Adapting Large Multilingual Machine Translation Models to Unseen Low Resource Languages via Vocabulary Substitution and Neuron Selection

Mohamed AbdelGhaffar mohamed.abdelghaffar.guc.masters@gmail.com
German University in Cairo

Amr Hussein ElMogy amr.elmougy@guc.edu.eg
German University in Cairo

Nada Ahmed Hamed Sharaf nada.hamed@giu-uni.de
German International University, Cairo, Egypt

Abstract

We propose a method to adapt multilingual Machine Translation models to a low resource language (LRL) that was not included during the pre-training/training phases. We utilize data from a closely related High resource language (HRL) to fine-tune the model. Along with that use neuron-ranking analysis to select neurons that are most influential to the high resource language (HRL) and fine-tune only this subset of the deep neural network's neurons. We experiment with three mechanisms to compute such ranking. To allow for the potential difference in writing scripts between the HRL and LRL we utilize an alignment model to substitute HRL elements of the predefined vocab with appropriate LRL ones. In our experiments our method improves on both zero-shot and the stronger baseline of directly fine-tuning the MBART50 model on the low-resource data by 3 BLEU points in $Tajik \rightarrow English$ and 1.6 BLEU points in $English \rightarrow Tajik$ using Persian as the closest HRL on the FLORES101 devtest test set. We also show that as we simulate smaller data amounts, the gap between our method and direct fine-tuning continues to widen.

1 Introduction

Large Multilingual Machine Translation models have been achieving state-of-the-art (SOTA) performances on Machine Translation task (MT) in recent years (Tran et al., 2021). These models have also been shown to achieve gains on low-resource languages (LRL), all be it at the cost of slight regressions in the performances of high-resource language (HRL), they have even enabled translation on language pairs with zero parallel data (Johnson et al., 2017). In this work we define LRLs as languages that have less than 1M parallel training sentences. Whereas HRLs have a training set in the order of tens of millions of parallel sentences. Adapting these models to previously unseen LRLs can be challenging. Re-training these models can be expensive in terms of money and time. This is due to the large number of parameters and training data required to train these models, thus the need for powerful and expensive GPUs for extended periods of time. Moreover the training mechanism of standard modern tokenizers (most notably sentencepiece (Kudo and Richardson, 2018)) which assumes monolingual data of all languages that the model is to support would be present before training the model. This can lead to over-segmentation and high oov-rates (out of vocabulary) of LRL sentences when compared to

HRL sentences. This in turn leads to longer sequences, resulting in hindering the learning. We conducted our own analysis on the MBART50 (Tang et al., 2020) vocabulary set to demonstrate this (see section 4.3). To mitigate this effect we use a slightly modified version of vocabulary substitution (Garcia et al., 2021) that uses an HRL to LRL alignment model as a guide.

Furthermore given that LRLs by definition have a limited number of parallel training data samples, fine-tuning multilingual models can be especially tricky considering how easy it would be for the model to over-fit on the training data and/or be especially sensitive to the noisy data samples. While previous approaches have tried to augment LRL data via the HRL training dataset, we hypothesised that being selective regarding which neurons to fine-tune - in other words allowing the loss function's gradients to fall only into a certain subset of the network's neurons (hence only actually changing the weights of said subset) - would act as regularization technique thus mitigating the effect of over-fitting and noise in the LRL training set while maintaining or amplifying the gains originating from cross-lingual sharing with the HRL. We experiment with three techniques to compute the aforementioned sub-net:

Gradient Analysis : We compute the gradient of the output w.r.t each individual neuron, gradients across different time-steps are aggregated (we experiment with multiple aggregation functions). The gradients are then aggregated again across multiple sentences we consider the *magnitude* of the final outcome to be the neuron's importance score.

Activation Magnitude : We track the magnitude of the activation (output) of each neuron across multiple time-steps and multiple sentences. The activations go through the same two-layered aggregation procedure as with gradient analysis.

LASS : First proposed in (Lin et al., 2021). In order to compute the importance of neurons to a certain language L, we start by fine-tuning the model on L's data. Then we sort neurons by the absolute difference in the weights of the neuron between the original and fine-tuned models.

2 Background

Many of the recent breakthroughs in deep Natural Language Processing (NLP) models have relied heavily on growing the model in depth and number of parameters. This has shown to significantly improve model performance on many few-shot and zero-shot NLP tasks, with low resource machine translation being one of those tasks (Chowdhery et al., 2022). This however remains largely limited by the availability of a large quantity of resources for pre-training and/or fine-tuning such models. This leads us to believe that there is still benefit from the study and improvement of methods to adapt existing dense models to a new low-resource language. (Philip et al., 2020) explores adding an unseen language by training monolingual adapters. Adapters are small components that are trained traditionally while the rest of the network has been frozen (Bapna and Firat, 2019). While theoretically this requires no parallel data, it can easily be adapted to our setup by fine-tuning both the source language encoder adapter and the target language decoder adapter on the quantity of parallel data that is available. We compare against this approach on section 4.

Multiple approaches study how to add previously unseen languages while trying to maintain performance on the rest of the languages such as (Garcia et al., 2021) and (Berard, 2021). In this work on the other hand we are more interested in maximizing the performance of the model on the new LRL regardless of it's effect on other language-pairs. The assumption being the MNMT model after being tuned towards the LRL can later be distilled into a more deployment-friendly architecture (Kim and Rush, 2016). Previous work has shown that fine-tuning on a mix of HRL and LRL to be beneficial (Lakew et al., 2019; Neubig and Hu, 2018).

Other approaches relied on pivoting on the HRL, the hypothesis being that translation from HRL to LRL should be an easier task (Xia et al., 2019). We evaluate against this in section 4. All of these approaches however treat the multilingual model as a black box, and as far as we know no attempt has been made at being selective as to targeting certain neurons during fine-tuning.

3 Neuron Selection And Vocab Substitution

We describe our method in this section. The goal is to adapt a multilingual model to low resource, previously unseen language (LRL) by leveraging the model's knowledge of a similar high resource language (HRL). Our method can be broken down into three (possibly four) main stages:

1. Fine-tune the model on HRL data. We discuss the rationale behind this on section 3.1.
2. Classify the model's neurons into three groups:
   (a) Important Neurons that should be updated during LRL fine-tuning
   (b) Important Neurons that should not be updated during fine-tuning (for example neurons that capture English language specific properties)
   (c) Unimportant neurons that should be zero-d out.
3. (optional) Vocab-substitute from HRL-vocab to LRL vocab. This has been found to be especially useful when there is a large lexical gap between the HRL and LRL languages (for example written in two different scripts).
4. Fine-tune the model from (1), specifically the set of weights from (2.a), on the LRL data available.

In the reminder of this section we explain in some detail each phase.

3.1 Fine-tune on HRL data

Given that the HRL is by definition closely related to LRL, it stands to reason that biasing the model towards the HRL might make for a better base model for the LRL than the vanilla multilingual model. We show in 4 that this has in fact been useful.

Another important reason to fine-tune the model on HRL is that it is a prerequisite for multiple neuron sorting techniques detailed in 3.2.

3.2 Neurons Selection

The amount of *change* that the multilingual model exhibits during the fine-tuning stage is affected by a multitude of factors (for example how different are training samples from the ones the model has seen, the difficulty/complexity of the new training samples, the size of the fine-tuning dataset, etc). In our scenario the LRL has relatively few training samples. This means that directly fine-tuning all of the parameters models could potentially lead to the model exhibiting some of the following undesirable phenomena :

- **Over-fitting on noise patterns in the training data.** Due to the fact the model's extra capacity (that is model parameters were specific to other languages/language pairs than the ones we are interested in) is larger than the amount of useful information within the LRL data. Thus the risk of the fine-tuned model *forgetting* important linguistic information/properties from the HRL (that are potentially shared with the LRL) and *memorizing* the new training data is high, and could lead to hallucinations. Bounding the fine-tuning

process to the most "important" neurons can help mitigate this by limiting the model's degrees of freedom thus acting as a regularizer.

- **Missing out on potential quality of translation gains.** As we show in 4, neuron selection leads to better end-to-end quality of translation compared to fine-tuning all of the neurons directly. We hypothesized that since the fine-tuning process is limited to the most important HRL-related neurons, cross-lingual sharing would be maximized. This is also inline with the fact that with limited training samples, assuming we do not increase the learning rate value which would be dangerous, the amount of change that the model exhibits is limited. This lead us to hypothesize that steering the gradients towards the most important neurons would lead to better performance.
- **Longer training time.** This is a direct result of having to re-learn some of the patterns shared between the HRL and LRL. As stated above we hypothesize that in the case of fine-tuning all of the neurons, cross-lingual sharing could potentially be sub-optimal.

We experiment with various methods to determine the 'importance' of each neurons. We also test the importance of the neurons within two different environments.

1. Importance of neurons for language pair of interest $HRL \leftrightarrow EN$.
2. Importance of neurons for other high resourced language pairs for calibration (e.g $EN \leftrightarrow DE$ and $EN \leftrightarrow FR$.)

Neurons that are found to be important under both 1 and 2 are considered to be English-language-specific. The values of it's respective weights are not changed during fine-tuning. Whereas neurons that are found to be unimportant under 1 are zero-d out during inference and fine-tuning. Lastly neurons that are found to be important under 1 only are fine-tuned and used during inference.

We mention in more details the neuron ranking methods we considered during this work and briefly describe the rationale behind them.

3.2.1 Gradient Analysis

It is a simple importance measure where we compute the gradient of the network output w.r.t to the input features (Lei et al., 2016).

$$E_{gradient}(X, c) = \nabla f(X)_c \tag{1}$$

where f(X) is the model logits.

We adapt this method to our needs by capturing the gradient of the output w.r.t to a neuron's output instead of the actual input features. Given that the gradient is computed per-timestep we use the *average of absolute* (see eq. 2) of the value of the gradient per-timestep and that value is averaged across different input sentences. We also experiment with *max of absolute* (eq. 3).

$$Importance(X, n) = \frac{\sum_{t=0}^{t=|X|-1} |\nabla f(X)_{c_{ti}}|}{|X|} \tag{2}$$

$$Importance(X, n) = \frac{\max_{t=0}^{t=|X|-1} |\nabla f(X)_{c_{ti}}|}{|X|} \tag{3}$$

where: c_{ti} is the selected output token at timestep t_i, $|X|$ is the length of input sequence X and n is the neuron we are interested in.

In practice we found it better, empirically, to use intrinsic functions (perplexity) to compute neuron importance than to use the cross-entropy loss (see table 3) .

3.2.2 Activation Magnitude

In this method we track the magnitude of the output of the neuron of interest across different time steps of the deep neural network's execution. We use similar strategies to aggregate these values per input sequence to what was used in 3.2.1.

3.2.3 LASS

Proposed in (Lin et al., 2021), this method ranks the *weights* of the model by computing the change that weight exhibits after fine-tuning a multilingual model on a certain language pair. The rationale here being that weights that exhibit change the most during fine-tuning are language-dependent since fine-tuning on the language-pair had the most impact on their values. Formally, given the parameters of a multilingual model θ_0, a language pair $s_i \rightarrow t_i$:

1. Finetune θ_0 on $D_{s_i \rightarrow t_i}$ (i.e the dataset of the language pair of intereset), it is assumed that the resulting set of parameters (referred to as $\theta_{s_i \rightarrow t_i}$ would have amplified the set of language-dependent weights).

2. The *importance* of each weight is computed as:

$$importance(\theta_j) = \theta_{s_i \rightarrow d_i j} - \theta_{0j} \tag{4}$$

where θ_j denotes the j^{th} weight of the set of the deep neural network's weights.

The importance of the *neuron* is computed as the average of the importance scores of it's weights.

3.3 Vocab Substitution

Depending on the relation between the HRL and LRL, vocab substitution can provide a boost to both convergence speed and quality of translation. Specifically when the HRL and the LRL are written in two different scripts. This hinders learning since the *textual representation* of the two languages is different despite being phonetically potentially similar, and in some cases mutually intelligible. We adapt the vocab substitution algorithm from (Garcia et al., 2021).

Our method assumes the existence of a small $HRL \leftrightarrow LRL$ training corpus. We also assume the existence of sufficiently large LRL and HRL monolingual corpora.

1. Given the original multilingual vocab (V_m), we use the HRL monolingual corpus to find the subset that represent the HRL (V_{hrl}). We remove elements that do not occur in the corpus more than a certain threshold. This is to help mitigate the noise in the corpus.

2. We use the LRL monolingual corpus to train a new vocab set (V_{lrl}). We experiment with different vocab sizes, the only constraint being $|V_{lrl}| <= |V_{hrl}|$.

3. Next we try to learn an appropriate mapping f, between the elements of ($V_{LRL} \rightarrow V_{lrl}$). To do so we train an alignment model using the parallel corpus.

4. Finally we apply the vocab substitution as follows:

   (a) Elements of V_{lrl} that belong to V_m are kept as is.

   (b) We sort the alignments extracted from the parallel corpus discerningly by the number of occurrences. We refer to the extracted alignments as a set of (e_{hrl}, e_{lrl}), where e_{hrl} is vocabulary element that belongs to the HRL and e_{lrl} is a vocabulary element that belongs to the LRL.

   (c) for each pair (e_{hrl}, e_{lrl}):

i. if e_{lrl} has already been placed in V_m, we skip this pair, since we definitely encountered a better alignment pair.

ii. else we replace e_{hrl} by e_{lrl} in V_m. i.e:

$$V_m = (V_m - \{e_{hrl}\}) \cup \{e_{lrl}\} \tag{5}$$

(d) Elements of V_{lrl} that are still un-assigned replace random elements of V_{hrl}.

3.4 Fine-tune On LRL data

Starting from HRL-fine-tuned model:

- We zero-out the weights of neurons that have been considered to be unimportant in order to nullify the output of that specific neuron.
- We Freeze the weights of neurons that have been considered to be English-specific. The rationale is that these weights are well-trained, so keeping their output while freezing the weights is sensible. We also experiment with jointly training them.
- The remaining neurons (important neurons that are HRL-specific). Are actively modified during fine-tuning to adapt to differences between the HRL and LRL.

4 Performance Evaluation

We chose to adapt MBART50 (Tang et al., 2020) to Tajik language (LRL) with Persian being its closest HRL language. We collect the training data for both languages from OPUS [1]. Tajik and Persian pose an especially interesting challenge since they are mutually intelligible, but written in different scripts (Cyrillic and Perso-Arabic respectively). For either language pair we use FLORES-dev as validation and FLORES-devtest for evaluation (Goyal et al., 2022). All of the computations were performed on a single Tesla T4 GPU with 16 GB of RAM.

To compare our results to (Xia et al., 2019) we train a $Persian \leftrightarrow Tajok$ MT model. We collect the training data from OPUS and use FLORES "dev" and "devtest" as validation and test sets respectively.

4.1 Data Quality

We apply some basic rule-based filtering on the data.

1. Punctuation Ratio: We remove sentence-pairs where either side has a punctuation ratio $>$ 0.5. We adapt this filter from (Fan et al., 2020) [2].
2. Length Ratio Filtering: We only keep the sentence-pairs where the longer sentence is less than three-times the shorter one. With sentence length being determined via number of characters. This method has also been used in literature (Pinnis, 2018).
3. Script Verification: This step verifies that each sentnece is *mostly* written in its respective language's script (Latin for English, Perso-Arabic for Persian..etc). We use unicodedata2 [3] to determine the script of each character (we exclude Numeric/Punctuation characters since they are mostly script-agnostic).

See Table 1 for a detailed recount of the effect of each of the filters on the amounts of data of both language pairs.

[1] https://opus.nlpl.eu/

[2] https://github.com/facebookresearch/fairseq/blob/main/examples/m2m_100/process_data/remove_too_much_punc.py

[3] https://gist.githubusercontent.com/anonymous/2204527/raw/e940a6862de340cf23d7653969e181427176fc9b/unicodedata2.py

Filter Step	FA - EN (M)	TG - EN (M)
Original	12.8	0.268
+Punctuation Ratio	10.5	0.2
+ Length Ratio	9.36	0.194
+ Script Verification	9.35	0.19

Table 1: Number of sentences per language-pair after each filtering step.

Experiment	FA - EN	EN-FA	TG - EN	EN-TG
MBART50-Large	20.6	12.9	0.18	0.01
Full data Finetune	28.6	15.1	15.3	**9.3**
Filtered Data Finetune	**31.4**	**15.8**	**16.01**	9.1

Table 2: BLEU scores of MBART model pre-fine-tuning and post-fine-tuning.

4.2 HRL Fine-tuning

To verify that applied data filtering techniques did not harm the performance we examine its effect by fine-tuning pre and post data filtering (see table 2). We use a learning rate of 0.00003 with Polynomial decay learning rate scheduler. We set the patience window to 15 validation-runs while setting the validation interval to 500 updates. We set the batch size to 1000 tokens.

4.3 Vocab Substitution

We start by building a sentencepiece model for Tajik(TG) only. We set the vocab size to 4k although this might be a hyper-parameter that would require tuning in other scenarios/language-pairs. As stated above we collect the $FA \leftrightarrow TG$ training corpus from OPUS. We segment the Persian (FA) side using the MBART sentencepiece model while the TG side is segmented using the newly trained sentencepiece model. The parallel corpus is then used to train an alignment model and then extract piece-wise alignments. We use Fastalign [3] to train the alignment model.

To quantify the effect of vocab substitution on Tajik sentences, we compute the average number of setnencepiece tokens per sentence of the Tajik side of the validation set using both the new sentencepiece model and the original MBART setnencepiece model. We find that on average the sequence length of a Tajik sentence using the original MBART model is approximately 64.7 tokens/sentence whereas using the new model this value drops to 45.6 . This means that Tajik sentences on average were 30% shorter when using the new model. This in addition to

[3] https://github.com/clab/fast_align

Experiment	TG - EN	EN - TG
Direct Fine-tuning (No selection)	16.01	9.1
+ Vocab Substitution	16.9	9.5
Mixed Fine-tuning (FA/TG - EN)	16.21	7.6
Pivot on HRL (FA)	7.3	3.49
Mono Adapters + Fine-tune	15.7	9.2
Neuron Activation + vocab sub	17.3	9.4
LASS + vocab sub	18.75	9.2
Loss Gradient Analysis + vocab sub	18.9	10.2
Perplexity Gradient Analysis + vocab sub	**19.03**	**10.7**

Table 3: Evaluating our method against multiple baselines on $TG \leftrightarrow EN$ BLEU score.

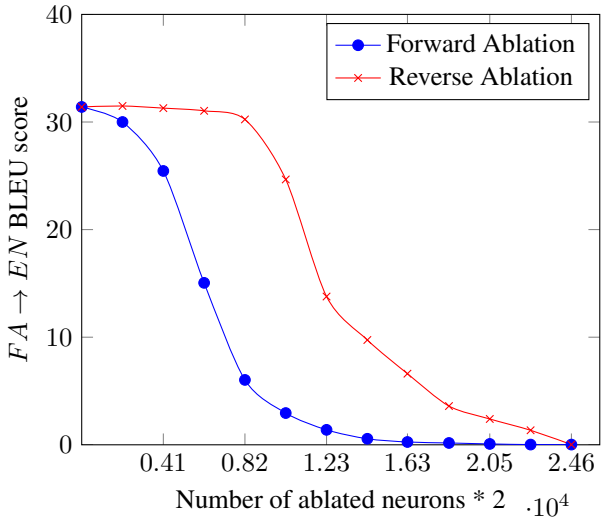

Figure 1: Forward vs Backward ablation on Encoder-Decoder attention Neurons, X-axis represents the number of zero-d out neurons and the Y-axis is the BLEU score.In Forward Ablation curve we remove-according to our ranking- the most important neurons first, while reverse ablation means we start with the most unimportant

quality of translation gains described in table 3 has also sped up training by approximately 15% in terms of time. We also observe that the oov-rate (Number of $< UNK >$ Tokens divided by the Total Number of Tokens) drops from 1.3% using the original MBART sentencepiece model to just 0.035% using the new sentencepiece model. These statistics were also collected by tokenizing the Tajik side of the validation set.

4.4 Neuron Selection

As detailed in section 3.2, we experiment with multiple methods of neuron ranking. We conduct a neuron-ablation study on the Gradient Analysis ranking, specifically for $FA \rightarrow EN$ fine-tuned model and examine the effect on BLEU score (see figure below) to verify our implementation. We limit the ablation study to the output neurons of the encoder-decoder attention module of all 12 layers (a total of 12288 neurons). The gradient analysis was performed on the validation set, and we observe that computing the gradient of the *perplexity* of the output sentence achieves better performance compared to the cross-entropy loss function.

This was not conducted for LASS nor Activation magnitude since their implementations have already been verified, we instead run end-to-end comparisons and determine which is best using BLEU score. Table 3 shows a comparison between the three aforementioned ranking methods. We observe that gradient analysis slightly outperforms LASS, while both of them show significant gains when compared to tracking the magnitude of each neuron's output. We also find that when selecting the top 20% most important neurons the intersection between using the cross-entropy loss and perplexity functions is around 96% of the selected neurons, and that is why it is to be expected that difference in BLEU score between the two methods is mostly less than 1 point. We did however find it consistently better to use perplexity as opposed to cross-entropy loss.

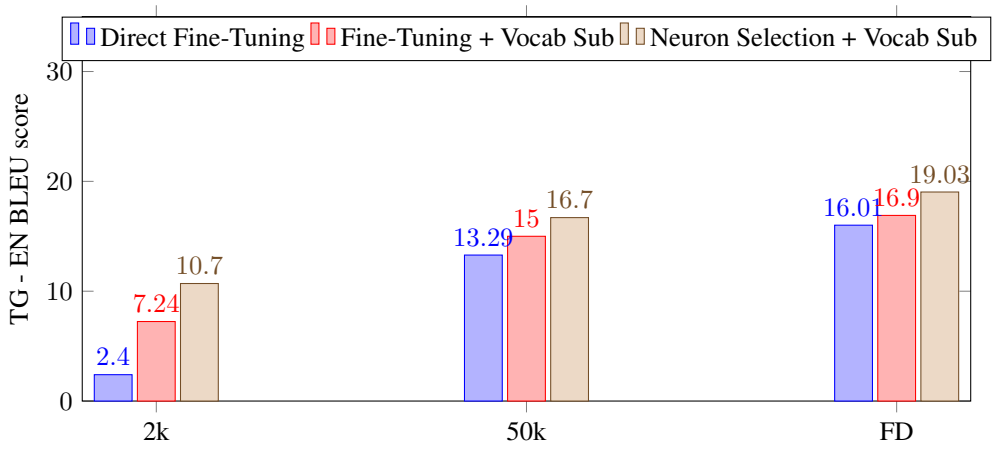

Figure 2: Effect of varying data-set size on BLEU scores. FT denotes direct Fine-tuning, VS is vocab substitution, NS is neuron selection and FD denotes full data-set size.

4.5 LRL Fine-Tuning

Using the neuron ranking methods described above, we fine-tune the top 20% most important neurons. It's also worth noting that for $Tajik \rightarrow English$ we freeze the decoder (other than the encoder-decoder attention module which is subject of neuron selection), while for $English \rightarrow Tajik$ we freeze the encoder. The idea here is that the model has been trained on a lot of English data, hence we limit the training to the encoder for $Tajik \rightarrow English$ and the decoder for $English \rightarrow Tajik$. This stems from the fact that when generating $Tajik$ we need not modify the encoder since it has already been well trained to consume English sentences we only need to train the decoder to consume the encoder representations and generate Tajik sentences and vice versa . The results are described in table 3.

We also conduct another series of experiments to examine the effect of reducing the training dataset size on the performance of our method and compare it to directly fine-tuning all of the neurons of the model. We randomly select two subsets from $TG \leftrightarrow EN$ training set of sizes 2k and 50k.

In figure 2 below we show the effect of varying the training set data size across different fine-tuning techniques.

5 Conclusion And Future Work

In this section we discuss the results detailed in the previous section and propose possible extension to the analysis described throughout the paper.

As detailed in table 3 we find that our method achieves better performance compared to earlier approaches. We also find that as we *emulate* smaller training dataset sizes the relative improvement in performance between our method and simply fine-tuning the whole model on the LRL continues to grow as hypothesized. Namely the gap between our method and direct fine-tuning grows from roughly 3 BLEU points at full data to 8.2 points when we limit the data to 2k samples.

For future work we propose extending this analysis horizontally by experimenting with more than one HRL-LRL, and vertically by applying the analysis on other available MNMT. We find that M2M might be a good candidate for this, and is especially interesting since it already has language-dependant parameters. Another interesting area of research would be how

to integrate self-supervised learning into this setup. Specifically how will our method fair given zero parallel data but assuming the abundance of monolingual data.

References

Bapna, A. and Firat, O. (2019). Simple, scalable adaptation for 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 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Berard, A. (2021). Continual learning in multilingual NMT via language-specific embeddings. *CoRR*, abs/2110.10478.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, O., Wenzek, G., Chaudhary, V., Goyal, N., Birch, T., Liptchinsky, V., Edunov, S., Grave, E., Auli, M., and Joulin, A. (2020). Beyond english-centric multilingual machine translation. *arXiv preprint*.

Garcia, X., Constant, N., Parikh, A. P., and Firat, O. (2021). Towards continual learning for multilingual machine translation via vocabulary substitution. *CoRR*, abs/2103.06799.

Goyal, N., Gao, C., Chaudhary, V., Chen, P.-J., Wenzek, G., Ju, D., Krishnan, S., Ranzato, M., Guzmán, F., and Fan, A. (2022). The Flores-101 Evaluation Benchmark for Low-Resource and Multilingual Machine Translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. *CoRR*, abs/1606.07947.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Lakew, S. M., Karakanta, A., Federico, M., Negri, M., and Turchi, M. (2019). Adapting multilingual neural machine translation to unseen languages. *CoRR*, abs/1910.13998.

Lei, T., Barzilay, R., and Jaakkola, T. S. (2016). Rationalizing neural predictions. *CoRR*, abs/1606.04155.

Lin, Z., Wu, L., Wang, M., and Li, L. (2021). Learning language specific sub-network for multilingual machine translation. *CoRR*, abs/2105.09259.

Neubig, G. and Hu, J. (2018). Rapid adaptation of neural machine translation to new languages. *CoRR*, abs/1808.04189.

Philip, J., Berard, A., Gallé, M., and Besacier, L. (2020). Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Pinnis, M. (2018). Tilde's parallel corpus filtering methods for WMT 2018. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 939–945, Belgium, Brussels. Association for Computational Linguistics.

Tang, Y., Tran, C., Li, X., Chen, P., Goyal, N., Chaudhary, V., Gu, J., and Fan, A. (2020). Multilingual translation with extensible multilingual pretraining and finetuning. *CoRR*, abs/2008.00401.

Tran, C., Bhosale, S., Cross, J., Koehn, P., Edunov, S., and Fan, A. (2021). Facebook AI WMT21 news translation task submission. *CoRR*, abs/2108.03265.

Xia, M., Kong, X., Anastasopoulos, A., and Neubig, G. (2019). Generalized data augmentation for low-resource translation. *CoRR*, abs/1906.03785.