance of choosing the optimal settings with respect to data size.

2 Hyper-Parameter Exploration

In this section, we first discuss the importance of choosing an appropriate degree of subword segmentation before we describe the other optimal hyper-parameter settings.

Vocabulary representation. In order to improve the translation of rare words, word segmentation approaches such as Byte-Pair-Encoding (BPE) (Sennrich et al., 2016) have become standard practice in NMT. This is especially true for language pairs with small amounts of data where rare words are a common phenomenon. Sennrich and Zhang (2019) show that reducing the number of BPE merge operations...
can result in substantial improvements of up to 5 BLEU points for a recurrent NMT model. It is natural to assume that reducing the BPE vocabulary is similarly effective for Transformer.

### Architecture tuning
A current observation in neural networks, and in particular in Transformer architectures, is that increasing the number of model parameters improves performance (Raffel et al., 2019; Wang et al., 2019). However, those findings are mostly obtained for scenarios with ample training data and it is not clear if they are directly applicable to low-resource conditions. While Biljon et al. (2020) show that using fewer Transformer layers improves the quality of low-resource NMT, we expand our exploration towards the effects of using a narrow and shallow Transformer by reducing i) the number of layers in both the encoder and decoder, ii) the number of attention heads, iii) feed-forward layer dimension (\(d_{\text{ff}}\)), and iv) embedding dimensions (\(d_{\text{model}}\)).

### Regularization
Following Sennrich and Zhang (2019), we analyze the impact of regularization by applying dropouts to various Transformer components (Konda et al., 2015). In addition to regular dropout which is applied to the output of each sub-layer (feed-forward and self-attention) and after adding the positional embedding in both encoder and decoder (Vaswani et al., 2017), we employ attention dropout after the softmax for self-attention and also activation dropout inside the feed-forward sub-layers. Moreover, we drop entire layers using layer dropout (Fan et al., 2020). We further drop words in the embedding matrix using discrete word dropout (Gal and Ghahramani, 2016). We also experiment with larger label-smoothing factors (Müller et al., 2019).

## 3 Experiments
### 3.1 Experimental setup
Exploring all possible values for several hyper-parameters at once is prohibitively expensive from a computational perspective. Possible ways to circumvent this are random search (Bergstra and Bengio, 2012) or grid search for one hyper-parameter at a time. For simplicity, we opt for the latter. Table 1 shows the order in which the hyper-parameters are tuned. Once the optimal value of a hyper-parameter has been determined, it remains fixed for later steps; see Table 2. Obviously, there are no guarantees that this will result in a global optimum.

To be comparable with Sennrich and Zhang (2019), we take the TED data from the IWSLT 2014 German-English (De-En) shared translation task (Cettolo et al., 2014). We apply punctuation normal-
3.2 Results and discussions

**BPE effect.** To evaluate the effect of different degrees of BPE segmentation on performance, we consider merge operations ranging from 1k to 30k, training BPE on the full training corpus instead of subsets and also removing infrequent subword units when applying the BPE model. In contrast to earlier results for an RNN model, we observe that discarding infrequent subword units under extreme low-resource conditions is detrimental to the performance of Transformer. Sennrich and Zhang (2019) report that reducing BPE merge operations from 30k to 5k improves performance (+4.9 BLEU). We find that the same reduction in merge operations affects the Transformer model far less (+0.6 BLEU). We observe no significant differences between training BPE on the full training corpus and training on subsets. Thus, we always train BPE on subsets with an optimized number of merge operations (see Table 3).

**Architecture effect.** Table 2 shows the results of our system optimizations alongside the performance of our baselines. We notice that Transformer-big performs poorly on all datasets, which is most likely due to the much larger number of parameters requiring substantially larger training data. The system column in Table 2 shows our optimization steps on the 5k dataset, which are also applied to the larger datasets.

We gain substantial improvements over Transformer-base for various subset sizes. For the smallest dataset, as expected, reducing Transformer depth and width, including number of attention heads, feed-forward dimension, and number of layers along with increasing the rate of different regularization techniques is highly effective (+6 BLEU). The largest improvements are obtained by increasing the dropout rate (+1.4 BLEU), adding layer dropout to the decoder (+1.6 BLEU), and adding word dropout to the target side (+0.8 BLEU). Most of these findings also hold for the 10k and 20k datasets, but differ

<table>
<thead>
<tr>
<th>ID</th>
<th>System</th>
<th>5k</th>
<th>10k</th>
<th>20k</th>
<th>40k</th>
<th>80k</th>
<th>165k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Transformer-big</td>
<td>3.3</td>
<td>3.4</td>
<td>4.3</td>
<td>4.7</td>
<td>5.1</td>
<td>5.5</td>
</tr>
<tr>
<td>2</td>
<td>Transformer-base</td>
<td>8.3</td>
<td>11.9</td>
<td>16.8</td>
<td>23.2</td>
<td>28.0</td>
<td>32.1</td>
</tr>
<tr>
<td>3</td>
<td>2 + feed-forward dimension (2048 → 512)</td>
<td>8.8</td>
<td>12.0</td>
<td>16.7</td>
<td>22.3</td>
<td>27.7</td>
<td>31.7</td>
</tr>
<tr>
<td>4</td>
<td>3 + attention heads (8→ 2)</td>
<td>9.2</td>
<td>12.7</td>
<td>19.0</td>
<td>23.6</td>
<td>28.7</td>
<td>32.3</td>
</tr>
<tr>
<td>5</td>
<td>4 + dropout (0.1→ 0.3 )</td>
<td>10.6</td>
<td>17.0</td>
<td>21.9</td>
<td>26.7</td>
<td>31.0</td>
<td>33.4</td>
</tr>
<tr>
<td>6</td>
<td>5 + layers (6 → 5)</td>
<td>10.9</td>
<td>16.9</td>
<td>21.9</td>
<td>26.0</td>
<td>30.2</td>
<td>33.0</td>
</tr>
<tr>
<td>7</td>
<td>6 + label smoothing (0.1→ 0.6)</td>
<td>11.3</td>
<td>16.5</td>
<td>22.0</td>
<td>26.9</td>
<td>30.4</td>
<td>33.3</td>
</tr>
<tr>
<td>8</td>
<td>7 + decoder layerDrop (0 → 0.3)</td>
<td>12.9</td>
<td>17.3</td>
<td>22.5</td>
<td>26.9</td>
<td>30.3</td>
<td>33.1</td>
</tr>
<tr>
<td>9</td>
<td>8 + target word dropout (0 → 0.1)</td>
<td>13.7</td>
<td>18.1</td>
<td>23.1</td>
<td>27.0</td>
<td>30.7</td>
<td>33.0</td>
</tr>
<tr>
<td>10</td>
<td>9 + activation dropout (0 → 0.3)</td>
<td><strong>14.3</strong></td>
<td><strong>18.3</strong></td>
<td><strong>23.6</strong></td>
<td><strong>27.4</strong></td>
<td><strong>30.4</strong></td>
<td>32.6</td>
</tr>
</tbody>
</table>

Table 2: Results of Transformer optimized on the 5k dataset for different subsets and full corpus of IWSLT14 German → English. Averages over three runs from three different samples are reported.
### Table 3: Default parameters for Transformer-base and optimal settings for different dataset sizes based on the De→En development data.

<table>
<thead>
<tr>
<th>BPE operations</th>
<th>default</th>
<th>5k</th>
<th>10k</th>
<th>20k</th>
<th>40k</th>
<th>80k</th>
<th>165k</th>
</tr>
</thead>
<tbody>
<tr>
<td>feed-forward dimension</td>
<td>2048</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
<td>2048</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td>attention heads</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>dropout</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>layers</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>label smoothing</td>
<td>0.1</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>enc/dec layerDrop</td>
<td>0.0/0.0</td>
<td>0.0/0.3</td>
<td>0.0/0.2</td>
<td>0.0/0.2</td>
<td>0.0/0.1</td>
<td>0.0/0.1</td>
<td>0/0.1</td>
</tr>
<tr>
<td>src/tgt word dropout</td>
<td>0.0/0.0</td>
<td>0.0/0.1</td>
<td>0.0/0.1</td>
<td>0.1/0.1</td>
<td>0.1/0.1</td>
<td>0.2/0.2</td>
<td>0.2/0.2</td>
</tr>
<tr>
<td>activation dropout</td>
<td>0.0</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>batch size</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>8192</td>
<td>12288</td>
</tr>
</tbody>
</table>

Table 3: Results for Transformer-base optimized. T-opt results for Be, Gl, Sl, and Sk use the optimized settings on De→En development data for 5k, 10k, 10k, and 40k training examples, respectively.

Figure 1: Comparison between RNN and Transformer with base and optimized settings.

for larger subsets. By applying these settings to the 10k, 20k, 40k, 80k, and 165k datasets, BLEU scores increase by +6.4, +6.8, +4.2, +2.4, and +0.5 points, respectively. However, the effect of each adjustment is different for each dataset. For example, reducing the feed-forward layer dimension to 512 is only effective for the two smallest subsets.

We also conducted experiments with different values for the learning rate and warm-up steps using the inverse-square root learning rate scheduler, as implemented within Fairseq (Ott et al., 2019), which is slightly different from the proposed learning rate scheduler in the original Transformer paper. However, we did not observe any improvements over the default Transformer learning rate scheduler.

**Optimized parameter settings.** Table 3 shows the optimal settings for each dataset size, achieved by tuning the parameters on the development data. We observe that a shallower Transformer combined with a smaller feed-forward layer dimension and BPE vocabulary size is more effective under lower-resource conditions. However, as mentioned above, Transformer is not as sensitive to the BPE vocabulary size as RNNs and reducing the embedding dimension size is not effective.

Vaswani et al. (2017) and Chen et al. (2018) show that reducing the number of attention heads decreases the BLEU score under high-resource conditions. Raganato et al. (2020) show that using one
<table>
<thead>
<tr>
<th>sentences</th>
<th>words (En)</th>
<th>BLEU T-base</th>
<th>BLEU T-opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>En→De</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5k</td>
<td>100k</td>
<td>6.4</td>
<td>11.3</td>
</tr>
<tr>
<td>10k</td>
<td>200k</td>
<td>9.3</td>
<td>15.6</td>
</tr>
<tr>
<td>20k</td>
<td>410k</td>
<td>13.5</td>
<td>20.8</td>
</tr>
<tr>
<td>40k</td>
<td>830k</td>
<td>20.2</td>
<td>24.5</td>
</tr>
<tr>
<td>80k</td>
<td>1.6M</td>
<td>24.2</td>
<td>27.2</td>
</tr>
<tr>
<td>165k</td>
<td>3.4M</td>
<td>27.4</td>
<td>29.8</td>
</tr>
</tbody>
</table>

Table 5: Results for En→De based on the optimal settings for De→En for the corresponding corpus size (see Table 3).

The BLEU improvements in the bottom half of Table 4 show that determining the optimal settings on one language pair (De→En) is also effective for actual low-resource language pairs, especially if the size of the training data is taken into account. Furthermore, the results in Table 5 show that the optimal settings for De→En also hold for the opposite translation direction of the same language pair. They even carry over to translating from English for actual low-resource language pairs, see Table 6 which can be considered the more challenging scenario (Aharoni et al., 2019). Note that the results of Tables 5 and 6 and the bottom half of Table 4 are obtained by using the closest systems optimized on De→En subsets with respect to their number of training sentences.

To compare Transformer with an RNN architecture, we replicate the baseline and optimized RNN for low-resource NMT, as described in (Sennrich and Zhang, 2019), on our datasets. Figure 1 shows the BLEU scores for different data sizes. Surprisingly, even without any hyper-parameter optimization, Transformer performs much better than the RNN model under very limited data conditions. However, the optimized Transformer only outperforms the optimized RNN with more than 20k training examples.

4 Conclusion

In this paper, we study the effects of hyper-parameter settings for the Transformer architecture under various low-resource data conditions. While our findings are largely in line with previous work (Sennrich and Zhang, 2019) for RNN-based models, we show that very effective optimizations for RNN-based models such as reducing the number of BPE merge operations or using small batch sizes are less effective or even hurt performance. Our experiments show that a proper combination of Transformer configurations combined with regularization techniques results in substantial improvements over a Transformer system with default settings for all low-resource data sizes. However, under extremely low-resource conditions an optimized RNN model still outperforms Transformer.
References


