This Dissertation

entitled

IMPROVING NEURAL MACHINE TRANSLATION FOR LOW-RESOURCE LANGUAGES

typeset with nddiss2$_\varepsilon$ v3.2017.2 (2017/05/09) on July 30, 2021 for

Toan Q. Nguyen

This LaTeX 2$_\varepsilon$ classfile conforms to the University of Notre Dame style guidelines as of Fall 2012. However it is still possible to generate a non-conformant document if the instructions in the class file documentation are not followed!

> Be sure to refer to the published Graduate School guidelines at `http://graduateschool.nd.edu` as well. Those guidelines override everything mentioned about formatting in the documentation for this nddiss2$_\varepsilon$ class file.

This page can be disabled by specifying the "`noinfo`*" option to the class invocation.* (i.e.,`\documentclass[...,noinfo]{nddiss2e}`)

This page is *NOT* part of the dissertation/thesis. It should be disabled before making final, formal submission, but should be included in the version submitted for format check.

nddiss2$_\varepsilon$ documentation can be found at these locations:

`http://graduateschool.nd.edu`
`https://ctan.org/pkg/nddiss`

IMPROVING NEURAL MACHINE TRANSLATION FOR LOW-RESOURCE LANGUAGES

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Toan Q. Nguyen

David Chiang, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
July 2021

IMPROVING NEURAL MACHINE TRANSLATION FOR LOW-RESOURCE LANGUAGES

Abstract

by

Toan Q. Nguyen

Neural Machine Translation (NMT), which exploits the power of continuous representation and neural networks, has become the de facto standard choice in both academic research and industry. While the early non-attentional, recurrent neural network-based NMT already showed impressive performance, with the invention of the attention mechanism, it has been pushed further and achieved state-of-the-art performance. However, like other neural networks, these NMT systems are often data-hungry and require millions of training examples to achieve competitive results.

Unfortunately, only a few language pairs have this privilege. Most fall into the low-resource domain and often have only around 100k training sentence pairs or less. Some examples could be the LORELEI[1] or IWSLT datasets. This lack of data is particularly critical for the early recurrent neural network-based NMT systems, which are often outperformed by the traditional Phrase-based Machine Translation (PBMT) ones in low-resource scenarios. It also means more advanced techniques are required to effectively train a good NMT system for low-resource languages.

In this dissertation, I will show that with better use of training data, better normalization or improved model architecture, we can in fact build a competitive NMT system with only limited data at hand. On data, I propose two simple methods to improve NMT

[1] https://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

performance by better exploiting training resources. The first approach makes use of the relationship between languages for better transfer learning from one language pair to the other. The second one is a simple data augmentation via concatenation which can yield on average +1 BLEU on several language pairs. On normalization, I show how simple ℓ_2-based normalization at word embedding and hidden state levels can significantly improve translation for low-resource languages, with a particular focus on rare words. Finally, on model architecture, I investigate three simple yet effective changes. First, I propose a simple lexical module which can alleviate the mistranslation issue for rare words. Second, I study the Transformer model and show how a simple rearrangement of its components can improve both training and performance for low-resource languages. Lastly, I explore the untied positional attention in Transformer and demonstrate how it can improve both performance and interpretability.

CONTENTS

FIGURES

TABLES

ACKNOWLEDGMENTS

This work was supported in part by University of Southern California subcontract 67108176 under DARPA contract HR0011-15-C-0115. I was also supported by a fellowship from the Vietnam Education Foundation. I would like to express great appreciation to Dr. Sharon Hu for letting me use her group's GPU cluster (supported by NSF award 1629914), and to NVIDIA corporation for the donation of a Titan X GPU. A part of this thesis was also based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9116. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

A PhD is a long journey, and I would not make it without the support from so many people. I try my best to find all the names but if I miss any, please do know I always appreciate your kind help.

To David, thank you for being my advisor. Thank you for always being there when I have a question (even on weekends). Thanks for teaching me so many things from writing, editing, coding, brainstorming to pushing ideas further. And most important of all, thanks for always being so patient with me.

I would like to thank my committee members: Kyunghyun Cho, Tim Weninger, and Walter Scheirer for many insightful questions, discussions, and for reading this.

I would also want to thank my labmates: Kenton, Tomer, Arturo, Antonis, Brian, Justin,

Xing, Stephen, Darcey, and Huadong. Thank you for always reading my papers and giving detailed comments. Thanks for all the questions and discussions in group meetings. And thanks for driving me around :). To Tomer, I'm always grateful for our talks and your advices. To Stephen and Kenton, thanks for reading my thesis and providing such meticulous comments. Moving to a big country is tough, thank you Kenton for making it so much easier. Your parties and road trips certainly made South Bend's crazy winter much more bearable. Next time we meet, beer is on me, and I'll put a keg at your place first.

I would not be able to start my graduate study without the help from the Vietnam Education Foundation (VEF), University of Notre Dame, the Center for Research Computing, the ISSA office and most personally, Joyce Yeats. Thank you for always helping out, for bearing with my questions, for giving me such a great opportunity and great friends along the way.

To my South Bend friends, Kevin, Linh Đới, Lân Đinh, Duy Nguyễn, Hà Linh, anh Hậu, anh Luân, chị Loan, chị Lan, Quân, Ngọc, and many others, thanks for always giving me a Tết vibe, be it Tết or not. My PhD life would be totally intolerable without you all.

To my colleagues at Amazon and Google, anh Thuỵ, Julian, Katrin, Yuzhong, Liangwei, Shobhana, Julia, Colin, George, Pallavi, and Sweta, thank you for such great summers with tons of fun and the best work experience.

To my Boston friends, anh Ân, Hiền, Ốc, Ngọc, Huy, Thanh, Quân Mai, Mai Linh, Dzung, Zaw, Cún, Trang, Thông, and Nora, thanks for such great memories this whole crazy year. You all are such a wonderful group of friends. I'm so happy for my sister and brother-in-law to have found you. I'm moving soon but when I have a chance, I will find my way back here to join you again. Please take good care of my family.

To my dearest friends, Thảo Châu, anh Hoàng, anh Huy, Thái Nguyên, Hoà, Cường, Quyên, my goddaughter Mít, Rio Lâm, anh Sơn, bé Linh, Hạnh Em, Huy, Huyên, Nghĩa, Đức Lê, Đức Hồ, Hoàng Dương, thanks for countless emotional support through the years. To Rio, thank you for always being there to listen to me. To my best friend Cường, his wife

Quyên, and their baby daughter Mít, thank you for always giving me your unconditional support. To bé Linh, thanks for always being so kind.

To someone else, thanks for the thumbs up :).

To my family, anh Văn, Lý Mập, anh Sơn, Zee, Hiếu, Vy, chị Trâm, bé Mơ, Tin, Mon, Mi, bác Hai Sơn, bác Vân, and many others, thank you for always being there for me. To Lý Mập and anh Văn, thank you for taking care of the family. I would not be here today without your selfless support. To Zee and Hiếu, thanks for enduring me this past 5+ years. I learn so much from you two and I'm always looking forward to our future trips across America and the world. To you all, I am deeply grateful for taking care of Mom while I'm away, thank you!

Lời cuối con muốn dành cho Ba Việt và Mẹ Lan. Con cảm ơn Ba Mẹ đã luôn đặt việc học hành của con lên đầu. Con mang ơn Ba Mẹ vì đã hy sinh quá nhiều cho con. Con yêu Ba Mẹ. Con nhớ Ba rất nhiều!

A portrait of me, circa 2016–2021. Painted by my talented cousin bé Mơ.

CHAPTER 1

INTRODUCTION

Neural Machine Translation (NMT) is part of the *sequence-to-sequence learning* framework using neural networks. First introduced by Sutskever et al. [83], the model consists of an *encoder* which encodes the source sentence into a fixed-length vector, from which the *decoder* generates the target sentence. While achieving impressive results, this model still performs well below traditional phrase-based systems (PBMT). As pointed out by Cho et al. [11], the main reason lies in the limited capacity of the fixed-length vector representation to encode long sentences. To address this problem, Bahdanau et al. [6] invent the *attention mechanism* which allows the decoder to, at every time step, dynamically select a subset of encoder outputs to focus on instead of relying on a fixed-length representation. This *encoder-decoder-attention* model significantly improves NMT translation on long sentences and is the first NMT system to achieve comparable performance with phrase-based ones.

Even though Bahdanau et al. [6] successfully train a NMT model that outperforms the PBMT system, they only experiment on a large dataset (348M tokens were used for training). Most languages in the world do not fall in this high-resource category. For example, the LORELEI[1] datasets often have only a few thousands to around 100k sentence pairs per language pair. Like other neural networks, NMT is known for being data-hungry which makes it difficult to train a good NMT system for such low-resource languages. In fact, Zoph et al. [100] show that out-of-the-box NMT systems perform significantly poorer

[1] https://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

for NMT in this domain. Most notably, Koehn and Knowles [38] quantitatively showed that NMT needed up to around 100M tokens of training data to match PBMT's performance (Figure 1.1). Digging deeper, Arthur et al. [4] point out that NMT often makes more mistakes for rare words than PBMT, which I will demonstrate in chapter 5 is a critical problem for low-resource languages.

This dissertation consists of four reviewed publications [57, 58, 59, 60] and one unfinished work. They are all under a common theme that we can improve NMT performance for low-resource languages via **better data exploitation**, **better modeling**, or **better normalization**. Specifically:

Better data exploitation: In chapter 3, I show that we can improve NMT performance on a language pair using another related language pair by exploiting their similarities at subword level. In chapter 4, I study a simple but effective method for data augmentation via concatenation for low-resource NMT and explain why it works.

Better modeling: I show that we can improve NMT by using a better neural network architecture. In particular, for RNN-based NMT, I propose a lexical module which can significantly alleviates the rare word mistranslation problem (chapter 5). For Transformer, I suggest a better placement of layer normalization and residual connections which helps to improve stability of training (chapter 6). Finally, in chapter 7, I study a simple change to Transformer's attention by untying positional attention from word attention. I show how this could significantly improve performance for low-resource language pairs and helps with model interpretability.

Better normalization: I show that using simple ℓ_2 normalization, we can improve both the rare word mistranslation and training (chapter 5 and 6).

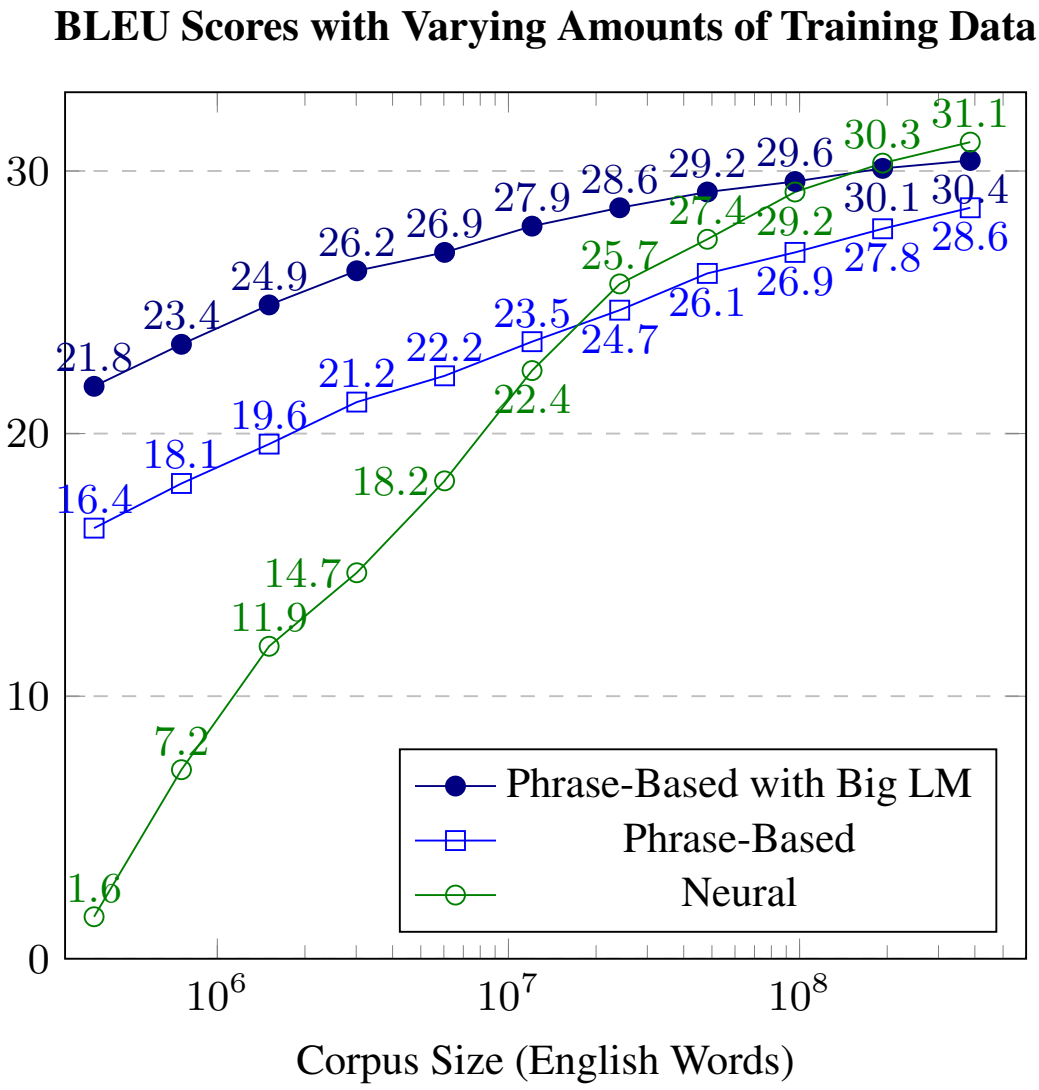


Figure 1.1: Performance of PBMT and NMT with different amount of data (Koehn and Knowles [38])

Thesis statement: *While NMT inherently requires large amount of data for training, with better data exploitation, better modeling, and better normalization we can successfully train strong NMT systems for low-resource languages.*

CHAPTER 2

BACKGROUND

Because NMT is a sequence-to-sequence learning problem, the apparent choice for the encoder or decoder architecture is recurrent neural networks (RNNs). Oftentimes, RNN variants such as LSTM [27] or GRU [12] are used. On the other hand, Transformer [87] takes a radical departure from this. This model relies entirely on the attention mechanism for the encoder to compute source sentence representation and for the decoder to remember which part of the target sentence has been decoded so far. Aside from its state-of-the-art performance, Transformer is also appealing for its train-time parallelism which better exploits the power of graphics processing units (GPUs). Transformer was not the first to replace RNNs; in fact, an earlier work by Gehring et al. [17] uses convolutional neural networks (CNN) instead of LSTM and achieves state-of-the-art performance on many NMT tasks. However, since Transformer has been shown to perform better and has become the *de facto* choice for NMT, I will discuss only it in this document and refer readers to Gehring et al. [17] for more details of the CNN-based approach.

In this chapter, I will provide a rough overview of the RNN-based and attention-based (Transformer) NMT models. [83, 12, 6, 11, 49] are good resources for a more complete picture of the former. For the latter, aside from the original work [87], Rush [71] provides a complete dissection on the model and is a good reference for implementation.

2.1 RNN-based NMT

A RNN-based NMT often consists of a source embedding $W_s \in \mathcal{R}^{d \times V_x}$, a target input embedding $W_y \in \mathcal{R}^{d \times V_y}$, a target output embedding $W_o \in \mathcal{R}^{V_y \times d}$, an encoder f and a de-

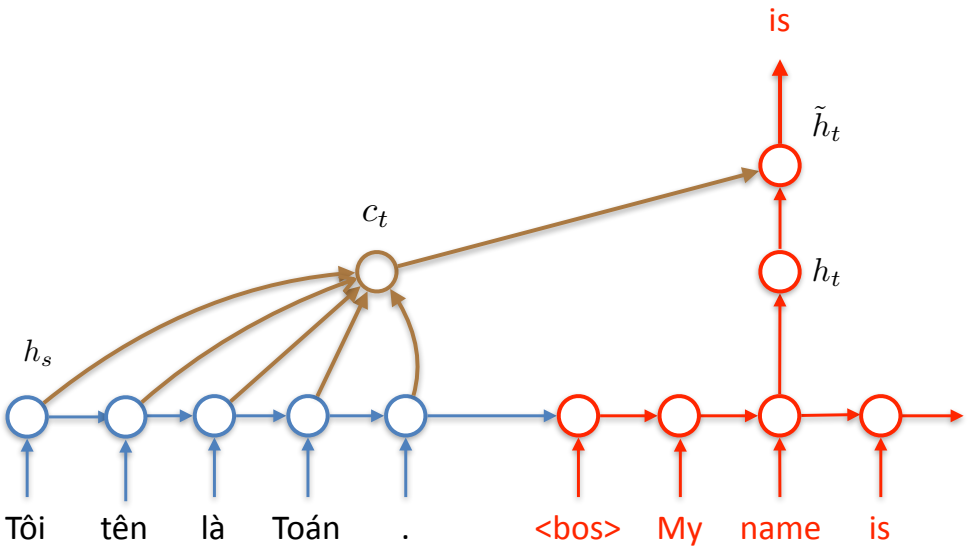


Figure 2.1: An RNN-based NMT with attention model. The encoder is represented in blue, the decoder is in red and the attention mechanism is in brown. Best viewed in color.

coder g. V_x, V_y, d are source vocabulary size, target vocabulary size and embedding size, respectively. Both f and g are some variants of RNN; typically, an LSTM or a GRU is used.

Each source sentence is represented as a sequence of one-hot vectors $(x_1, \cdots, x_{L_x})$, $x_i \in \mathcal{R}^{V_x}$, with L_x being the source sentence length. The input to the encoder is:

$$\mathbf{x} = (W_s x_1, \cdots, W_s x_{L_s}) \tag{2.1}$$

The encoder reads in this sequence step-by-step and generates, at time step s, an output vector $h_s = f(x_s, h_{s-1})$. The decoder works similarly and generates, at time step t, an output vector $h_t = g(y_t, h_{t-1})$. This h_t is used by the attention mechanism to calculate a weight α_s for each h_s. The context vector c_t is then simply a weighted sum over all encoder outputs:

$$c_t = \sum_{s=1}^{L_x} \alpha_s h_s \tag{2.2}$$

While I refer readers to [6] or [49] for other ways to calculate α_s, a simple approach called *dot product* by Luong et al. [49] is:

$$\mathrm{e}_s = h_t^T h_s \tag{2.3}$$

$$\alpha_s = \frac{\exp(\mathrm{e}_s)}{\sum_{s'=1}^{L_x} \exp(\mathrm{e}_{s'})} \tag{2.4}$$

The decoder output h_t and context vector c_t are combined, either by a nonlinear feed-forward neural network or simply summed together, to produce the final output vector $\tilde{h}_t$. Finally, the predicted probability of the t-th target word is:

$$p(y_t \mid y_{<t}, \mathbf{x}) = \mathrm{softmax}(W_o \tilde{h}_t) \tag{2.5}$$

The model is trained to maximize the conditional probability:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{L_y} p(y_t \mid y_{<t}, \mathbf{x}) \tag{2.6}$$

with L_y be the target sentence length.

Figure 2.1 gives a visual representation of these architectural mechanisms.

2.2 Transformer

The main difference between Transformer model and an RNN-based NMT model is Transformer replaces the RNN with the self-attention mechanism as shown in Figure 2.2. While I refer readers to Vaswani et al. [87] for a more complete formulation, in this section I will briefly describe four main components of Transformer.

Multi-head Attention In Transformer, each attention layer could have multiple *heads* whose outputs are combined at the end. Given an input source sequence $\mathbf{x} = (x_1, \cdots, x_{L_s}), x_i \in \mathcal{R}^d$, we can pack this sequence into a matrix $X = \mathrm{concat}(\mathbf{x}), X \in \mathcal{R}^{L_s \times d}$. The i-th attention

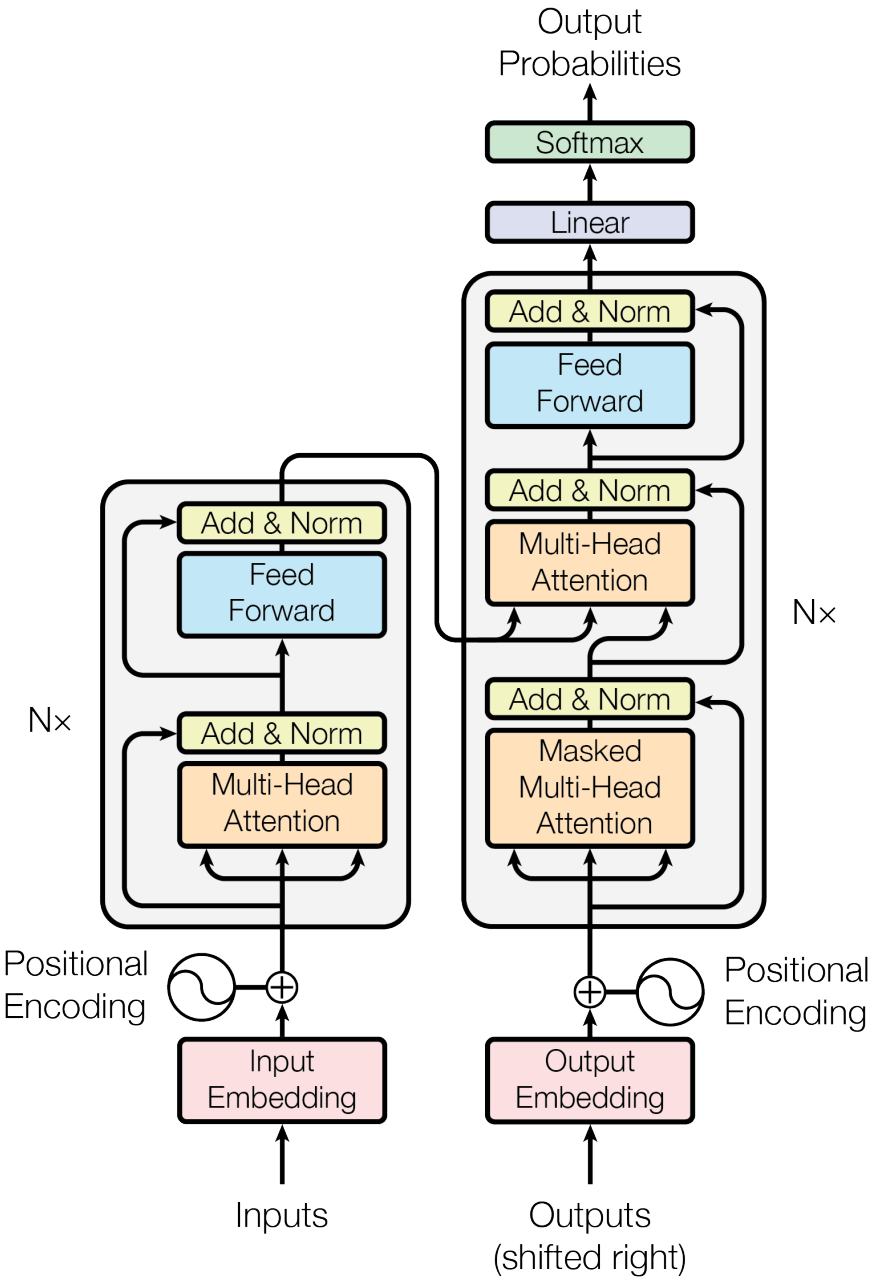


Figure 2.2: The Transformer model architecture [87]

head maps the input X into a query (Q), a key (K), and a value (V):

$$Q = XW_i^Q \tag{2.7}$$

$$K = XW_i^K \tag{2.8}$$

$$V = XW_i^V \tag{2.9}$$

where $W_i^Q, W_i^K, W_i^V \in \mathcal{R}^{d \times d_k}$ are learnable variables. The output of the i-th attention head is then defined as:

$$H_i = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.10}$$

These matrices H_i are then combined and mapped to the final output:

$$H = \text{concat}(H_1, \cdots, H_n)W^O \tag{2.11}$$

where $W^O \in \mathcal{R}^{d \times d}$ is also a learnable parameter.

Oftentimes, we set $d = nd_k$ so that there is no increase in both number of parameters or computations compared to using a single attention model. It works similarly on the decoder, but we also need to make sure it does not see the future. This can be done by simply setting the corresponding values within the softmax in equation 2.10 to $-\infty$. Since these two attention mechanisms allow encoder or decoder to attend to themselves, they are also called *self-attention*. At each decoder layer, the decoder also employs a similar attention mechanism to attend to encoder's output, we call this *cross-attention*. One important

change is:

$$Q = YW_i^Q \tag{2.12}$$

$$K = XW_i^K \tag{2.13}$$

$$V = XW_i^V \tag{2.14}$$

where X is the encoder's output, Y is the current decoder's output.

Feed-forward Networks The second important component of Transformer is the feed-forward networks. This is essentially two linear layers with RELU [19]:

$$FFN(X) = max(0, XW_1 + b_1)W_2 + b_2 \tag{2.15}$$

where $W_1 \in \mathcal{R}^{d \times d_{ff}}$, $W_w \in \mathcal{R}^{d_{ff} \times d}$, $b_1, b_2 \in \mathcal{R}^d_{ff}$, $d_{ff} > d$.

Layer Normalization and Residual Connection Residual connections [24] are employed at every sublayer of Transformer. This means the input to each multi-head attention or feed-forward layer is summed element-wise to their output. The output from each residual addition is then fed through a layer normalization layer [5]. The placement of the layer normalization is actually very important to Transformer's stability which we will discuss in section 6.

Positional Encoding To let Transformer be aware of the order of the sequence, we use a cosine positional embedding which is defined as:

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d}}) \tag{2.16}$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d}}) \tag{2.17}$$

This positional embedding is simply summed element-wise with the word embedding before being fed into the encoder and decoder.

CHAPTER 3

TRANSFER LEARNING ACROSS LOW-RESOURCE, RELATED LANGUAGES FOR NEURAL MACHINE TRANSLATION

In this chapter, I will introduce the first data exploitation method. More specifically, I show how we can make use of the relatedness between languages to improve transfer learning from one language pair to another. The content of this part was published at IJCNLP 2017 [58].

Abstract *We present a simple method to improve neural translation of a low-resource language pair using parallel data from a related, also low-resource, language pair. The method is based on the transfer method of Zoph et al., but whereas their method ignores any source vocabulary overlap, ours exploits it. First, we split words using Byte Pair Encoding (BPE) to increase vocabulary overlap. Then, we train a model on the first language pair and transfer its parameters, including its source word embeddings, to another model and continue training on the second language pair. Our experiments show that transfer learning helps word-based translation only slightly, but when used on top of a much stronger BPE baseline, it yields larger improvements of up to 4.3 BLEU.*

3.1 Introduction

A common strategy to improve learning of low-resource languages is to use resources from related languages [54]. However, adapting these resources is not trivial. NMT offers some simple ways of doing this. For example, Zoph et al. [100] train a *parent* model on a (possibly unrelated) high-resource language pair, then use this model to initialize a

child model which is further trained on a low-resource language pair. In particular, they showed that a French-English model could be used to improve translation on a wide range of low-resource language pairs such as Hausa-, Turkish-, and Uzbek-English.

In this chapter, we explore the opposite scenario, where the parent language pair is also low-resource, but related to the child language pair. We show that, at least in the case of three Turkic languages (Turkish, Uzbek, and Uyghur), the original method of Zoph et al. [100] does not always work, but it is still possible to use the parent model to considerably improve the child model.

The basic idea is to exploit the relationship between the parent and child language lexicons. Zoph et al.'s original method makes no assumption about the relatedness of the parent and child languages, so it effectively makes a random assignment of the parent-language word embeddings to child-language words. But if we assume that the parent and child lexicons are related, it should be beneficial to transfer source word embeddings from parent-language words to their child-language equivalents.

Thus, the problem amounts to finding a representation of the data that ensures a sufficient overlap between the vocabularies of the languages. To do this, we map the source languages to a common alphabet and use Byte Pair Encoding (BPE) [78] on the union of the vocabularies to increase the number of common subwords.

In our experiments, we show that transfer learning helps word-based translation, but not always significantly. But when used on top of a much stronger BPE baseline, it yields larger and statistically significant improvements. Using Uzbek as a parent language and Turkish and Uyghur as child languages, we obtain improvements over BPE of 0.8 and 4.3 BLEU, respectively.

3.2 Method

The basic idea of our method is to extend the transfer method of Zoph et al. [100] to share the parent and child's source vocabularies, so that when source word embeddings are

transferred, a word that appears in both vocabularies keeps its embedding. In order for this to work, it must be the case that the parent and child languages have considerable vocabulary overlap, and that when a word occurs in both languages, it often has a similar meaning in both languages. Thus, we need to process the data to make these two assumptions hold as much as possible.

3.2.1 Transliteration

If the parent and child language have different orthographies, it should help to map them into a common orthography. Even if the two use the same script, some transformation could be applied; for example, we might change French *-eur* endings to Spanish *-or*. Here, we take a minimalist approach. Turkish and Uzbek are both written using Latin script, and we did not apply any transformations to them. Our Uyghur data is written in Arabic script, so we transliterated it to Latin script using an off-the-shelf transliterator.[1] The transliteration is a string homomorphism, replacing Arabic letters with English letters or consonant clusters independent of context.

3.2.2 Segmentation

To increase the overlap between the parent and child vocabularies, we use BPE to break words into subwords. For the BPE merge rules to not only find the common subwords between two source languages but also ensure consistency between source and target segmentation among each language pair, we learn the rules from the union of source and target data of both the parent and child models. The rules are then used to segment the corpora. It is important to note that this results in a single vocabulary, used for both the source and target languages in both models.

[1] `https://cis.temple.edu/~anwar/code/latin2uyghur.html`

TABLE 3.1

NUMBER OF TOKENS ($\times 10^6$) AND SENTENCES ($\times 10^3$) IN TRAINING DATA

	word-based		BPE 5k		BPE 60k	
model	toks	sents	toks	sents	toks	sents
Uzb par	1.5	102	2.4	92	1.9	103
Tur chd	0.9	56	1.5	50	1.2	57
Uzb par	1.5	102	2.4	90	2.0	103
Uyg chd	1.7	82	2.1	77	2.0	88

3.3 Experiments

We used Turkish-, Uzbek-, and Uyghur-English parallel texts from the LORELEI program. We tokenized all data using the Moses toolkit [39]; for Turkish-English experiments, we also truecased the data. For Uyghur-English, the word-based models were trained in the original Uyghur data written in Arabic script; for BPE-based systems, we transliterated it to Latin script as described above.

For the word-based systems, we fixed the vocabulary size and replaced out-of-vocabulary words with `_UNK`. We tried different sizes for each language pair; however, each word-based system's target vocabulary size is limited by that of the child, so we could only use up to 45,000 word types for Turkish-English and 20,000 for Uyghur-English.

The BPE-based systems could make use of bigger vocabulary size thanks to the combination of both parent and child source and target vocabularies. We varied the number of BPE merge operations from 5,000 to 60,000. Instead of using a fixed vocabulary cutoff,

we used the full vocabulary; to ensure the model still learns how to deal with unknown words, we trained on two copies of the training data: one unchanged, and one in which all rare words (whose frequency is less than 5) were replaced with `_UNK`. Accordingly, the number of epochs was halved.

Following common practice, we fixed an upper limit on training sentence length (discarding longer sentences). Because the BPE-based systems have shorter tokens and therefore longer sentences, we set this upper limit differently for the word-based and BPE-based systems to approximately equalize the total size of the training data. This led to a limit of 50 tokens for word-based models and 60 tokens for BPE-based models. See Table 3.1 for statistics of the resulting datasets.

We trained using Adadelta [96], with a minibatch size of 32 and dropout with a dropout rate of 0.2. We rescaled the gradient when its norm exceeded 5. For the Uzbek-English to Turkish-English experiment, the parent and child models were trained for 100 and 50 epochs, respectively. For the Uzbek-English to Uyghur-English experiment, the parent and child models were trained for 50 and 200, respectively. As mentioned above, the BPE models were trained for half as many epochs because their data is duplicated.

We used beam search for translation on the dev and test sets. Since NMT tends to favor short translations [11], we use the length normalization approach of Wu et al. [94] which uses a different score $s(e \mid f)$ instead of log-probability:

$$s(e \mid f) = \frac{\log p(e \mid f)}{lp(e)}$$

$$lp(e) = \frac{(5 + |e|)^{\alpha}}{(5 + 1)^{\alpha}}.$$

We set $\alpha = 0.8$ for all of our experiments.

We stopped training when the tokenized BLEU score was maximized on the development set. We also optimized the vocabulary size and the number of BPE operations for the word-based and BPE-based systems, respectively, to maximize the tokenized BLEU on the

development set.

After translation at test time, we rejoined BPE segments, recased, and detokenized. Finally, we evaluated using case-sensitive BLEU.

As a baseline, we trained a child model using BPE but without transfer (that is, with weights randomly initialized). We also compared against a word-based baseline (without transfer) and two word-based systems using transfer without vocabulary-sharing, corresponding with the method of Zoph et al. [100]: one where the target word embeddings are fine-tuned, and one where they are frozen.

3.4 Results and Analysis

Our results are shown in Table 3.2. In this low-resource setting, we find that transferring word-based models does not consistently help. On Turkish-English, both transfer methods give only a statistically insignificant improvement ($p > 0.05$); on Uyghur-English, transfer without freezing target embeddings helps somewhat, but transfer with freezing helps only insignificantly.

In both language pairs, the models that use BPE perform much better than their word-based counterparts. When we apply transfer learning to this much stronger baseline, we find that the relative improvements actually increase; that is, the combined effect of BPE and transfer learning is more than additive. On Turkish-English, the improvement is +0.8 BLEU over the BPE baseline; on Uyghur-English, a healthy +4.3 over the BPE baseline.

A similar pattern emerges when we examine the best BLEU scores on the development set (Figure 3.1). Whereas word-based transfer methods help very little for Turkish-English, and help or hurt slightly for Uyghur-English, our BPE-based transfer approach consistently improves over both the baseline and transfer word-based models. We surmise that the improvement is primarily due to the vocabulary overlap created by BPE (see Table 3.3).

TABLE 3.2

TEST BLEU SCORES

		baseline		transfer		transfer+freeze	
		BLEU	size	BLEU	size	BLEU	size
Tur-Eng	word-based	8.1	30k	8.5*	30k	8.6*	30k
	BPE	12.4	10k	13.2†	20k	—	—
Uyg-Eng	word-based	8.5	15k	10.6†	15k	8.8*	15k
	BPE	11.1	10k	15.4‡	8k	—	—

Whereas transfer learning at word-level does not always help, our method consistently yields a significant improvement over the stronger BPE systems. Scores are case-sensitive **test** BLEU. Key: `size` = vocabulary size (word-based) or number of BPE operations (BPE). The symbols † and ‡ indicate statistically significant improvements with $p < 0.05$ and $p < 0.01$, respectively, while $*$ indicates a statistically insignificant improvement ($p > 0.05$).

TABLE 3.3

AMOUNT OF CHILD'S SOURCE TYPES THAT APPEAR IN PARENT

task	settings	train	dev
Tur-Eng	word-based 30k	3.9%	3.6%
	BPE 20k	58.8%	25.0%
Uyg-Eng	word-based 15k	0.5%	1.7%
	BPE 8k	57.2%	48.5%

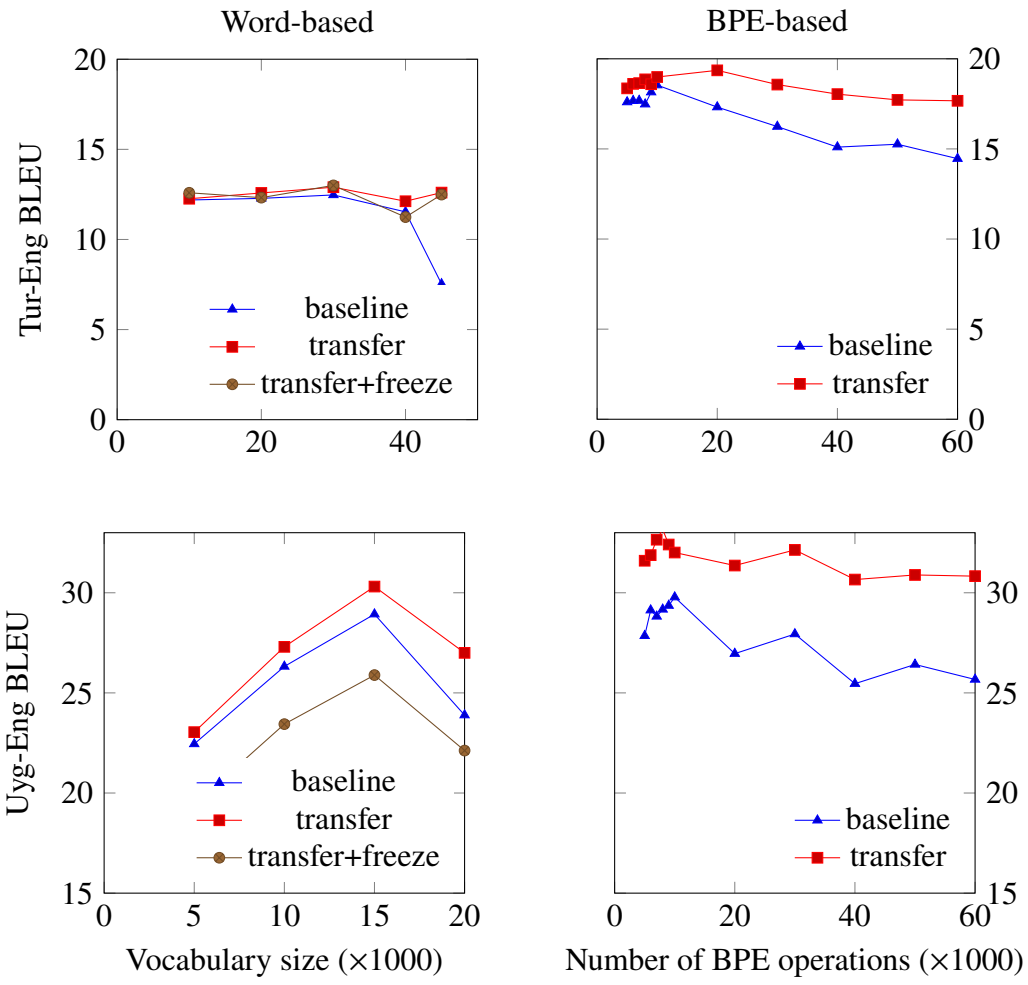


Figure 3.1. Tokenized **dev** BLEU scores for various settings as a function of the number of word/subword types. Key: baseline = train child model only; transfer = train parent, then child model; +freeze = freeze target word embeddings in child model.

CHAPTER 4

DATA AUGMENTATION BY CONCATENATION FOR LOW-RESOURCE TRANSLATION: A MYSTERY AND A SOLUTION

In the previous chapter, we have looked into how to exploit the relatedness between languages to improve transfer learning from one language pair to the other. In this one, I will introduce another method to make use of training data. In particular, I will show how pseudo-data from concatenation of training sentences can improve performance for low-resource settings. This chapter appeared as a publication at IWSLT 2021 [60].

Abstract *In this chapter, we investigate the driving factors behind concatenation, a simple but effective data augmentation method for low-resource neural machine translation. Our experiments suggest that discourse context is unlikely the cause for concatenation improving BLEU by about +1 across four language pairs. Instead, we demonstrate that the improvement comes from three other factors unrelated to discourse: context diversity, length diversity, and (to a lesser extent) position shifting.*

4.1 Introduction

Many attempts have been made to augment neural machine translation (MT) systems to use discourse context [33, 80, 75, 98, 82, 42, 35, 85, 99, 31]. One particularly simple method is to concatenate consecutive pairs of sentence-pairs during training, but not during translation [86, 56, 40].[1] In this chapter, we confirm that this simple method helps, by

[1] As this work was being finalized, Kondo et al. [40] published independent work also presenting random concatenation as data augmentation for NMT. They find that concatenation helps the model translate long sentences better, while the focus of the present paper is to explain thoroughly why it helps.

roughly +1 BLEU across four low-resource language pairs. But we demonstrate that the reason it helps is *not* discourse context, because concatenating *random* pairs of sentence-pairs yields the same improvement.

Instead, we view concatenation as a kind of data augmentation or noising method (one which pleasantly requires no alteration to the text, unlike data augmentation methods that disturb word order [7, 2] or replace words with automatically-selected words [16, 15, 93]). Concatenating random sentences is easier than concatenating consecutive sentences, because many parallel corpora discard document boundaries, drop sentence-pairs, or even reorder sentence-pairs, so it can be difficult to know which sentence-pairs are truly consecutive.

But the fact that random concatenation helps so much creates a mystery, which is the focus of the paper. If the reason is not discourse context, what is the reason? We consider three new hypotheses:

- Random concatenation creates greater diversity of positions, because it lets the model see sentences shifted by effectively random distances.
- Random concatenation creates greater diversity of contexts, helping the model learn what *not* to attend to.
- Random concatenation creates greater diversity of sentence lengths within a mini-batch.

Through a careful ablation study, we demonstrate that all three of these factors more or less contribute to the improvement, and together completely explain the improvement.

4.2 Concatenation

We first present the concatenation methods and confirm that they improve low-resource translation.

4.2.1 Methods

Let $D_{\text{orig}} = \{(x_i, y_i) \mid i = 1, \ldots, N\}$ be the original training data. We consider two concatenation strategies:

`CONSEC` Concatenate consecutive sentence-pairs: $D_{\text{new}} = \{(x_i x_{i+1}, y_i y_{i+1}) \mid i = 1, \ldots, N - 1\}$.

`RAND` Same as `CONSEC`, but randomly permute D_{orig} before concatenation.

For example, consider the following en→vi sentence pairs:

And I think back . → *Và tôi nghĩ lại .*

I think back to my father . → *Tôi nghĩ lại về cha tôi .*

With <BOS>/<EOS> markings, the concatenated sentence-pairs would be:

source input: *And I think back . <EOS> I think back to my father . <EOS>*

target input: *<BOS> Và tôi nghĩ lại . <BOS> Tôi nghĩ lại về cha tôi .*

target output: *Và tôi nghĩ lại . <EOS> Tôi nghĩ lại về cha tôi . <EOS>*

Since consecutive training examples often come from the same document, `CONSEC` lets the model look at some of the discourse context during training. In `RAND`, however, the concatenated sentences are almost always unrelated. In both cases, we train models on the combined data, $D_{\text{orig}} \cup D_{\text{new}}$.

4.2.2 Initial experiments

We experiment on four low-resource language pairs: {Galician, Slovak} to English and English to {Hebrew, Vietnamese} [70, 47] using Transformer [87]. We use the same setup as Nguyen and Salazar [59], with PreNorm, FixNorm and ScaleNorm, as it has been shown to perform well on low-resource tasks. Since the data comes pre-tokenized, we only apply BPE. Data statistics and hyper-parameters are summarized in Table 6.1.

TABLE 4.1

DATA AND MODEL STATISTICS

	train/dev/test sents. (x1000)	train steps/epoch	epochs	layers	heads	dropout	BPE
gl→en	10/0.68/1	100	1000	4	4	0.4	3k
sk→en	61/2.27/2.45	600	200	6	8	0.3	8k
en→vi	133/1.55/1.27	1500	200	6	8	0.3	8k
en→he	210/4.52/5.51	2000	200	6	8	0.3	8k

For baseline, the training data is D_{orig}. For concatenation, we first create D_{new}, then combine it with D_{orig} to create the training data. Following Morishita et al. [52], we randomly shuffle the training data and read it in chunks of 10k examples. Each chunk is sorted by source length before being packed into minibatches of roughly 4096 source/target tokens each.

We calculate tokenized BLEU using `multi-bleu.perl` [39] and measure statistical significance using bootstrap resampling [37].

As seen in Table 4.2, concatenation consistently outperforms the baseline across all datasets with significant improvement ($p < 0.01$) on almost every case. We observe that there is generally more improvement with less training data. For example, en→he with more than 200k training examples gets only +0.5 BLEU, but gl→en with only 10k sentences achieves +1.3 BLEU. On average, this method yields +1 BLEU over all four language pairs. We can also see that concatenating consecutive or random sentence pairs results in similar performance. For this reason, all the following ablation studies are conducted with `RAND` unless noted otherwise

TABLE 4.2

CONCATENATION RESULTS

	gl→en		sk→en		en→vi		en→he		average			
	dev	test	dev	test	dev	test	dev	test	dev	Δ	test	Δ
baseline	22.9	20.7	29.2	30.3	29.0	32.7	30.3	28.1	27.8		28.0	
CONSEC	24.9	22.9†	30.3	31.5†	29.2	33.5†	30.6	28.6†	28.8	+1.0	29.1	+1.1
RAND	25.3	23.1†	30.3	31.6†	29.2	33.0	30.8	28.5†	28.9	+1.1	29.0	+1.0

Consecutive (CONSEC) and random (RAND) concatenation give the same BLEU improvement across our four low-resource language pairs. † = statistically significant improvement on the test set compared to baseline ($p < 0.01$).

4.3 Analysis

Why does a method as simple as concatenation help so much? We reject the initial hypothesis that the model is assisted by discourse context (§4.3.1) and consider three new hypotheses related to data augmentation (§4.3.2–§4.3.4).

4.3.1 Discourse context

Since consecutive sentences often come from the same document, CONSEC provides the model with more discourse context during training. For RAND, however, the two sentences in a generated example are unlikely to have any relation at all. Despite this difference, we can see from Table 4.2 that both CONSEC and RAND achieve similar performance.

To better understand whether discourse context plays any role here, we conduct a simple experiment. We perform concatenation just as in CONSEC and RAND, but on the dev set (as well as the training set), and measure BLEU on the concatenated dev set. The new BLEU scores are shown in Table 4.3, showing that even having discourse context available at translation time does not enable CONSEC to do better than RAND. While we acknowledge

TABLE 4.3

DEV BLEU SCORES FOR CONSEC AND RAND

	dev BLEU				
	gl→en	sk→en	en→vi	en→he	avg
CONSEC	23.5	29.6	29.7	31.1	28.5
RAND	24.0	29.2	29.4	31.3	28.5

Even when we concatenate consecutive sentence-pairs during translation, CONSEC does not outperform RAND. All BLEU scores in this table are computed on concatenated versions of the dev sets, and so are not comparable with the scores in other tables.

that there could be improvement due to discourse context that is not captured by BLEU, we can also say that the gain in BLEU that we do observe with concatenation is independent of the availability of discourse context.

4.3.2 Position shifting

Since the Transformer uses absolute positional encodings, if a word is observed only a few times, the model may have difficulty generalizing to occurrences in other positions. Moreover, if there are too few long sentences, the model may have difficulty translating words very far from the start of the sentence. In concatenation, the second sentence is shifted by a random distance n with n being the first sentence's length in the sense that its positions are indexed from n instead of 0. We hypothesize that this allows the model to see, and thus, to be better-trained on more positions.

If the improvement indeed comes from position shifting, we should be able to reproduce it without concatenation. In concatenation, we train on $D_{\text{orig}} \cup D_{\text{new}}$. While D_{new} has the same number of sentences as D_{orig} (§4.2.1), each sentence is a concatenation of two

TABLE 4.4

POSITION SHIFTING ABLATION STUDY

Row		gl→en	sk→en	en→vi	en→he	avg	Δ
1	**baseline**	22.9	29.2	29.0	30.3	27.8	
2	**baseline + sim-shift**	22.7	29.8†	29.0	30.4	28.0	+0.2
3	**baseline + uniform-shift**	23.8	29.8†	29.3	30.5	28.4	+0.6
4	**RAND**	25.3†	30.3†	29.2	30.8†	28.9	+1.1
5	**RAND + uniform-shift**	25.5†	30.7†	29.14	30.7†	29.0	+1.2

Position shifting improves accuracy somewhat, but the version of position shifting that mimics that of concatenation (sim-shift) gives less of an improvement than shifting by distances uniformly sampled from [0, 100] (uniform-shift). All BLEU scores are on dev sets. † = statistically significant improvement compared to baseline ($p < 0.01$).

sentences in D_{orig}. This means that in total, 1/3 of sentences are shifted. So, we simulate the position-shifting that occurs in concatenation as follows. For each sentence-pair (f_i, e_i) in the training data, with probability 1/3, choose a random training sentence pair (f_j, e_j) and shift f_i by $|f_j|$ and e_i by $|e_j|$. We call this system sim-shift.

We also try a more uniform shifting method, called uniform-shift, in which we sample, with probability 0.1, distances s and t uniformly from [0, 100] and shift f_i by s and shift e_i by t.

Lines 1–3 in Table 4.4 show that both uniform-shift and sim-shift do help somewhat. Surprisingly, sim-shift is outperformed by uniform-shift, especially for gl→en with a gap of 0.9 BLEU. We attribute this to the fact that uniform-shift tends to shift sentences for longer distances and hence better generalizes to longer sentences. Indeed, as shown in Figure 4.1 (bottom), most training sentences in gl→en are shorter than 60. In Figure 4.1 (top), we see that uniform-shift outperforms sim-shift by the largest margin on the longest sentences. Nevertheless, adding uniform-shift on top of RAND (Table 4.4, row 5) only improves it very

TABLE 4.5

FEATURE ABLATION STUDY

Row		gl→en	sk→en	en→vi	en→he	avg	Δ
1	**RAND**	25.3†	30.3†	29.2	30.8†	28.9	
2	**RAND + mask**	24.3†	30.0†	28.9	30.6	28.5	−0.4
3	**RAND + sep-batch**	24.9†	30.1†	29.1	30.6	28.7	−0.2
4	**RAND + mask + sep-batch**	23.2	29.8†	29.3	30.5	28.2	−0.7
5	**RAND + mask + sep-batch + reset-pos**	23.1	29.6†	28.9	30.5	28.0	−0.9
6	**baseline**	22.9	29.2	29.0	30.3	27.8	−1.1

Masking attention to prevent concatenated sentences from attending to one another (**mask**) reduces accuracy. Forming minibatches so as to prevent concatenation from increasing length diversity (**sep-batch**) also reduces accuracy. When we do both and also remove the effect of position shifting (**reset-pos**), we eliminate essentially all the improvement due to concatenation. All BLEU scores are on dev sets. † = statistically significant improvement compared to baseline ($p < 0.01$)

slightly.

To conclude, we show that position shifting can have a positive impact on low-resource NMT. However, it seems to contribute only a small part of the improvement due to concatenation, as we will confirm below (§4.3.5).

4.3.3 Context diversity

In an attention layer, each query word is free to attend to any key word, and the model must learn to distinguish the keys that are related to a query from those that are not. Let us call the former *positive contexts* and the latter *negative contexts*. While positive contexts are important for determining how to translate a word, it is not trivial to generate more positive contexts, as it requires creating more parallel sentences that actually use the word. By contrast, creating more negative contexts is easy; this is what concatenation does. So

one hypothesis is that concatenation helps by creating more negative contexts to improve the model's ability to attend to positive contexts.

To test this, we modify `RAND` by masking all self-attentions so that, in each concatenated example, each sentence can only attend to itself and not the other sentence. Similarly, in cross-attention, each target sentence can only attend to its corresponding source sentence, not the other one. Table 4.5, row 2 shows that this masking removes a large part of the improvement due to concatenation, showing that the availability of negative contexts during training does help during translation.

4.3.4 Length diversity

The last possible effect of concatenation that we consider is also the most subtle. Following previous work [52, 64], we first sort sentences by length, then splitting into minibatches of a fixed number of tokens. This puts sentences of similar lengths into the same minibatch, which improves computation efficiency as there is less padding. However, as observed by Morishita et al. [52], short and long sentences are qualitatively different, so creating a minibatch of only short sentences or only long sentences approximates the full gradient less well than a minibatch of random sentences would.

With random concatenation, we again put examples of similar lengths into the same minibatch, but each example may consist of two sentences of very different lengths. Thus, it improves diversity within a minibatch while retaining efficiency. We hypothesize that this greater length diversity is part of the reason concatenation helps.

To evaluate this hypothesis, we try a different batch generation strategy from the one described above in Section 4.2.2. In this setup, called **sep-batch**, we make two changes. First, the creation of D_{new} comes after sorting by sentence length (but before division into minibatches), so that in D_{new}, each example comes from two similar-length ones. Second, we create batches from D_{orig} and D_{new} separately so there is no mixture of short sentences in D_{orig} and long sentences in D_{new}.

As we can see in Table 4.5, removing length diversity (**sep-batch**, row 3) causes a small negative impact of −0.2 BLEU. So length diversity may be a contributing factor to concatenation's improvement.

4.3.5 Feature ablation

We have shown that all three hypotheses (position diversity, context diversity, and length diversity) seem to contribute to the BLEU improvement due to concatenation. To see whether these hypotheses exhaustively explain it, we test all three together. First, we apply **mask** and **sep-batch** together, resulting in a drop of −0.7 BLEU (Table 4.5, row 4).

Finally, to remove the effect of position shifting, we additionally reset the positions of the second sentence in every concatenated example so they start at 0 again (**reset-pos**). Applying this on top of **mask** and **sep-batch**, it brings about the largest drop of −0.9 BLEU compared to `RAND`, resulting in a final model that is very close to the baseline. Indeed, this model is only significantly different from the baseline on sk→en ($p < 0.01$). We conclude that these three hypotheses completely account for the improvement due to concatenation.

4.4 Conclusion

Random concatenation is a simple and surprisingly effective data augmentation method for low-resource NMT. Although the improvement of +1 BLEU it yields seems mysterious at first, we have shown that it can be explained by the fact that concatenation increases positions, context, and length diversity. Of these three factors, context diversity seems to be the most important.

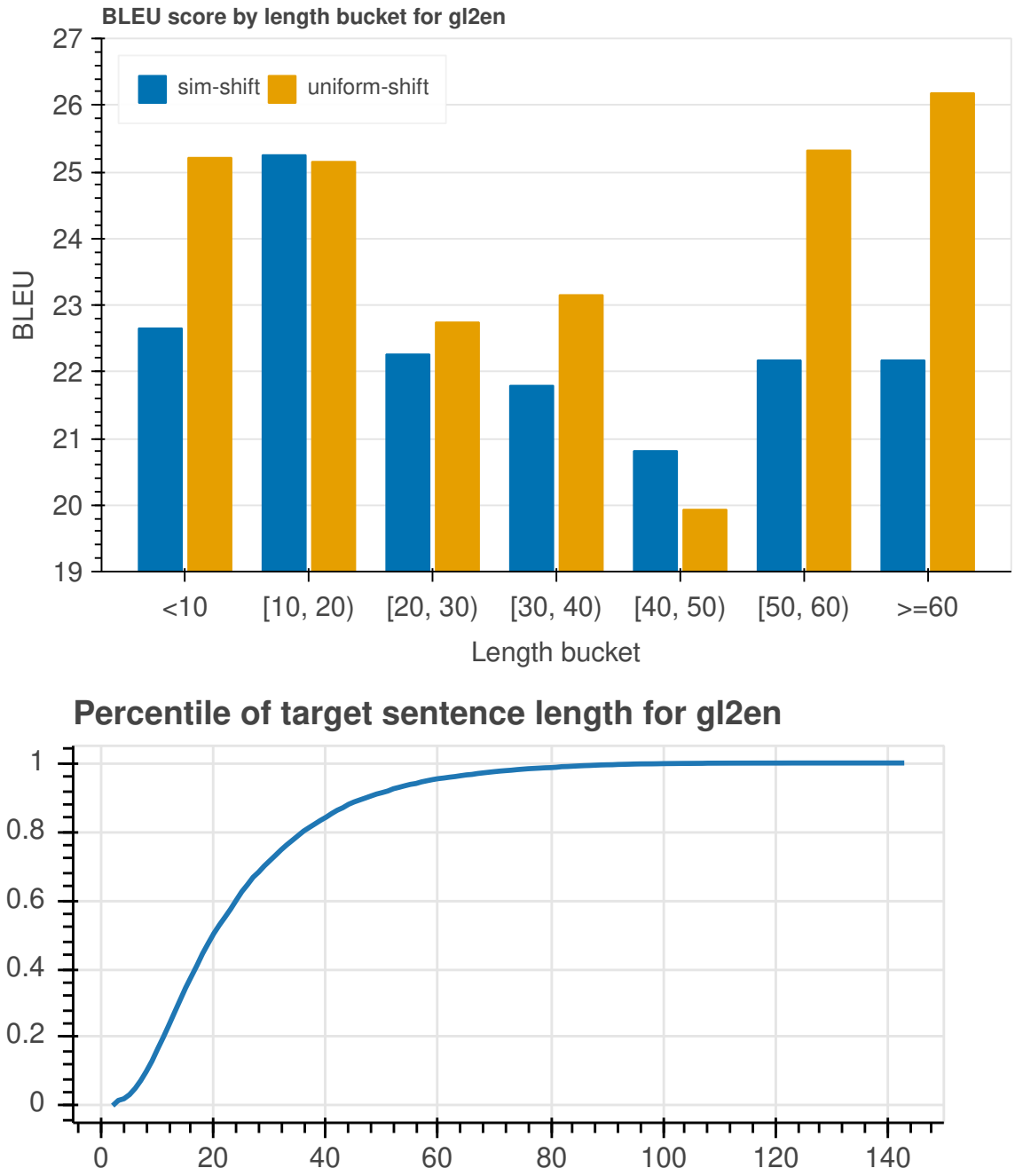


Figure 4.1. gl2en: dev BLEU scores by length bucket (top) and its train length percentile (bottom).

CHAPTER 5

IMPROVING LEXICAL CHOICE IN NEURAL MACHINE TRANSLATION

Previously, we have shown two simple ways to better exploit data to improve NMT performance. In this chapter, we shall look into *better modeling* and *better normalization*. More specifically, we show how simple ℓ_2 normalization can improve rare-word mistranslation in particular and training in general. To further alleviate the rare-word translation issue, we use a simple lexical module which lets model focus more on the source side information when translating. This chapter was published as a long paper at NAACL 2018 [57].

Abstract *We explore two solutions to the problem of mistranslating rare words in neural machine translation. First, we argue that the standard output layer, which computes the inner product of a vector representing the context with all possible output word embeddings, rewards frequent words disproportionately, and we propose to fix the norms of both vectors to a constant value. Second, we integrate a simple lexical module which is jointly trained with the rest of the model. We evaluate our approaches on eight language pairs with data sizes ranging from 100k to 8M words, and achieve improvements of up to +4.3 BLEU, surpassing phrase-based translation in nearly all settings.*

5.1 Introduction

Despite their competitive performance, there are still many open problems in NMT [38]. One particular issue is mistranslation of rare words. For example, consider the Uzbek sentence:

Source: Ammo muammolar hali ko'p, deydi amerikalik olim Entoni Fauchi.

Reference: But still there are many problems, says American scientist Anthony Fauci.

Baseline NMT: But there is still a lot of problems, says James Chan.

At the position where the output should be *Fauci*, the NMT model's top three candidates are *Chan*, *Fauci*, and *Jenner*. All three surnames occur in the training data with reference to immunologists: Fauci is the director of the National Institute of Allergy and Infectious Diseases, Margaret (not James) Chan is the former director of the World Health Organization, and Edward Jenner invented smallpox vaccine. But *Chan* is more frequent in the training data than *Fauci*, and *James* is more frequent than either *Anthony* or *Margaret*.

Because NMT learns word representations in continuous space, it tends to translate words that "seem natural in the context, but do not reflect the content of the source sentence" [4]. This coincides with other observations that NMT's translations are often fluent but lack accuracy [92, 94].

Why does this happen? At each time step, the model's distribution over output words e is

$$p(e) \propto \exp\left(W_e \cdot \tilde{h} + b_e\right)$$

where W_e and b_e are a vector and a scalar depending only on e, and $\tilde{h}$ is a vector depending only on the source sentence and previous output words. We propose two modifications to this layer. First, we argue that the term $W_e \cdot \tilde{h}$, which measures how well e fits into the context $\tilde{h}$, favors common words disproportionately, and show that it helps to fix the norm of both vectors to a constant. Second, we add a new term representing a more direct connection from the source sentence, which allows the model to better memorize translations of rare words.

Below, we describe our models in more detail. Then we evaluate our approaches on eight language pairs, with training data sizes ranging from 100k words to 8M words, and show improvements of up to +4.3 BLEU, surpassing phrase-based translation in nearly all

settings. Finally, we provide some analysis to better understand why our modifications work well[1].

5.2 Neural Machine Translation

Given a source sequence $f = f_1 f_2 \cdots f_m$, the goal of NMT is to find the target sequence $e = e_1 e_2 \cdots e_n$ that maximizes the objective function:

$$\log p(e \mid f) = \sum_{t=1}^{n} \log p(e_t \mid e_{<t}, f).$$

The predicted probability distribution of the t'th target word is:

$$p(e_t \mid e_{<t}, f) = \text{softmax}(W^o \tilde{h}_t + b^o). \tag{5.1}$$

The rows of the output layer's weight matrix W^o can be thought of as embeddings of the output vocabulary, and sometimes are in fact tied to the embeddings in the input layer, reducing model size while often achieving similar performance [28, 69]. We verified this claim on some language pairs and found out that this approach usually performs better than without tying, as seen in Table 5.1. For this reason, we always tie the target embeddings and W^o in all of our models.

5.3 Normalization

The output word distribution (5.1) can be written as:

$$p(e) \propto \exp\left(\|W_e\| \, \|\tilde{h}\| \cos\theta_{W_e,\tilde{h}} + b_e\right),$$

[1]Our code for this work can be found at `https://github.com/tnq177/improving_lexical_choice_in_nmt`

TABLE 5.1

PRELIMINARY DEV BLEU SCORES

	ha-en	tu-en	hu-en
untied embeddings	17.2	11.5	26.5
tied embeddings	17.4	13.8	26.5
don't normalize $\tilde{h}_t$	18.6	14.2	27.1
normalize $\tilde{h}_t$	20.5	16.1	28.8

Preliminary experiments show that tying target embeddings with output layer weights performs as well as or better than the baseline, and that normalizing $\tilde{h}$ is better than not normalizing $\tilde{h}$. All numbers are BLEU scores on development sets, scored against tokenized references.

where W_e is the embedding of e, b_e is the e'th component of the bias b^o, and $\theta_{W_e,\tilde{h}}$ is the angle between W_e and $\tilde{h}$. We can intuitively interpret the terms as follows. The term $\|\tilde{h}\|$ has the effect of sharpening or flattening the distribution, reflecting whether the model is more or less certain in a particular context. The cosine similarity $\cos\theta_{W_e,\tilde{h}}$ measures how well e fits into the context. The bias b_e controls how much the word e is generated; it is analogous to the language model in a log-linear translation model [62].

Finally, $\|W_e\|$ also controls how much e is generated. Figure 5.1 shows that it generally correlates with frequency. But because it is multiplied by $\cos\theta_{W_e,\tilde{h}}$, it has a stronger effect on words whose embeddings have direction similar to $\tilde{h}$, and less effect or even a negative effect on words in other directions. We hypothesize that the result is that the model learns $\|W_e\|$ that are disproportionately large.

For example, returning to the example from Section 5.1, these terms are shown in Table 5.2. Observe that $\cos\theta_{W_e,\tilde{h}}$ and even b_e both favor the correct output word *Fauci*, whereas $\|W_e\|$ favors the more frequent, but incorrect, word *Chan*. The most frequently-mentioned

TABLE 5.2

EXAMPLE DOT PRODUCT TERMS FROM BASELINE

e	$\|\|W_e\|\|$	$\|\|\tilde{h}\|\|$	$\cos\theta_{W_e,\tilde{h}}$	b_e	logit
Chan	5.25	19.5	0.144	−1.53	13.2
Fauci	4.69	19.5	0.154	−1.35	12.8
Jenner	5.23	19.5	0.120	−1.59	10.7

immunologist trumps other immunologists.

To solve this issue, we suggest to fix the norm of all target word embeddings to some value r. Following the weight normalization approach of Salimans and Kingma [73], we reparameterize W_e as $r\frac{\boldsymbol{v}_e}{\|\boldsymbol{v}_e\|}$, but keep r fixed.

A similar argument could be made for $\|\tilde{h}_t\|$: because a large $\|\tilde{h}_t\|$ sharpens the distribution, causing frequent words to more strongly dominate rare words, we might want to limit it as well. We compared both approaches on a development set and found that replacing $\tilde{h}_t$ in equation (5.1) with $r\frac{\tilde{h}_t}{\|\tilde{h}_t\|}$ indeed performs better, as shown in Table 5.1.

5.4 Lexical Translation

The attentional hidden state $\tilde{h}$ contains information not only about the source word(s) corresponding to the current target word, but also the contexts of those source words and the preceding context of the target word. This could make the model prone to generate a target word that fits the context but doesn't necessarily correspond to the source word(s). Count-based statistical models, by contrast, don't have this problem, because they simply don't model any of this context. Arthur et al. [4] try to alleviate this issue by integrating a count-based lexicon into an NMT system. However, this lexicon must be trained separately

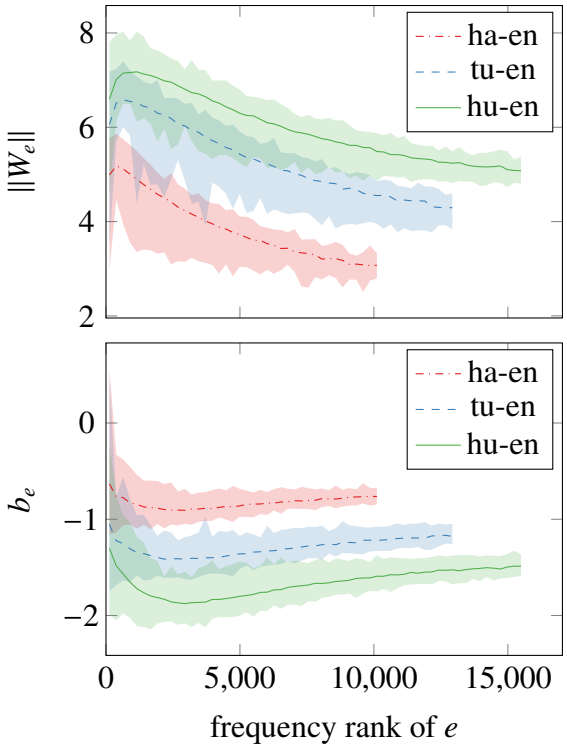


Figure 5.1. The word embedding norm $\|W_e\|$ generally correlates with the frequency of e, except for the most frequent words. The bias b_e has the opposite behavior. The plots show the median and range of bins of size 256.

using GIZA++ [61], and its parameters form a large, sparse array, which can be difficult to store in GPU memory.

Instead, we use a simple feedforward neural network (FFNN) that is trained jointly with the rest of the NMT model to generate a target word based directly on the source word(s). Let f_s ($s = 1, \ldots, m$) be the embeddings of the source words. We use the attention weights to form a weighted average of the embeddings (not the hidden states, as in the main model) to give an average source-word embedding at each decoding time step t:

$$f_t^\ell = \tanh \sum_s a_t(s) f_s.$$

Then we use a one-hidden-layer FFNN with skip connections [24]:

$$h_t^\ell = \tanh(W f_t^\ell) + f_t^\ell$$

and combine its output with the decoder output to get the predictive distribution over output words at time step t:

$$p(y_t \mid y_{<t}, x) = \text{softmax}(W^o \tilde{h}_t + b^o + W^\ell h_t^\ell + b^\ell).$$

For the same reasons that were given in Section 5.3 for normalizing $\tilde{h}_t$ and the rows of W_t^o, we normalize h_t^ℓ and the rows of W^ℓ as well. Note, however, that we do not tie the rows of W^ℓ with the word embeddings; in preliminary experiments, we found this to yield worse results.

5.5 Experiments

We conducted experiments testing our normalization approach and our lexical model on eight language pairs using training data sets of various sizes. This section describes the systems tested and our results.

5.5.1 Data

We evaluated our approaches on various language pairs and datasets:

- Tamil (ta), Urdu (ur), Hausa (ha), Turkish (tu), and Hungarian (hu) to English (en), using data from the LORELEI program.
- English to Vietnamese (vi), using data from the IWSLT 2015 shared task.[2]
- To compare our approach with that of Arthur et al. [4], we also ran on their English to Japanese (ja) KFTT and BTEC datasets.[3]

[2] `https://nlp.stanford.edu/projects/nmt/`

[3] `http://isw3.naist.jp/~philip-a/emnlp2016/`

TABLE 5.3

DATA AND MODEL STATISTICS

	tokens $\times 10^6$	vocab $\times 10^3$	layers num/size
ta-en	0.2/0.1	4.0/3.4	1/512
ur-en	0.2/0.2	4.2/4.2	1/512
ha-en	0.8/0.8	10.6/10.4	2/512
tu-en	0.8/1.1	21.1/13.3	2/512
uz-en	1.5/1.9	29.8/17.4	2/512
hu-en	2.0/2.3	27.3/15.7	2/512
en-vi	2.1/2.6	17.0/7.7	2/512
en-ja (BTEC)	3.6/5.0	17.8/21.8	4/768
en-ja (KFTT)	7.8/8.0	48.2/49.1	4/768

Statistics of data and models: effective number of training source/target tokens, source/target vocabulary sizes, number of hidden layers and number of units per layer.

We tokenized the LORELEI datasets using the default Moses tokenizer, except for Urdu-English, where the Urdu side happened to be tokenized using Morfessor FlatCat ($w = 0.5$). We used the preprocessed English-Vietnamese and English-Japanese datasets as distributed by Luong et al., and Arthur et al., respectively. Statistics about our data sets are shown in Table 5.3.

5.5.2 Systems

We compared our approaches against two baseline NMT systems:

untied, which does not tie the rows of W_o to the target word embeddings, and

tied, which does.

In addition, we compared against two other baseline systems:

Moses: The Moses phrase-based translation system [39], trained on the same data as the NMT systems, with the same maximum sentence length of 50. No additional data was used for training the language model. Unlike the NMT systems, Moses used the full vocabulary from the training data; unknown words were copied to the target sentence.

Arthur: Our reimplementation of the discrete lexicon approach of Arthur et al. [4]. We only tried their `auto` lexicon, using GIZA++ [61], integrated using their `bias` approach. Note that we also tied embedding as we found it also helped in this case.

Against these baselines, we compared our new systems:

fixnorm: The normalization approach described in Section 5.3.

fixnorm+lex: The same, with the addition of the lexical translation module from Section 5.4.

5.5.3 Details

Model We use the *global attentional* model with *general scoring function* and *input feeding* by Luong et al. [49]. Following their practice, we fed the source sentences to the encoder in reverse order during both training and testing. Information about the number and size of hidden layers is shown in Table 5.3. The word embedding size is always equal to the hidden layer size.

Following common practice, we only trained on sentences of 50 tokens or less. We limited the vocabulary to word types that appear no less than 5 times in the training data and map the rest to `UNK`. For the English-Japanese and English-Vietnamese datasets, we used the vocabulary sizes reported in their respective papers [4, 47].

For **fixnorm**, we tried $r \in \{3, 5, 7\}$ and selected the best value based on the development set performance, which was $r = 5$ except for English-Japanese (BTEC), where $r = 7$. For

fixnorm+lex, because $W_s\tilde{h}_t + W^\ell h_t^\ell$ takes on values in $[-2r^2, 2r^2]$, we reduced our candidate r values by roughly a factor of $\sqrt{2}$, to $r \in \{2, 3.5, 5\}$. A radius $r = 3.5$ seemed to work the best for all language pairs.

Training We trained all NMT systems with Adadelta [96]. All parameters were initialized uniformly from $[-0.01, 0.01]$. When a gradient's norm exceeded 5, we normalized it to 5. We also used dropout on non-recurrent connections only [95], with probability 0.2. We used minibatches of size 32. We trained for 50 epochs, validating on the development set after every epoch, except on English-Japanese, where we validated twice per epoch. We kept the best checkpoint according to its BLEU on the development set.

Inference We used beam search with a beam size of 12 for translating both the development and test sets. Since NMT often favors short translations [11], we followed Wu et al. [94] in using a modified score $s(e \mid f)$ in place of log-probability:

$$s(e \mid f) = \frac{\log p(e \mid f)}{lp(e)}$$
$$lp(e) = \frac{(5 + |e|)^\alpha}{(5 + 1)^\alpha}$$

We set $\alpha = 0.8$ for all of our experiments.

Finally, we applied a postprocessing step to replace each `UNK` in the target translation with the source word with the highest attention score [50].

Evaluation For translation into English, we report case-sensitive NIST BLEU against detokenized references. For English-Japanese and English-Vietnamese, we report tokenized, case-sensitive BLEU following Arthur et al. [4] and Luong and Manning [47]. We measure statistical significance using bootstrap resampling [37].

TABLE 5.4

TEST BLEU SCORES

	untied	**tied**	**fixnorm**	**fixnorm+lex**	**Moses**	**Arthur**
ta-en	10.3	11.1	14 (+2.9)	15.3 (+4.2)	10.5 (−0.6)	14.1 (+3.0)
ur-en	7.9	10.7	12 (+1.3)	13 (+2.3)	14.6 (+3.9)	12.5 (+1.8)
ha-en	16.0	16.6	20 (+3.4)	21.5 (+4.9)	22.2 (+5.6)	18.7 (+2.1)
tu-en	12.2	12.6	16.4 (+3.8)	19.1 (+6.5)	18.1 (+5.5)	16.3 (+3.7)
uz-en	14.9	15.7	18.2 (+2.5)	19.3 (+3.6)	17.2 (+1.5)	17.1 (+1.4)
hu-en	21.6	23.0	24.0 (+1.0)	25.3 (+2.3)	21.3 (−1.7)	22.7 (-0.3)†
en-vi	25.1	25.3	26.8 (+1.5)	27 (+1.7)	26.7 (+1.4)	26.2 (+0.9)
en-ja (BTEC)	51.2	53.7	52.9 (-0.8)†	51.3 (−2.6)†	46.8 (−6.9)	52.4 (−1.3)†
en-ja (KFTT)	24.1	24.5	26.1 (+1.6)	26.2 (+1.7)	21.7 (−2.8)	—

Test BLEU of all models. Differences shown in parentheses are relative to **tied**, with a dagger (†) indicating an *insignificant* difference in BLEU ($p > 0.01$). While the method of Arthur et al. [4] does not always help, **fixnorm** and **fixnorm+lex** consistently achieve significant improvements over **tied** ($p < 0.01$) except for English-Japanese (BTEC). Our models also outperform the method of Arthur et al. on all tasks and outperform Moses on all tasks but Urdu-English and Hausa-English.

5.6 Results and Analysis

5.6.1 Overall

Our results are shown in Table 5.4. First, we observe, as has often been noted in the literature, that NMT tends to perform poorer than PBMT on low resource settings (note that the rows of this table are sorted by training data size).

Our **fixnorm** system alone shows large improvements (shown in parentheses) relative to **tied**. Integrating the lexical module (**fixnorm+lex**) adds in further gains. Our **fixnorm+lex** models surpass Moses on all tasks except Urdu- and Hausa-English, where it is 1.6 and 0.7

BLEU short respectively.

The method of Arthur et al. [4] does improve over the baseline NMT on most language pairs, but not by as much and as consistently as our models, and often not as well as Moses. Unfortunately, we could not replicate their approach for English-Japanese (KFTT) because the lexical table was too large to fit into the computational graph.

For English-Japanese (BTEC), we note that, due to the small size of the test set, all systems except for Moses are in fact not significantly different from **tied** ($p > 0.01$). On all other tasks, however, our systems significantly improve over **tied** ($p < 0.01$).

5.6.2 Impact on translation

In Table 5.5, we show examples of typical translation mistakes made by the baseline NMT systems. In the Uzbek example (top), **untied** and **tied** have confused *34* with `UNK` and *700*, while in the Turkish one (middle), they incorrectly output other proper names, *Afghan* and *Myanmar*, for the proper name *Kenya*. Our systems, on the other hand, translate these words correctly.

The bottom example is the one introduced in Section 5.1. We can see that our **fixnorm** approach does not completely solve the mistranslation issue, since it translates *Entoni Fauchi* to `UNK UNK` (which is arguably better than *James Chan*). On the other hand, **fixnorm+lex** gets this right. To better understand how the lexical module helps in this case, we look at the top five translations for the word *Fauci* in **fixnorm+lex** in Table 5.6. As we can see, while $\cos\theta_{W_e,\tilde{h}}$ might still be confused between similar words, $\cos\theta_{W^l_e,h_l}$ significantly favors *Fauci*.

5.6.3 Alignment and unknown words

Both our baseline NMT and **fixnorm** models suffer from the problem of shifted alignments noted by Koehn and Knowles [38]. As seen in Figure 5.2a and 5.2b, the alignments for those two systems seem to shift by one word to the left (on the source side). For ex-

ample, *nói* should be aligned to *said* instead of *Telekom*, and so on. Although this is not a problem *per se*, since the decoder can decide to attend to any position in the encoder states as long as the state at that position holds the information the decoder needs, this becomes a real issue when we need to make use of the alignment information, as in unknown word replacement [50]. As we can see in Figure 5.2, because of the alignment shift, both **tied** and **fixnorm** incorrectly replace the two unknown words (in bold) with *But Deutsche* instead of *Deutsche Telekom*. In contrast, under **fixnorm+lex** and the model of Arthur et al. [4], the alignment is corrected, causing the `UNK`s to be replaced with the correct source words.

5.6.4 Impact of r

The single most important hyper-parameter in our models is r. Informally speaking, r controls how much surface area we have on the hypersphere to allocate to word embeddings. To better understand its impact, we look at the training perplexity and dev BLEUs during training with different values of r. Table 5.7 shows the train perplexity and best tokenized dev BLEU on Turkish-English for **fixnorm** and **fixnorm+lex** with different values of r.

As we can see, a smaller r results in worse training perplexity, indicating underfitting, whereas if r is too large, the model achieves better training perplexity but decrased dev BLEU, indicating overfitting.

5.6.5 Lexicon

One byproduct of **lex** is the lexicon, which we can extract and examine simply by feeding each source word embedding to the FFNN module and calculating $p^\ell(y) = \text{softmax}(W^\ell h^\ell + b_\ell)$. In Table 5.8, we show the top translations for some entries in the lexicons extracted from **fixnorm+lex** for Hungarian, Turkish, and Hausa-English. As expected, the lexical distribution is sparse, with a few top translations accounting for the most probability mass.

5.6.6 Byte Pair Encoding

Byte-Pair-Encoding (BPE) [78] is commonly used in NMT to break words into word-pieces, improving the translation of rare words. For this reason, we reran our experiments using BPE on the LORELEI and English-Vietnamese datasets. Additionally, to see if our methods work in high-resource scenarios, we run on the WMT 2014 English-German (en-de) dataset,[4] using *newstest2013* as the development set and reporting tokenized, case-sensitive BLEU on *newstest2014* and *newstest2015*.

We validate across different numbers of BPE operations; specifically, we try {1k, 2k, 3k} merge operations for ta-en and ur-en due to their small sizes, {10k, 12k, 15k} for the other LORELEI datasets and en-vi, and 32k for en-de. Using BPE results in much smaller vocabulary sizes, so we do not apply a vocabulary cut-off. Instead, we train on an additional copy of the training data in which all types that appear once are replaced with `UNK`, and halve the number of epochs accordingly. Our models, training, and evaluation processes are largely the same, except that for en-de, we use a 4-layer decoder and 4-layer bidirectional encoder (2 layers for each direction).

Table 5.9 shows that our methods also significantly improve the translation when used with BPE, for both high and low resource language pairs. With BPE, we are only behind Moses on Urdu-English.

5.7 Related Work

The closest work to our **lex** model is that of Arthur et al. [4], which we have discussed already in Section 5.4. Recent work by Liu et al. [45] has very similar motivation to that of our **fixnorm** model. They reformulate the output layer in terms of directions and magnitudes, as we do here. Whereas we have focused on the magnitudes, they focus on the directions, modifying the loss function to try to learn a classifier that separates the classes'

[4] `https://nlp.stanford.edu/projects/nmt/`

directions with something like a margin. Wang et al. [90] also make the same observation that we do for the **fixnorm** model, but for the task of face verification.

Handling rare words is an important problem for NMT that has been approached in various ways. Some have focused on reducing the number of UNKs by enabling NMT to learn from a larger vocabulary [30, 51]; others have focused on replacing UNKs by copying source words [21, 20, 50]. However, these methods only help with unknown words, not rare words. An approach that addresses both unknown and rare words is to use subword-level information [78, 13, 48]. Our approach is different in that we try to identify and address the root of the rare word problem. We expect that our models would benefit from more advanced UNK-replacement or subword-level techniques as well.

Recently, Liu and Kirchhoff [44] have shown that their baseline NMT system with BPE already outperforms Moses for low-resource translation. However, in their work, they use the Transformer network [87], which is quite different from our baseline model. It would be interesting to see if our methods benefit the Transformer network and other models as well.

5.8 Conclusion

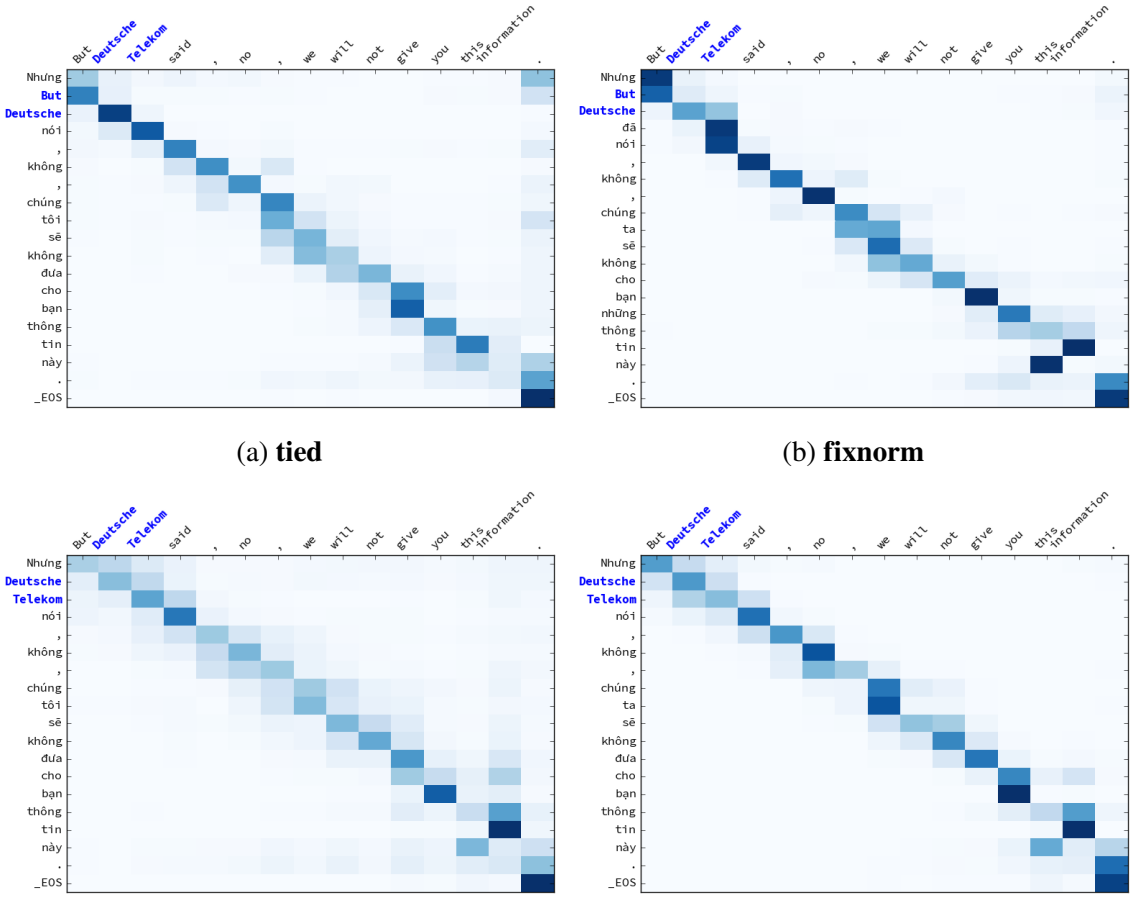
In this section, we have presented two simple yet effective changes to the output layer of a NMT model. Both of these changes improve translation quality substantially on low-resource language pairs. In many of the language pairs we tested, the baseline NMT system performs poorly relative to phrase-based translation, but our system surpasses it (when both are trained on the same data). We conclude that NMT, equipped with the methods demonstrated here, is a more viable choice for low-resource translation than before, and are optimistic that NMT's repertoire will continue to grow.

TABLE 5.5

EXAMPLE TRANSLATIONS

input	Dushanba kuni Hindistonda kamida **34** kishi halok bo'lgani xabar qilindi .
reference	At least **34** more deaths were reported Monday in India .
untied	At least **UNK** people have died in India on Monday .
tied	It was reported that at least **700** people died in Monday .
fixnorm	At least **34** people died in India on Monday .
fixnorm+lex	At least **34** people have died in India on Monday .
input	Yarın **Kenya'da** bir yardım konferansı düzenlenecek .
reference	Tomorrow a conference for aid will be conducted in **Kenya** .
untied	Tomorrow there will be an **Afghan** relief conference .
tied	Tomorrow there will be a relief conference in **Myanmar** .
fixnorm	Tomorrow it will be a aid conference in **Kenya** .
fixnorm+lex	Tomorrow there will be a relief conference in **Kenya** .
input	Ammo muammolar hali ko'p , deydi amerikalik olim **Entoni Fauchi** .
reference	But still there are many problems , says American scientist **Anthony Fauci** .
untied	But there is still a lot of problems , says **James Chan** .
tied	However , there is still a lot of problems , says American scientists .
fixnorm	But there is still a lot of problems , says American scientist **UNK UNK** .
fixnorm+lex	But there are still problems , says American scientist **Anthony Fauci** .

Example translations, in which **untied** and **tied** generate incorrect, but often semantically related, words, but **fixnorm** and/or **fixnorm+lex** generate the correct ones.



(a) **tied** (b) **fixnorm**

(c) **fixnorm+lex** (d) Arthur et al. [4]

Figure 5.2. While the **tied** and **fixnorm** systems shift attention to the left one word (on the source side), our **fixnorm+lex** model and that of Arthur et al. [4] put it back to the correct position, improving unknown-word replacement for the words *Deutsche Telekom*. Columns are source (English) words and rows are target (Vietnamese) words. Bolded words are unknown.

TABLE 5.6

EXAMPLE DOT PRODUCT TERMS FROM **FIXNORM+LEX**

e	$\cos\theta_{W_e,\tilde{h}}$	$\cos\theta_{W_e^l,h_l}$	$b_e + b_e^l$	logit
Fauci	0.522	0.762	−8.71	7.0
UNK	0.566	−0.009	−1.25	5.6
Anthony	0.263	0.644	−8.70	2.4
Ahmedova	0.555	0.173	−8.66	0.3
Chan	0.546	0.150	−8.73	−0.2

TABLE 5.7

IMPACT OF R

system	r	train ppl	dev BLEU
fixnorm	3	3.9	13.6
	5	2.5	16.1
	7	2.3	14.4
fixnorm+lex	2	4.2	12.3
	3.5	2.0	17.5
	5	1.4	16.0

When r is too small, high train perplexity and low dev BLEU indicate underfitting; when r is too large, low train perplexity and low dev BLEU indicate overfitting.

TABLE 5.8

EXTRACTED LEXICON EXAMPLES

hu-en	244	244 (0.599) document (0.005) By (0.003)
	befektetéseinek	investments (0.151) investment (0.017) Investments (0.015)
	kutatás-fejlesztésre	research (0.227) Research (0.040) Development (0.014)
tu-en	ifade	expression (0.109) expressed (0.061) express (0.056)
	cumhurbaşkanı	President (0.573) president (0.030) Republic (0.027)
	Göstericiler	protesters (0.115) demonstrators (0.050) Protesters (0.033)
ha-en	😒	😒 (0.469) cholera (0.003) EOS (0.001)
	Wayoyin	phones (0.414) wires (0.097) mobile (0.088)
	manzonsa	Prophet (0.080) His (0.041) Messenger (0.015)

Top five translations for some entries of the lexical tables extracted from **fixnorm+lex**. Probabilities are shown in parentheses.

TABLE 5.9

TEST BLEU SCORES FOR BPE-BASED SYSTEMS

	tied	**fixnorm**	**fixnorm+lex**	
ta-en	13	15 (+2.0)	15.9	(+2.9)
ur-en	10.5	12.3 (+1.8)	13.7	(+3.2)
ha-en	18	21.7 (+3.7)	22.3	(+4.3)
tu-en	19.3	21 (+1.7)	22.2	(+2.9)
uz-en	18.9	19.8 (+0.9)	21	(+2.1)
hu-en	25.8	27.2 (+1.4)	27.9	(+2.1)
en-vi	26.3	27.3 (+1.0)	27.5	(+1.2)
en-de (newstest2014)	19.7	22.2 (+2.5)	20.4	(+0.7)
en-de (newstest2015)	22.5	25 (+2.5)	23.2	(+0.7)

Our models significantly improve over the baseline ($p < 0.01$) for both high and low resource when using BPE.

CHAPTER 6

TRANSFORMERS WITHOUT TEARS: IMPROVING THE NORMALIZATION OF SELF-ATTENTION

In the previous chapter, we have shown how to improve NMT performance via better modeling and better normalization for RNN-based NMT. Recently, Transformer has replaced RNN to be the *de facto* model choice for many sequence-to-sequence problems. Despite its impressive performance, Transformer is known to be unstable to train and other works that use it often focus on high-resource settings only. In this chapter, we will show how a simple rearrangement of Transformer components can improve its training (**better modeling**). We also apply our normalization methods in chapter 5 and show how they can enhance Transformer performance for low-resource languages (**better normalization**). This chapter appeared as a publication at IWSLT 2019 [59].

Abstract *We evaluate three simple, normalization-centric changes to improve Transformer training. First, we show that pre-norm residual connections (*PRENORM*) and smaller initializations enable warmup-free, validation-based training with large learning rates. Second, we propose ℓ_2 normalization with a single scale parameter (*SCALENORM*) for faster training and better performance. Finally, we reaffirm the effectiveness of normalizing word embeddings to a fixed length (*FIXNORM*). On five low-resource translation pairs from TED Talks-based corpora, these changes always converge, giving an average +1.1 BLEU over state-of-the-art bilingual baselines and a new 32.8 BLEU on IWSLT'15 English-Vietnamese. We observe sharper performance curves, more consistent gradient norms, and a linear relationship between activation scaling and decoder depth. Surprisingly, in*

the high-resource setting (WMT'14 English-German), SCALENORM *and* FIXNORM *remain competitive but* PRENORM *degrades performance.*

6.1 Introduction

Transformer [87] has become the dominant architecture for neural machine translation (NMT) due to its train-time parallelism and strong downstream performance. Related work [22, 81] has investigated various modifications to improve the efficiency of its multi-head attention and feedforward sublayers. Our work focuses on *layer normalization* (LAYERNORM) [5], which we show has an outsized role in the convergence and performance of the Transformer in two ways:

Placement of normalization. The original Transformer uses *post-norm residual units* (POSTNORM), where layer normalization occurs after the sublayer and residual addition. However, Chen et al. [10] found that *pre-norm residual units* (PRENORM), where layer normalization occurs immediately before the sublayer, were instrumental to their model's performance. Wang et al. [91] compare the two, showing that PRENORM makes backpropagation more efficient over depth and training Transformers with deep, 30-layer encoders.

Our work demonstrates additional consequences in the base (≤6-layer encoder) Transformer regime. We show that PRENORM enables warmup-free, validation-based training with large learning rates even for small batches, in contrast to past work on scaling NMT [63]. We also partly reclaim POSTNORM's stability via smaller initializations, although PRENORM is less sensitive to this magnitude and can improve performance. However, despite PRENORM's recent adoption in many NMT frameworks, we find it degrades base Transformer performance on WMT'14 English-German.

Choice of normalization. Santurkar et al. [74] show that batch normalization's effectiveness is not from reducing internal covariate shift, but from smoothing the loss landscape.

They achieve similar or better performance with non-variance-based normalizations in image classification. Hence, we propose replacing LAYERNORM with the simpler *scaled* ℓ_2 *normalization* (SCALENORM), which normalizes activation vectors to a *single* learned length g. This is both inspired by and synergistic with jointly fixing the word embedding lengths (FIXNORM) [57]. These changes improve the training speed and low-resource performance of the Transformer without affecting high-resource performance.

On five low-resource pairs from the TED Talks [70] and IWSLT '15 [9] corpora, we first train state-of-the-art Transformer models (+4.0 BLEU on average over the best published NMT bitext-only numbers). We then apply PRENORM, FIXNORM, and SCALENORM for an average total improvement of +1.1 BLEU, where each addition contributes at least +0.3 BLEU (Section 6.3), and attain a new 32.8 BLEU on IWSLT '15 English-Vietnamese. We validate our intuitions in Section 6.4 by showing sharper performance curves (i.e., improvements occur at earlier epochs) and more consistent gradient norms. We also examine the per-sublayer g's learned by SCALENORM, which suggest future study[1].

6.2 Background

6.2.1 Identity mappings for transformers

Residual connections [24] were first introduced to facilitate the training of deep convolutional networks, where the output of the ℓ-th layer F_ℓ is summed with its input:

$$\boldsymbol{x}_{\ell+1} = \boldsymbol{x}_\ell + F_\ell(\boldsymbol{x}_\ell). \tag{6.1}$$

The identity term $\boldsymbol{x}_\ell$ is crucial to greatly extending the depth of such networks [25]. If one were to scale $\boldsymbol{x}_\ell$ by a scalar λ_ℓ, then the contribution of $\boldsymbol{x}_\ell$ to the final layer F_L is $(\prod_{i=\ell}^{L-1} \lambda_i)\boldsymbol{x}_\ell$. For deep networks with dozens or even hundreds of layers L, the term $\prod_{i=\ell}^{L-1} \lambda_i$

[1] Reimplementation available at `https://github.com/tnq177/transformers_without_tears`

becomes very large if $\lambda_i > 1$ or very small if $\lambda_i < 1$, for enough i. When backpropagating from the last layer L back to ℓ, these multiplicative terms can cause exploding or vanishing gradients, respectively. Therefore they fix $\lambda_i = 1$, keeping the total residual path an identity map.

The original Transformer applies LAYERNORM after the sublayer and residual addition (POSTNORM):

$$\boldsymbol{x}_{\ell+1} = \text{LAYERNORM}(\boldsymbol{x}_\ell + F_\ell(\boldsymbol{x}_\ell)). \tag{6.2}$$

We conjecture this has caused past convergence failures [68, 79], with LAYERNORMs in the residual path acting similarly to $\lambda_i \neq 1$; furthermore, warmup was needed to let LAYERNORM safely adjust scale during early parts of training. Inspired by He et al. [25], we apply LAYERNORM immediately before each sublayer (PRENORM):

$$\boldsymbol{x}_{\ell+1} = \boldsymbol{x}_\ell + F_\ell(\text{LAYERNORM}(\boldsymbol{x}_\ell)). \tag{6.3}$$

This is cited as a stabilizer for Transformer training [10, 91] and is already implemented in popular toolkits [88, 64, 26], though not necessarily used by their default recipes. Wang et al. [91] make a similar argument to motivate the success of PRENORM in training very deep Transformers. Note that one must append an additional normalization after both encoder and decoder so their outputs are appropriately scaled. We compare POSTNORM and PRENORM throughout Section 6.3.

6.2.2 Weight initialization

Xavier normal initialization [18] initializes a layer's weights $\boldsymbol{W}_\ell \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ (d_ℓ is the hidden dimension) with samples from a centered normal distribution with layer-dependent variance:

$$(\boldsymbol{W}_\ell)_{i,j} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{d_\ell + d_{\ell+1}}}\right). \tag{6.4}$$

Our experiments with this default initializer find that PostNorm sometimes fails to converge, especially in our low-resource setting, even with a large number of warmup steps. One explanation is that Xavier normal yields initial weights that are too large. In implementations of the Transformer, one scales the word embeddings by a large value (e.g., $\sqrt{d} \approx 22.6$ for $d = 512$), giving vectors with an expected square norm of d. LayerNorm's unit scale at initialization preserves this same effect. Since feedforward layers already have their weights initialized to a smaller standard deviation, i.e., $\sqrt{\frac{2}{d+4d}}$, we choose to reduce the attention layers' initializations from $\sqrt{\frac{2}{d+d}}$ to $\sqrt{\frac{2}{d+4d}}$ as well (SmallInit), as a corresponding mitigation. We evaluate the effect of this on PostNorm vs. PreNorm in Section 6.3.2.

6.2.3 Scaled ℓ_2 normalization and FixNorm

LayerNorm is inspired by batch normalization [29], both of which aim to reduce internal covariate shift by fixing the mean and variance of activation distributions. Both have been applied to self-attention [87, 41]. However, Santurkar et al. [74] show that batch normalization's success has little to do with covariate shift, but comes instead from smoothing the loss landscape. For example, they divide by the pre-centered ℓ_p norm instead of the variance and achieve similar or better results in image classification.

Hence, we propose replacing LayerNorm with *scaled ℓ_2 normalization*:

$$\textsc{ScaleNorm}(\boldsymbol{x}; g) = g\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}. \tag{6.5}$$

This can be viewed as projecting d-dimensional vectors onto a $(d-1)$-dimensional hypersphere with learned radius g. This expresses the inductive bias that each sublayer's activations has an ideal "global scale," a notion we empirically validate in Section 6.4.2. ScaleNorm replaces the $2d$ scale and shift parameters of LayerNorm with a single learned scalar, improving computational and parameter efficiency while potentially regu-

TABLE 6.1

DATA AND MODEL PROPERTIES

	# egs.	# src + tgt toks.	# iters/epoch	max epoch	# enc/dec layers	# heads/layer	dropout	# BPE
gl→en	10k	0.37M	100	1000	4	4	0.4	3k
sk→en	61k	2.32M	600	200	6	8	0.3	8k
en→vi	133k	5.99M	1500	200	6	8	0.3	8k
en→he	212k	7.88M	2000	200	6	8	0.3	8k
ar→en	214k	8.09M	2000	200	6	8	0.3	8k

larizing the loss landscape.

This bias has an explicit interpretation at the final layer: large inner products sharpen the output distribution, causing frequent words to disproportionately dominate rare words. This led Nguyen and Chiang [57] to introduce $\textsc{FixNorm}(\boldsymbol{w}) = g\frac{\boldsymbol{w}}{||\boldsymbol{w}||}$ with fixed g at the last linear layer, to maximize the angular difference of output representations and aid rare word translation. By making g learnable, we can apply ScaleNorm and FixNorm jointly, which means applying the following at the final linear layer:

$$\begin{aligned}(\textsc{ScaleNorm+FixNorm})(\boldsymbol{x}, \boldsymbol{w}; g) \\ = g\frac{\boldsymbol{w} \cdot \boldsymbol{x}}{||\boldsymbol{w}||||\boldsymbol{x}||}.\end{aligned} \tag{6.6}$$

Note that this combination at the last layer is equivalent to cosine normalization [46] with a learned scale.

6.2.4 Learning rates

Despite using an adaptive optimizer, Adam [36], Transformer training uses a learning rate (LR) schedule with a linear *warmup* and an inverse square root *decay* (InvSqrtDecay):

$$\text{LR}(n) = \frac{\lambda}{\sqrt{d}} \min\left(\frac{1}{\sqrt{n}}, \frac{n}{n_{\text{warmup}}^{1.5}}\right), \tag{6.7}$$

where d is the hidden dimension of the self-attention layers, and λ, n_{warmup} are hyperparameters that determine the highest learning rate achieved and the number of steps to reach it, respectively. These two hyperparameters have been the subject of much empirical study [68, 63]. In light of our modifications however, we revisit various aspects of this schedule:

Warmup-free training. We conjectured that warmup is primarily needed when using PostNorm to gradually learn LayerNorm parameters without gradient explosion/vanishing (Section 6.2.1). Hence, we evaluate both PreNorm and PostNorm without warmup in Section 6.3.4.

Large learning rates. To speed up training, one often explores using larger learning rates. In the context of Transformer, Ott et al. [63] and Aharoni et al. [1] take $\lambda \in \{2, 3\}$ instead of the conventional $\lambda = 1$. Ott et al. [63] showed that one can scale up Adam's learning rate to 10^{-3} with an extremely large batch (400k tokens). However, the improved convergence provided by our modifications could enable higher learning rates with much small batch sizes (4k tokens), as examined in Section 6.3.4.

Validation-based decay. For similar reasons, one might wish to adopt a classic validation-based decay, i.e., training at a high learning rate for as long as tenable, decaying rapidly when development scores flatline. This has inspired usage of fixed decay schemes upon convergence with InvSqrtDecay [14, 72]. We revisit ValDecay under our modifications, where we still perform a linear warmup but then multiply by a scale $\alpha_{\text{decay}} < 1$ when performance on a development set does not improve over *patience* evaluations.

6.3 Experiments and Results

We train Transformer models for a diverse set of five low-resource translation pairs from the TED Talks [70] and the IWSLT'15 [9] corpora. Details are summarized in Table 6.1.

6.3.1 Training details

Data and preprocessing. The pairs are English (en) to Hebrew (he), Vietnamese (vi), and Galician (gl), Slovak (sk), Arabic (ar) to English (en). Because the data is already preprocessed, we only apply BPE [78] with `fastBPE`[2]. Depending on the data size, we use different numbers of BPE operations.

We wanted to compare with the latest low-resource works of [55, 1] on the TED Talks corpus [70]. In particular, Aharoni et al. [1] identified 4 very low-resource pairs (<70k); we took the two (gl→en, sk→en) that were not extremely low (≤6k). They then identified 4 low-resource pairs with 100k-300k examples; we took the top two (ar→en, en→he). To introduce a second English-source pair and to showcase on a well-understood task, we used the en→vi pair from IWSLT'15 with an in-between number of examples (133k). In this way, we have examples of different resource levels, language families, writing directions, and English-source versus -target.

Model configuration. We set the hidden dimension of the feedforward sublayer to 2048 and the rest to 512, matching Vaswani et al. [87]. We use the same dropout rate for output of sublayers, ReLU, and attention weights. Additionally, we also do word dropout [77] with probability 0.1. However, instead of zeroing the word embeddings, we randomly replace tokens with `UNK`. For all experiments, we use label smoothing of 0.1 [84, 67]. The source and target's input and output embeddings are shared [69], but we mask out words that are

[2] `https://github.com/glample/fastBPE`

not in the target's vocabulary at the final output layer before softmax, by setting their logits to $-\infty$.

Training. We use a batch size of 4096 and optimize using Adam [36] with the default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Gradients are clipped when global norm exceeds 1.0 [66]. An epoch is a predefined number of iterations for each pair. We stop training when a maximum number of epochs has been met or the learning rate becomes too small (10^{-6}). We also do early stopping when the development BLEU has not improved for 20 evaluations. For gl→en, this number is 50. When doing validation-based decay, we use $\alpha_{decay} = 0.8$ and $patience = 3$. For complete data and model statistics, please refer to Table 6.1. The best checkpoint is selected based on the development BLEU score during training.

Evaluation. We report tokenized BLEU [65] with `multi-bleu.perl` to be comparable with previous works. We also measure statistical significance using bootstrap resampling [37]. For WMT'14 English-German, note that one needs to put compounds in ATAT format[3] before calculating BLEU score to be comparable with previous works.

6.3.2 Large vs. small initialization

To see the impact of weight initialization, we run training on the en→vi dataset using warmup steps of 4k, 8k, 16k (Table 6.2). With default initialization, POSTNORM fails to converge on this dataset even with a long warmup of 16k steps, only reaching 5.76 BLEU.

The second row shows that taking a smaller standard deviation on the attention weights (SMALLINIT) restores convergence to POSTNORM. Though the $\sqrt{2/5} \approx 0.63$ adjustment

[3] `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/get_ende_bleu.sh`

TABLE 6.2

DEV BLEU SCORES ON EN→VI USING XAVIER NORMAL INITIALIZATION

Xavier normal		# warmup steps		
		4k	8k	16k
Baseline	PostNorm	fail	fail	5.76
	PreNorm	28.52	28.73	28.32
SmallInit	PostNorm	28.17	28.20	28.62
	PreNorm	28.26	28.44	28.33

used here seems marginal, operations like residual connections and the products between queries and keys can compound differences in scale. Though both models now achieve similar performance, we note that PreNorm works in all setups, suggesting greater stability during training. For all remaining experiments, we use PostNorm and PreNorm with SmallInit. We find this choice does not affect the performance of PreNorm.

6.3.3 Scaled ℓ_2 normalization and FixNorm

To compare ScaleNorm and LayerNorm, we take 8k warmup steps for all further experiments. Since we tie the target input word embedding and the last linear layer's weight (Section 6.3.1), FixNorm is implemented by applying ℓ_2 normalization to the word embedding, with each component initialized uniformly in $[-0.01, 0.01]$. For non-FixNorm models, word embeddings are initialized with mean 0 and standard deviation $\sqrt{1/d}$ so they sum to unit variance. All g's in ScaleNorm are initialized to $\sqrt{d}$.

Table 6.3 shows our results along with some published baselines. First, note that our Transformer baselines with PostNorm + LayerNorm (1) are very strong non-

TABLE 6.3

TEST BLEU SCORES

	gl→en	**sk→en**	**en→vi**	**en→he**	**ar→en**	**avg Δ**
POSTNORM + LAYERNORM (published)	16.2	24.0	29.09	23.66	27.84	-4.05
POSTNORM + LAYERNORM (1)	18.47	29.37	31.94	27.85	33.39	+0.00
PRENORM + LAYERNORM (2)	19.09	29.45	31.92	28.13	33.79	+0.27
PRENORM + **fixnorm** + LAYERNORM (3)	19.38	29.50	32.45	28.39	$34.35^{\dagger}$	+0.61
PRENORM + **fixnorm** + SCALENORM (4)	$20.91^{\ddagger *}$	$30.25^{\ddagger *}$	32.79^{*}	28.44^{*}	34.15^{*}	+1.10

Test BLEU using POSTNORM or PRENORM and different normalization techniques. Published values are from Wang et al. [93], Neubig and Hu [55], Aharoni et al. [1]. †, ‡ and ∗ indicate significant improvement of (3) over (2), (4) over (3), and (4) over (1), respectively; $p < 0.01$ via bootstrap resampling [37].

multilingual NMT models on these pairs. They outperform the best published numbers, which are all Transformer models in the past year, by an average margin of +4.0 BLEU. Then, we see that PRENORM (2) achieves comparable or slightly better results than POSTNORM on all tasks. FIXNORM (3) gives an additional gain, especially on ar→en ($p < 0.01$).

Finally, we replace LAYERNORM with SCALENORM (4). SCALENORM significantly improves on LAYERNORM for two very low-resource pairs, gl→en and sk→en. On the other tasks, it performs comparably to LAYERNORM. Upon aggregating all changes, our final model with SCALENORM and FIXNORM improves over our strong baseline with POSTNORM on all tasks by an average of +1.1 BLEU ($p < 0.01$), with each change contributing an average of at least +0.3 BLEU. In Section 6.4.2, we further examine where the performance gains of SCALENORM come from.

Moreover, SCALENORM is also faster than LAYERNORM. Recall that for each vector of size d, LAYERNORM needs to compute mean, standard deviation, scaling, and shifting, which costs $O(7d)$ operations. For SCALENORM, we only need $O(3d)$ operations to perform normalization and global scaling. This does not account for further gains due to

TABLE 6.4

DEV BLEU SCORES WITH DIFFERENT LEARNING RATE SCHEDULERS

	gl→en	**sk→en**	**en→vi**	**en→he**	**ar→en**
NoWarmup	18.00	28.92	28.91	30.33	35.40
InvSqrtDecay	22.18	29.08	28.84	30.30	35.33
ValDecay	21.45	29.46	28.67	30.69	35.46
InvSqrtDecay + 2×LR	21.92	29.03	28.76	30.50	35.33
ValDecay + 2×LR	21.63	29.49	28.46	30.13	34.95

reduction in parameters. In our implementation, training with ScaleNorm is around 5% faster than with LayerNorm, similar to the speedups on NMT observed by Zhang and Sennrich [97]'s RMSNorm (which can be viewed as ScaleNorm with per-unit scales; see Section 6.4.2).

6.3.4 Learning rates

We compare the original learning rate schedule in equation 6.7 (InvSqrtDecay) with validation-based decay (ValDecay), possibly with no warmup (NoWarmup). We use $\lambda = 1$, $n_{warmup} = 8\text{k}$ for InvSqrtDecay and ValDecay. For NoWarmup, we instead use a learning rate of $3 \cdot 10^{-4}$ for all datasets. For both ValDecay and NoWarmup, we take $\alpha_{decay} = 0.8$ and *patience* = 3. For experiments with high learning rate, we use either ValDecay or InvSqrtDecay with $\lambda = 2$ (giving a peak learning rate of $\approx 10^{-3}$). All experiments use PreNorm + FixNorm + ScaleNorm.

In Table 6.4, we see that NoWarmup performs comparably to InvSqrtDecay and ValDecay except on gl→en. We believe that in general, one can do without warmup, though it remains useful in the lowest resource settings. In our 2×LR experiments, we can

TABLE 6.5

DEV BLEU SCORES USING NOWARMUP

	4 layers	5 layers	6 layers
POSTNORM	18.31	fails	fails
PRENORM	28.33	28.13	28.32

Development BLEU on en→vi using NOWARMUP, as number of encoder/decoder layers increases.

still attain a maximum learning rate of 10^{-3} without disproportionately overfitting to small datasets like gl→en.

One might hypothesize that VALDECAY converges more quickly to better minima than INVSQRTDECAY by staying at high learning rates for longer. However, both schedulers achieve similar results with or without doubling the learning rate. This may be due to the tail-end behavior of VALDECAY methods, which can involve multiplicative decays in rapid succession. Finally, our 2×LR experiments, while not yielding better performance, show that PRENORM allows us to train the Transformer with a very high learning rate despite small batches (4k tokens).

Since PRENORM can train without warmup, we wonder if POSTNORM can do the same. We run experiments on en→vi with NOWARMUP, varying the number of encoder/decoder layers. As seen in Table 6.5, POSTNORM often fails without warmup even with 5 or 6 layers. Even at 4 layers, one achieves a subpar result compared to PRENORM. This reaffirms Section 6.3.2 in showing that PRENORM is more stable than POSTNORM under different settings.

TABLE 6.6

WMT '14 ENGLISH-TO-GERMAN BLEU SCORES

	newstest2014
POSTNORM + LAYERNORM (Vaswani et al. [87])	27.3
PRENORM + LAYERNORM	26.83
PRENORM + **fixnorm** + SCALENORM	27.07
POSTNORM + LAYERNORM	27.58
POSTNORM + **fixnorm** + SCALENORM	27.57

6.3.5 High-resource setting

Since all preceding experiments were in low-resource settings, we examine if our claims hold in a high-resource setting. We train the Transformer base model on WMT '14 English-German using FAIRSEQ and report tokenized BLEU scores on *newstest2014*.

In Table 6.6, SCALENORM and FIXNORM achieve equal or better results than LAYERNORM. Since SCALENORM is also faster, we recommend using both as drop-in replacements for LAYERNORM in all settings. Surprisingly, in this task POSTNORM works notably better than PRENORM; one observes similar behavior in Wang et al. [91]. We speculate this is related to identity residual networks acting like shallow ensembles [89] and thus undermining the learning of the longest path; further study is required.

TABLE 6.7

TEST BLEU OF VARIOUS ℓ_2-BASED NORMALIZATION TECHNIQUES

	gl→en	**sk→en**	**en→vi**	**en→he**	**ar→en**
RMSNORM + **fixnorm**	20.92	30.36	32.54	28.29	33.67
SCALENORM + **fixnorm**	20.91	30.25	32.79	28.44	34.15
SCALENORM (g= $\sqrt{d}$) + **fixnorm** (learned)	21.18	30.36	32.66	28.19	34.11
SCALENORM (g= $\sqrt{d}$) + **fixnorm** (learned) + VALDECAY	20.36	30.45	32.83	27.97	33.98
SCALENORM (g= $\sqrt{d}$) + **fixnorm** (learned) + VALDECAY + 2×LR	21.15	30.57	31.81	25.00	28.92

We tried various ℓ_2-based normalization techniques with different numbers of learned g: $O(Ld)$ vs. $O(L)$ vs. $O(1)$.

6.4 Analysis

6.4.1 Performance curves

Figure 6.1 shows that PRENORM not only learns faster than POSTNORM, but also outperforms it throughout training. Adding FIXNORM also gives faster learning at first, but only achieves close performance to that with PRENORM and no FIXNORM. However, once paired with SCALENORM, we attain a better BLEU score at the end. Because of the slow warmup period, SCALENORM with warmup learns slower than SCALENORM without warmup initially; however, they all converge at about the same rate.

To visualize how PRENORM helps backpropagation, we plot the global gradient norms from our runs in Figure 6.2. POSTNORM produces noisy gradients with many sharp spikes, even towards the end of training. On the other hand, PRENORM has fewer noisy gradients with smaller sizes, even without warmup. LAYERNORM has lower global norms than SCALENORM + FIXNORM but it has more gradient components corresponding to normalization.

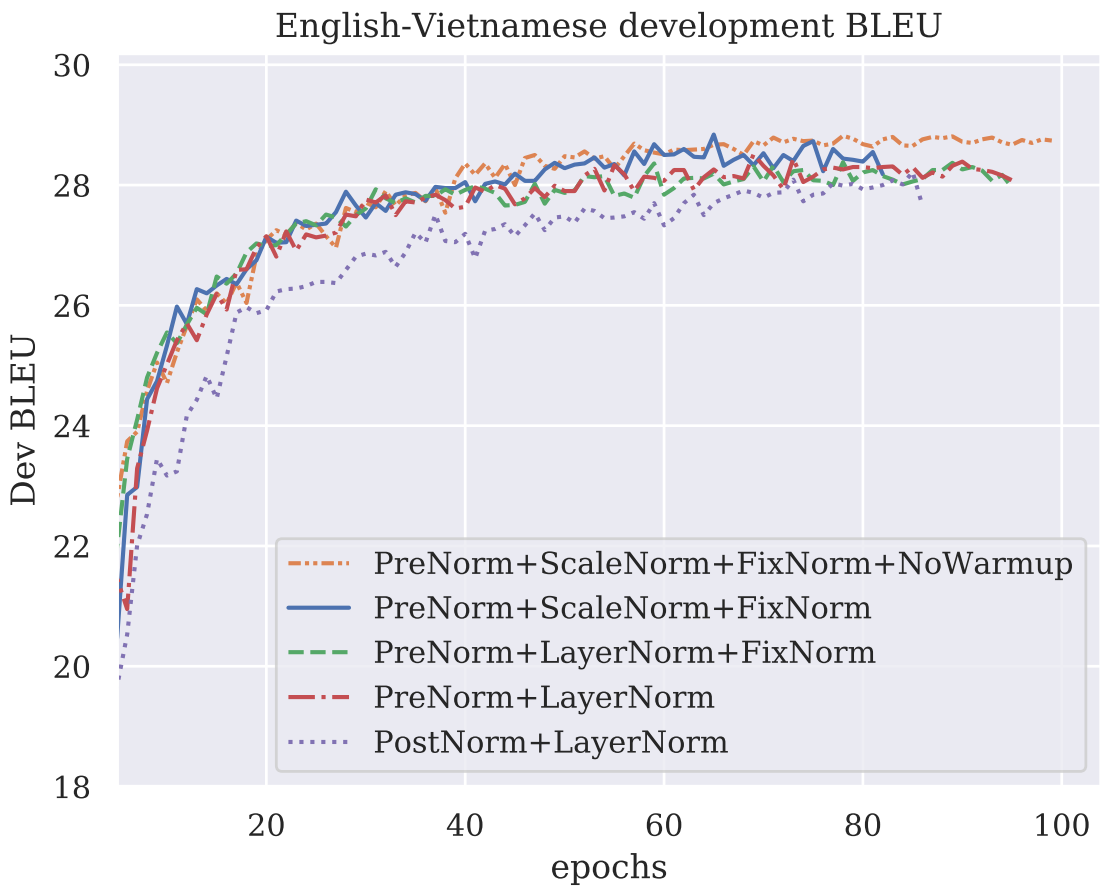


Figure 6.1. Development BLEU on en→vi with PostNorm or PreNorm, and with LayerNorm or ScaleNorm.

6.4.2 Activation scaling and the role of g

One motivation for ScaleNorm was that it expressed a good inductive bias for the global scaling of activations, independent of distributional stability (Section 6.2.3). In contrast, a contemporaneous work [97] studies *root mean square layer normalization* (RMSNorm), which still follows layer normalization's motivation but reduces overhead by forgoing additive adjustments, using only a scaling g_i per activation a_i. Despite their differing motives, tying the g_i of RMSNorm and dividing by $\sqrt{d}$ retrieves ScaleNorm.

Hence we can frame our comparisons in terms of number of learnable parameters. We rerun our PreNorm experiments with RMSNorm. We also consider fixing $g = \sqrt{d}$ for

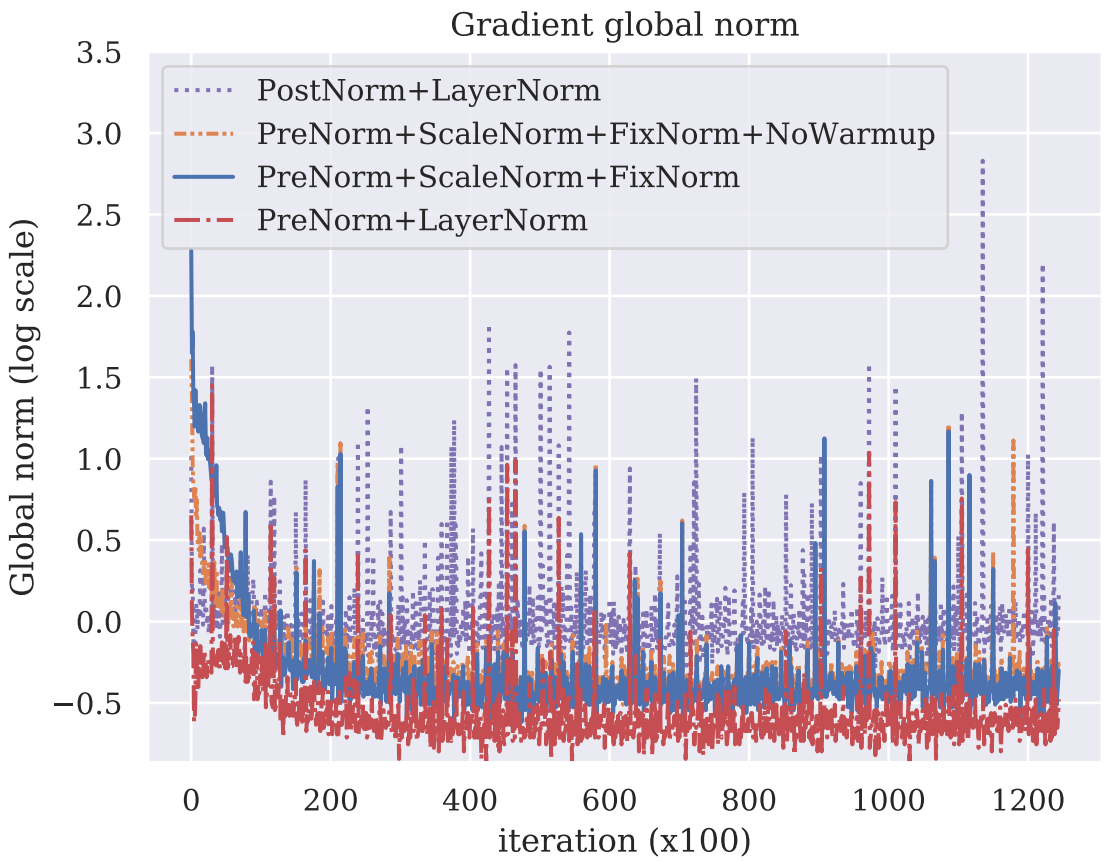


Figure 6.2. The global norm of gradients when using PostNorm or PreNorm, and with LayerNorm, ScaleNorm and FixNorm. Best viewed in color.

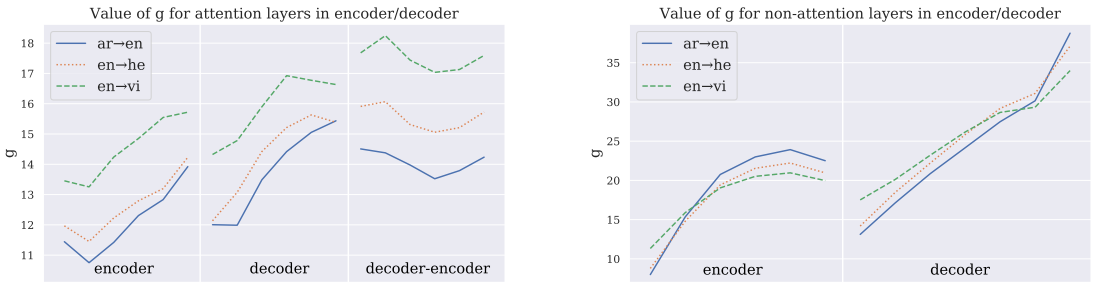


Figure 6.3: Learned g values for PreNorm + ScaleNorm + FixNorm models, versus depth. **Left:** Attention sublayers (*decoder-encoder* denotes decoder sublayers attending on the encoder). **Right:** Feedforward sublayers and the final linear layer.

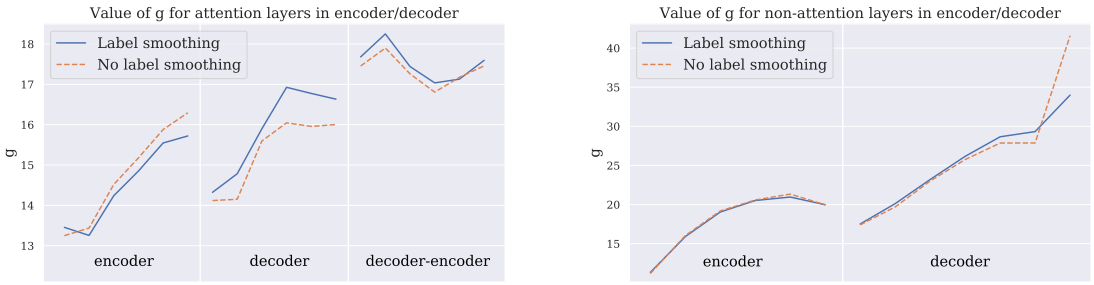


Figure 6.4: Learned g values for our PRENORM + SCALENORM + FIXNORM en→vi model (with and without label smoothing), versus depth. **Left** and **Right** are the same as in Figure 6.3.

SCALENORM, where only FIXNORM has learnable g. Table 6.7 shows that SCALENORM always performs comparably or better than RMSNORM. Surprisingly, the fixed-g model performs comparably to the one with learnable g. However, at higher learning rates, fixed-g models perform much worse on ar→en, en→he and en→vi (VALDECAY with and without 2×LR). We conjecture that learning g is required to accommodate layer gradients.

In Figure 6.3, we plot the learned g values for pairs with 100k+ examples. For all but the decoder-encoder sublayers, we observe a positive correlation between depth and g, giving credence to SCALENORM's inductive bias of global scaling. This trend is clearest in the decoder, where g linearly scales up to the output layer, perhaps in tandem with the discriminativeness of the hidden representations [43]. We also note a negative correlation between the number of training examples and the magnitude of g for attention sublayers, which may reflect overfitting.

Finally, to affirm our intuition for interpreting g, we plot g values with and without label smoothing (Figure 6.4). We see a difference in later layers of the decoder; there, removing label smoothing results in lower g values except at the output layer, where g increases sharply. This corresponds to the known overconfidence of translation models' logits, on which label smoothing has a downscaling effect [53].

CHAPTER 7

UNTIED POSITIONAL ATTENTION FOR NEURAL MACHINE TRANSLATION

In this chapter, we investigate the *untied positional attention* in Transformer. We will show this simple architectural change can provide significant improvement for several low-resource language pairs and better interpretations of how positions work in Transformer. Unbeknownst to me, it had been investigated months earlier by Ke et al. [34]. The main difference between theirs and mine is they applied it for language model pretraining and I applied it for NMT. Needless to say, it is not novel nor publishable; however, I think it still has merits for future NMT practitioners. For this reason, I decided to describe it here in this short chapter.

7.1 Introduction

Because Transformer processes input in parallel, it does not have a notion of sequence. To inform Transformer of a word's position, Vaswani et al. [87] propose to enhance the input with positional embedding. They experiment with both learned and sinusoidal embedding and find both perform comparably, with the latter having the advantages of being parameter-less and of the potential to generalize to longer sentences. In this section, we will briefly describe the sinusoidal positional embedding and how it interacts with self-attention, leading to our proposed untied positional attention. Interested readers could refer to section 2.2 for more details of the Transformer model.

The sinusoidal positional embedding is defined as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{7.1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{7.2}$$

and is summed element-wise to each word embedding at the input as seen in Figure 2.2.

Let $e_q, p_q \in \mathcal{R}^d$ and $e_k, p_k \in \mathcal{R}^d$ be the word and positional embedding of a query and a key respectively. In Transformer's attention, we first apply a linear projection of queries and keys before taking their dot product. Let $W_q, W_k \in \mathcal{R}^{d\times d}$ be the weights of the linear projections for queries and keys. Ignoring biases, the dot product is:

$$s = (e_q + p_q)W_q((e_k + p_k)W_k)^T \tag{7.3}$$

$$= (e_q + p_q)W_qW_k^T(e_k^T + p_k^T) \tag{7.4}$$

$$= e_qW_qW_k^Te_k^T + p_qW_qW_k^Tp_k^T + p_qW_qW_k^Te_k^T + e_qW_qW_k^Tp_k^T \tag{7.5}$$

In equation 7.5, the four terms measure the correlations between query and key, query's position and key's position, query's position and key, and query and key's position respectively. Our first observation is a word could appear anywhere in a sentence, therefore the third and fourth terms could be potentially dropped. It makes sense to measure the correlation between a query and a key, for example "New" often goes with "York". The same goes for positions, for example if we are using BPE then in self-attention, a query should look for keys located close to it (subwords of the same word). However, because the first and second terms model two entirely different things, using the same parameters W_q, W_k could be a bottleneck. This is also our second observation which leads to a simple solution: drop the last two terms and use different parameters for positions:

$$s = e_q W_q W_k^T e_k^T + p_q W'_q W'^T_k p_k^T \tag{7.6}$$

7.2 Experiments

Following chapter 4, we use Transformer and experiment on {Galician, Slovak} to English and English to {Hebrew, Vietnamese}. We kindly refer readers to previous chapter for data statistics and for the model configuration, training, and evaluation details.

Following equation 7.6, we use different parameters W'_q, W'_k for dot product between query positions and key positions. Similar to Ke et al. [34], we only compute the second term in 7.6 once and reuse it across layers to reduce computation. We call this model **untied-pos**.

The goal of **untied-pos** is to learn the correlation scores between every pair of query and key positions, i.e. to learn the value of some matrix $M \in \mathcal{R}^{L_q \times L_k}$ with L_q, L_k be the maximum lengths of query and key sentences. We can think of the term $p_q W'_q W'^T_k p_k^T$ as a low-rank matrix factorization of M with $W'_q W'_k$ be learnable parameters. This means the role of p_q, p_k is not important as long as they are pair-wise independent enough. For this reason, we also explore a variant of **untied-pos** in which p_q, p_k are randomly initialized vectors instead of sinusoidal. We call this model **random-untied-pos**.

7.3 Results and Analysis

Tables 7.1 and 7.2 show the BLEU scores on development and test sets respectively for all language pairs. We can see that untying positional attention from word attention yield on average more than +1 BLEU score over the baseline. Moreover, randomly initialized positional embedding perform comparably to its sinusoidal counterpart, significantly improving over the baseline for all language pairs except en→vi.

TABLE 7.1

DEV BLEU SCORES FOR UNTIED POSITIONAL ATTENTION

Row		gl→en	sk→en	en→vi	en→he	avg	Δ
1	**baseline**	22.86	29.15	29.01	30.26	27.82	+0.0
2	**untied-pos**	26.33	30.14	28.79	30.79	29.01	+1.19
3	**random-untied-pos**	25.71	30.20	28.90	30.65	28.87	+1.05

To understand what patterns positional attention learns, we normalize then plot the values $p_q W'_q W'^T_k p^T_k$ for all position pairs $i, j \in [0, 100]$. We find in the case of self-attention, some patterns are not easy to decipher but some clearly show the positional attention tends to look in a close proximity for nearby subwords. An example is shown in Figure 7.1 for decoder's (top) and encoder's (bottom) self-attentions. For cross-attention, every single head has similar trend which follows along the diagonal as seen in Figure 7.2. We also plot the target length versus source length as red dots in the same figure. We can see that the positional attention follows the length plot closely which indicates that the positional attention could be used to track the source sentence's length. Finally, in both self- and cross-attentions, we find both sinusoidal and randomly initialized positions tend to learn similar patterns (left vs right in Figures 7.1 and 7.2). This, along with the similar performance, indicates the type of positional encoding is probably not too important for untied positional attention.

TABLE 7.2

TEST BLEU SCORES FOR UNTIED POSITIONAL ATTENTION

Row		gl→en	sk→en	en→vi	en→he	avg	Δ
1	**baseline**	20.73	30.32	32.72	28.06	27.96	+0.0
2	**untied-pos**	23.21†	31.13†	32.95	28.70†	29.00	+1.04
3	**random-untied-pos**	23.20†	31.35†	32.87	28.43†	28.97	+1.01

† = statistically significant improvement on the test set compared to baseline ($p < 0.01$).

7.4 Conclusion

In this chapter, we explore the untied positional attention in Transformer. Our experiments demonstrate this simple change could significantly improve performance on low-resource language pairs. Furthermore, untying positional attention from word attention helps to reveal the patterns which the model learns from positions. This could potentially help future study on model interpretability.

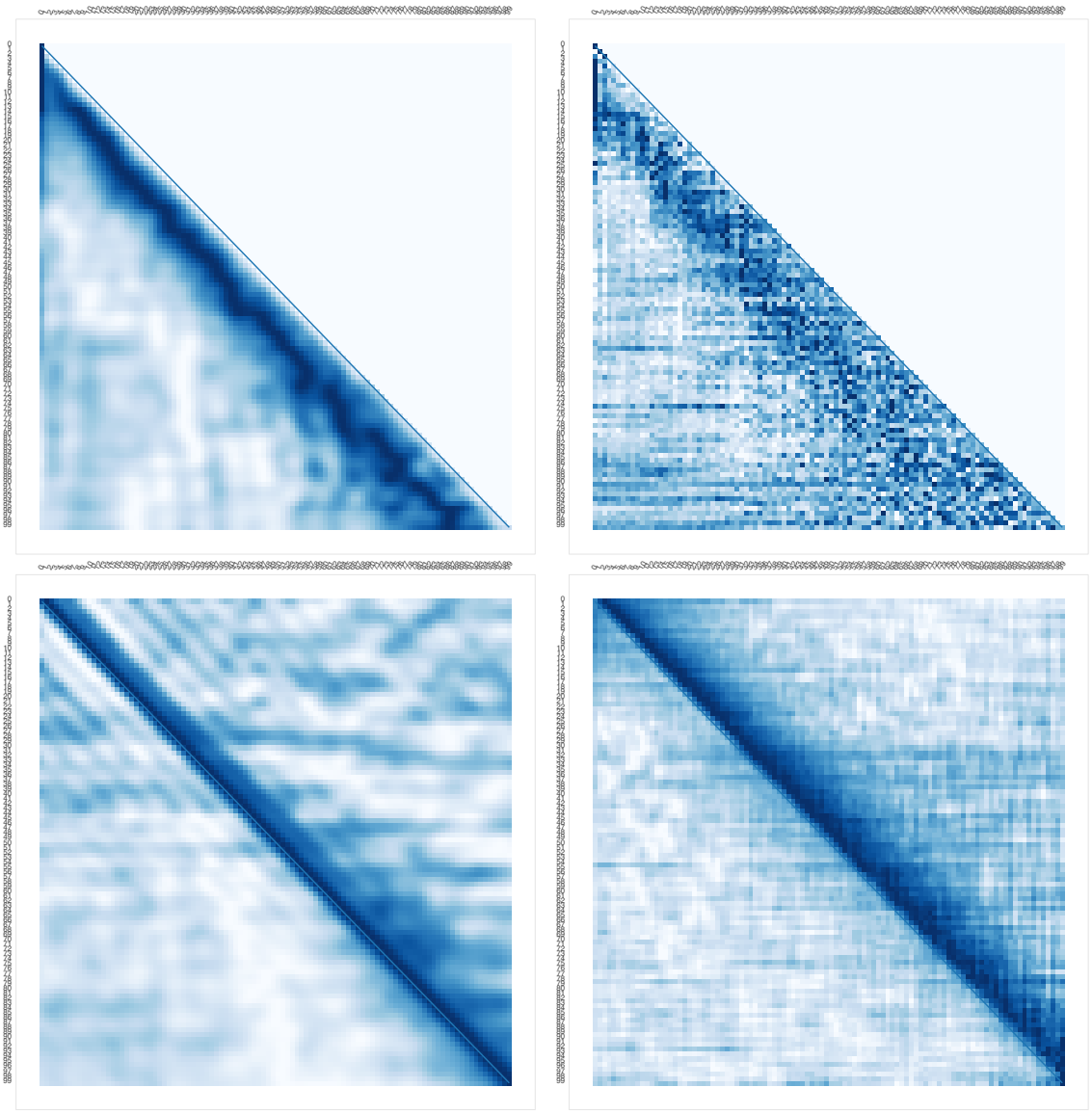


Figure 7.1. Two learned patterns from self-attention in decoder (top) and encoder (bottom) in **untied-pos** (left) and **random-untied-pos** (right).

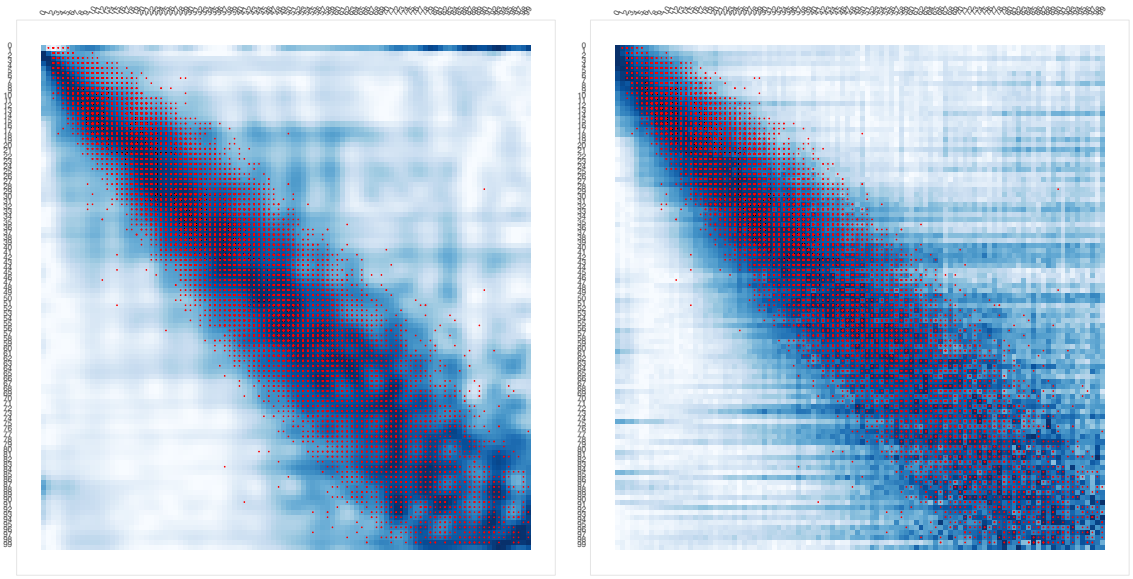


Figure 7.2. Positional attentions in **untied-pos** (left) and **random-untied-pos** (right) seem to track source sentence lengths (red dots).

CHAPTER 8

CONCLUSION

In this dissertation, I have covered some background on Neural Machine Translation with a particular focus on the low-resource domain. Through existing contributions and related work, I have demonstrated the weakness of vanilla NMT systems when dealing with limited data. To address this problem, I have proposed several methods under the theme of **better data exploitation**, **better normalization**, and **better modeling** to allow us to successfully train strong NMT models for low-resource languages.

More specifically, we have seen in chapter 3 how to make use of relationship between languages at subword-level to improve transfer learning from one language pair to another. On the same topic of **better data exploitation**, I also study a simple data augmentation method via concatenation in chapter 4 and validate its effectiveness in low-resource scenarios. On **better normalization**, I propose a simple ℓ_2-based normalization technique called **fixnorm** in chapter 5 to address the rare-word mistranslation problem. This technique, first studied in the context of RNN-based models, is further extended to Transformer in chapter 6. This also results in a new layer normalization method called ScaleNorm which is simpler, faster, and requires less parameters than the traditional Layer Normalization. Regarding **better modeling**, I propose three different changes to NMT models. In chapter 5, I design a lexical model which can improve translation for rare words such as name entities. In chapter 6, I study the Transformer model and make various adjustments to improve its training stability. Finally, in chapter 7, I show how untying the positional attention in Transformer can lead to improved performance for low-resource translation and better model interpretability.

To close this work, I would like to briefly discuss two questions. First, with so many techniques presented in this thesis, what is a recipe that I recommend for building a strong NMT model for a low-resource language pair (LRP)? Second, what is the future of low-resource NMT? To answer the first question, let us consider two scenarios: one in which there exists a high-resource language pair (HRP) that is related to our LRP (e.g., Uzbek-English as the HRP for Turkish-English), and one in which there is not such an HRP. In the former case, the best practice I am aware of is transfer learning. As we have seen in chapter 3, we could build a NMT model for HRP then continue training on LRP. Furthermore, related work on multilingual NMT [32, 23, 1, 55] has shown that we can simply train a NMT system on the union of the LRP's and the HRP's data. However, if such an HRP doesn't exist, we have to rely on LRP. In this case, I would suggest building a Transformer model as described in chapter 6 along with the data augmentation via concatenation as studied in chapter 4.

To answer the second question, I think we need to consider two aspects of NMT: availability and usability. By availability, I mean that for any language pair, there should exist a NMT model. At this level, the model might perform undesirably; it could potentially translate only simple sentences. By usability, I mean the model must translate accurately enough for casual communication. We need to get to availability first before considering usability. I think the only way to reach availability for every low-resource language is via unsupervised NMT [3], as this approach allows us to make use of monolingual data which is often widely available or could be easily collected. To reach usability, however, I still believe we must have a good amount of bitext data. Based on my experience, we should aim for at least 100k but ideally 1M examples per language pair. Bitext mining approaches [76] hopefully could help to lower the cost for data collecting. Nevertheless, we should be careful of the quality of this kind of data, as it often contains either wrong translations or harmful content [8]. Practically, we might have less data. In that case, we can always improve the unsupervised NMT models using whatever bitext data that is available with

methods explored in this thesis or in other work.

Needless to say, there is more to do to improve NMT in general and low-resource machine translation in particular. All the techniques discussed here are a part of that effort and hopefully they will shed some light onto previous and future research. For example, we know that multilingual training can improve translation performance for low-resource languages. Our study on language relatedness and transfer learning in chapter 3 can partially help to explain why this is the case. While all the normalization techniques in chapters 5 and 6 are invented to address the rare word mistranslation issue, they can also be considered as empirical evidence for how normalization can impact optimization as studied by Santurkar et al. [74]. The data augmentation method in chapter 4 represents a case in which an intuitive explanation might not be the reason behind some improvement and better explanations should be carefully sought out. Finally, a growing body of work is focusing on interpretability of Deep Learning models, and they might benefit from the study in chapter 7. Overall, I anticipate future work can benefit from these techniques to bring Machine Translation to more languages of the world.

BIBLIOGRAPHY

1. R. Aharoni, M. Johnson, and O. Firat. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1388. URL https://www.aclweb.org/anthology/N19-1388.

2. A. Anastasopoulos, A. Lui, T. Q. Nguyen, and D. Chiang. Neural machine translation of text from non-native speakers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3070–3080, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1311. URL https://www.aclweb.org/anthology/N19-1311.

3. M. Artetxe, G. Labaka, E. Agirre, and K. Cho. Unsupervised neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy2ogebAW.

4. P. Arthur, G. Neubig, and S. Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1162. URL https://www.aclweb.org/anthology/D16-1162.

5. J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *CoRR*, abs/1607.06450, 2015.

6. D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.

7. Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018. URL https://openreview.net/pdf?id=BJ8vJebC-.

8. I. Caswell, J. Kreutzer, L. Wang, A. Wahab, D. van Esch, N. Ulzii-Orshikh, A. Tapo, N. Subramani, A. Sokolov, C. Sikasote, M. Setyawan, S. Sarin, S. Samb, B. Sagot, C. Rivera, A. Rios, I. Papadimitriou, S. Osei, P. J. O. Suárez, I. Orife, K. Ogueji, R. A. Niyongabo, T. Q. Nguyen, M. Müller, A. Müller, S. H. Muhammad, N. Muhammad,

A. Mnyakeni, J. Mirzakhalov, T. Matangira, C. Leong, N. Lawson, S. Kudugunta, Y. Jernite, M. Jenny, O. Firat, B. F. P. Dossou, S. Dlamini, N. de Silva, S. Çabuk Ballı, S. Biderman, A. Battisti, A. Baruwa, A. Bapna, P. Baljekar, I. A. Azime, A. Awokoya, D. Ataman, O. Ahia, O. Ahia, S. Agrawal, and M. Adeyemi. Quality at a glance: An audit of web-crawled multilingual datasets, 2021.

9. M. Cettolo, J. Niehues, L. Bentivogli, R. Cattoni, and M. Federico. The IWSLT 2015 Evaluation Campaign. In *IWSLT*, pages 3–4, 2015. URL `http://workshop2015.iwslt.org/downloads/IWSLT_2015_EP_0.pdf`.

10. M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, and M. Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1008. URL `https://www.aclweb.org/anthology/P18-1008`.

11. K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

12. K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL `https://www.aclweb.org/anthology/D14-1179`.

13. J. Chung, K. Cho, and Y. Bengio. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1160. URL `https://www.aclweb.org/anthology/P16-1160`.

14. L. Dong, S. Xu, and B. Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888, 2018. doi: 10.1109/ICASSP.2018.8462506.

15. M. Fadaee, A. Bisazza, and C. Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-2090`.

16. F. Gao, J. Zhu, L. Wu, Y. Xia, T. Qin, X. Cheng, W. Zhou, and T.-Y. Liu. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1555. URL https://www.aclweb.org/anthology/P19-1555.

17. J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/gehring17a.html.

18. X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

19. X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

20. J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://www.aclweb.org/anthology/P16-1154.

21. C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1014. URL https://www.aclweb.org/anthology/P16-1014.

22. Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1315–1325, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1133. URL https://www.aclweb.org/anthology/N19-1133.

23. T.-L. Ha, J. Niehues, and A. Waibel. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*, 2016.

24. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. ISBN 9781467388504. doi: 10.1109/CVPR.2016.90.

25. K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

26. F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, and M. Post. The Sockeye neural machine translation toolkit. In *AMTA*, pages 200–207, 2018. URL https://www.aclweb.org/anthology/W18-1820/.

27. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8), 1997.

28. H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

29. S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, pages 448–456, 2015. ISBN 9780874216561. doi: 10.1007/s13398-014-0173-7.2.

30. S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1001. URL https://www.aclweb.org/anthology/P15-1001.

31. S. Jean, S. Lauly, O. Firat, and K. Cho. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*, 2017.

32. M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017. doi: 10.1162/tacl_a_00065. URL https://www.aclweb.org/anthology/Q17-1024.

33. M. Junczys-Dowmunt. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5321. URL https://www.aclweb.org/anthology/W19-5321.

34. G. Ke, D. He, and T.-Y. Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=09-528y2Fgf.

35. Y. Kim, D. T. Tran, and H. Ney. When and why is document-level context useful in neural machine translation? In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 24–34, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6503. URL https://www.aclweb.org/anthology/D19-6503.

36. D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

37. P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3250.

38. P. Koehn and R. Knowles. Six challenges for neural machine translation. In *Proc. First Workshop on Neural Machine Translation*. Association for Computational Linguistics, 2017.

39. P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-2045.

40. S. Kondo, K. Hotate, M. Kaneko, and M. Komachi. Sentence concatenation approach to data augmentation for neural machine translation. In *Proc. NAACL Student Research Workshop*, 2021. To appear.

41. W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

42. S. Läubli, R. Sennrich, and M. Volk. Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1512. URL https://www.aclweb.org/anthology/D18-1512.

43. D. Liang, Z. Huang, and Z. C. Lipton. Learning noise-invariant representations for robust speech recognition. In *SLT*, pages 56–63, 2018. doi: 10.1109/SLT.2018.8639575.

44. A. Liu and K. Kirchhoff. Context models for OOV word translation in low-resource languages. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 54–67, Boston, MA,

Mar. 2018. Association for Machine Translation in the Americas. URL https://www.aclweb.org/anthology/W18-1806.

45. W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 507–516, 2016.

46. C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In V. Kurková, Y. Manolopoulos, B. Hammer, L. S. Iliadis, and I. Maglogiannis, editors, *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I*, volume 11139 of *Lecture Notes in Computer Science*, pages 382–391. Springer, 2018. doi: 10.1007/978-3-030-01418-6_38. URL https://doi.org/10.1007/978-3-030-01418-6_38.

47. M.-T. Luong and C. D. Manning. Stanford neural machine translation systems for spoken language domain. In *Proc. IWSLT*, 2015.

48. M.-T. Luong and C. D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1100. URL https://www.aclweb.org/anthology/P16-1100.

49. T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

50. T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL https://www.aclweb.org/anthology/P15-1002.

51. H. Mi, Z. Wang, and A. Ittycheriah. Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 124–129, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2021. URL https://www.aclweb.org/anthology/P16-2021.

52. M. Morishita, Y. Oda, G. Neubig, K. Yoshino, K. Sudoh, and S. Nakamura. An empirical study of mini-batch creation strategies for neural machine translation. In

Proceedings of the First Workshop on Neural Machine Translation, pages 61–68, Vancouver, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3208. URL https://www.aclweb.org/anthology/W17-3208.

53. R. Müller, S. Kornblith, and G. E. Hinton. When does label smoothing help? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f1748d6b0fd9d439f71450117eba2725-Paper.pdf.

54. P. Nakov and H. T. Ng. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1367, Singapore, Aug. 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D09-1141.

55. G. Neubig and J. Hu. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1103. URL https://www.aclweb.org/anthology/D18-1103.

56. C. Ngo and T. H. Trinh. Better translation for Vietnamese. https://blog.vietai.org/sat/, 2021. (Accessed on 04/27/2021).

57. T. Nguyen and D. Chiang. Improving Lexical Choice in Neural Machine Translation. In *NAACL-HLT*, pages 334–343, 2018. doi: 10.18653/v1/n18-1031.

58. T. Q. Nguyen and D. Chiang. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan, Nov. 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-2050.

59. T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proc. Workshop on Spoken Language Translation*, 2019. doi: 10.5281/zenodo.3525484.

60. T. Q. Nguyen, K. Murray, and D. Chiang. Data augmentation by concatenation for low-resource translation: A mystery and a solution. In *The International Conference on Spoken Language Translation*, 2021.

61. F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. doi: 10.1162/089120103321337421. URL https://www.aclweb.org/anthology/J03-1002.

62. F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073133. URL https://www.aclweb.org/anthology/P02-1038.

63. M. Ott, S. Edunov, D. Grangier, and M. Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6301. URL https://www.aclweb.org/anthology/W18-6301.

64. M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.org/anthology/N19-4009.

65. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

66. R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org, 2013.

67. G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. E. Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR (Workshop)*, 2017. URL https://openreview.net/forum?id=HyhbYrGYe.

68. M. Popel and O. Bojar. Training Tips for the Transformer Model. *Prague Bull. Math. Linguistics*, 110(1):43–70, 2018. doi: 10.2478/pralin-2018-0002.

69. O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-2025.

70. Y. Qi, D. Sachan, M. Felix, S. Padmanabhan, and G. Neubig. When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation? In *NAACL-HLT*, pages 529–535, 2018. doi: 10.18653/v1/n18-2084.

71. A. Rush. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2509. URL https://www.aclweb.org/anthology/W18-2509.

72. J. Salazar, K. Kirchhoff, and Z. Huang. Self-attention Networks for Connectionist Temporal Classification in Speech Recognition. In *ICASSP*, pages 7115–7119, 2019. ISBN 978-1-4799-8131-1. doi: 10.1109/ICASSP.2019.8682539.

73. T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/ed265bc903a5a097f61d3ec064d96d2e-Paper.pdf.

74. S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

75. D. Saunders, F. Stahlberg, and B. Byrne. Using context in neural machine translation training objectives. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7764–7770, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.693. URL https://www.aclweb.org/anthology/2020.acl-main.693.

76. H. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán. WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361, Online, Apr. 2021. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.eacl-main.115.

77. R. Sennrich, B. Haddow, and A. Birch. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2323. URL https://www.aclweb.org/anthology/W16-2323.

78. R. Sennrich, B. Haddow, and A. Birch. 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, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

79. N. Shazeer and M. Stern. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *ICML*, pages 4603–4611, 2018.

80. D. Stojanovski and A. Fraser. Combining local and document-level context: The LMU Munich neural machine translation system at WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 400–406, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5345. URL https://www.aclweb.org/anthology/W19-5345.

81. S. Sukhbaatar, E. Grave, G. Lample, H. Jegou, and A. Joulin. Augmenting Self-attention with Persistent Memory. *CoRR*, abs/1907.01470, 2019.

82. Z. Sun, M. Wang, H. Zhou, C. Zhao, S. Huang, J. Chen, and L. Li. Capturing longer context for document-level neural machine translation: A multi-resolutional approach. *arXiv preprint arXiv:2010.08961*, 2020.

83. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

84. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.308. URL https://doi.org/10.1109/CVPR.2016.308.

85. X. Tan, L. Zhang, D. Xiong, and G. Zhou. Hierarchical modeling of global context for document-level neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1576–1585, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1168. URL https://www.aclweb.org/anthology/D19-1168.

86. J. Tiedemann and Y. Scherrer. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4811. URL https://www.aclweb.org/anthology/W17-4811.

87. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors,

Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

88. A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, L. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL http://arxiv.org/abs/1803.07416.

89. A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/37bc2f75bf1bcfe8450a1a41c200364c-Paper.pdf.

90. F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: L₂ hypersphere embedding for face verification. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1041–1049, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349062. doi: 10.1145/3123266.3123359. URL https://doi.org/10.1145/3123266.3123359.

91. Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1176. URL https://www.aclweb.org/anthology/P19-1176.

92. X. Wang, Z. Lu, Z. Tu, H. Li, D. Xiong, and M. Zhang. Neural machine translation advised by statistical machine translation. In *Proc. AAAI*, 2017.

93. X. Wang, H. Pham, Z. Dai, and G. Neubig. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1100. URL https://www.aclweb.org/anthology/D18-1100.

94. Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. arXiv:1609.08144.

95. W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization, 2014. arXiv:1409.2329.

96. M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

97. B. Zhang and R. Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

98. J. Zhang, H. Luan, M. Sun, F. Zhai, J. Xu, M. Zhang, and Y. Liu. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1049. URL https://www.aclweb.org/anthology/D18-1049.

99. Z. Zheng, X. Yue, S. Huang, J. Chen, and A. Birch. Towards making the most of context in neural machine translation. In *Proceedings of IJCAI-PRICAI*, 2020.

100. B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL https://www.aclweb.org/anthology/D16-1163.

This document was prepared & typeset with pdfLaTeX, *and formatted with* nddiss2$_\varepsilon$ *classfile (v3.2017.2[2017/05/09])*