Beyond English-Centric Multilingual Machine Translation

Angela Fan[*], **Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli**[†], **Armand Joulin**[†‡]
Facebook AI, *LORIA

Abstract

Existing work in translation demonstrated the potential of massively multilingual machine translation by training a single model able to translate between any pair of languages. However, much of this work is English-Centric by training only on data which was translated from or to English. While this is supported by large sources of training data, it does not reflect translation needs worldwide. In this work, we create a true Many-to-Many multilingual translation model that can translate directly between any pair of 100 languages. We build and open source a training dataset that covers thousands of language directions with supervised data, created through large-scale mining. Then, we explore how to effectively increase model capacity through a combination of dense scaling and language-specific sparse parameters to create high quality models. Our focus on non-English-Centric models brings gains of more than 10 BLEU when directly translating between non-English directions while performing competitively to the best single systems of WMT. We open-source our scripts so that others may reproduce the data, evaluation, and final M2M-100 model `here`.

1. Introduction

Multilingual Machine Translation (MMT) aims to build a single model to translate between any pair of languages. Neural network models have been very successful for bilingual machine translation (Bahdanau et al., 2014; Gehring et al., 2017; Vaswani et al., 2017) and more recently, neural MMT models have shown promising results (Firat et al., 2016; Zhang et al., 2020). Multilingual translation models factorize computation when translating to many languages and share information between similar languages, which benefits low resource directions (Arivazhagan et al., 2019) and enables zero-shot translation (Gu et al., 2019).

However, in the past, these systems have not performed as well as bilingual models when trained on the same language pairs (Johnson et al., 2017), as model capacity necessarily

∗. Corresponding Author. Email: `angelafan@fb.com`

†. Equal contribution. Order determined with coin flip.

‡. Holger Schwenk, Onur Celebi, Vishrav Chaudhary, Ahmed El-Kishky, Angela Fan, Edouard Grave, Zhiyi Ma, and Guillaume Wenzek worked on large-scale data mining, including improvements to fasttext, LASER, CCAligned, and CCMatrix. Siddharth Goyal worked on backtranslation. Michael Auli, Mandeep Baines, Shruti Bhosale, Tom Birch, Sergey Edunov, Angela Fan, Naman Goyal, Siddharth Goyal, and Vitaliy Liptchinsky worked on model scaling and scaling infrastructure. Michael Auli, Shruti Bhosale, Sergey Edunov, Angela Fan, Edouard Grave, Armand Joulin, Zhiyi Ma, and Holger Schwenk worked on model and experimental design. Michael Auli, Ahmed El-Kishky, Angela Fan, Edouard Grave, Armand Joulin, Zhiyi Ma, and Holger Schwenk wrote the paper.

must be split between many languages (Arivazhagan et al., 2019). This has been alleviated by increasing model capacity (Aharoni et al., 2019; Zhang et al., 2020), but increased model size also necessitates larger multilingual training datasets which are laborious and difficult to create. To ease this challenge, most prior work has focused on *English-Centric* datasets and models which translate from and to English but not between non-English languages. This English-Centric bias in the data and resulting models is not reflective of how people use translation and empirically leads to lower performance for non-English translation directions.

In this work, we create more diverse multilingual machine translation models by building a large-scale Many-to-Many dataset for 100 languages. We considerably reduce the complexity of this task through the automatic construction of parallel corpora (Artetxe and Schwenk, 2018b; Schwenk et al., 2019) with a novel data mining strategy that exploits language similarity to avoid mining all directions. We also leverage backtranslation to improve the quality of our model on zero-shot and low resource language pairs. Overall, we build the first true Many-to-Many dataset comprising 7.5B training sentences for 100 languages, providing direct training data for thousands of translation directions.

The quantity of data in a Many-to-Many dataset increases quadratically with the number of languages, making neural networks with standard capacity underfit rapidly. To that effect, we leverage progress in scaling (Kaplan et al., 2020; Arora et al., 2018) to train models that are over 50 times larger than current bilingual models with model parallelism (Huang et al., 2019; Shoeybi et al., 2019). Even with these tools, scaling the number of parameters hardly follows the quadratic increase in data induced by the Many-to-Many setting, and we propose several scaling strategies tailored to the specificities of our problem. In particular, we consider a deterministic mixture-of-experts strategy to split the model parameters into non-overlapping groups of languages which we train with a novel re-routing strategy. Language specific mixture-of-experts also reduce the need to densely update parameters and are more parallelizable in a multi-machine setting. Overall, combining these strategies allows us to scale the capacity of the models to a size of 15.4B parameters and still train them efficiently on hundreds of GPUs.

The resulting method allows us to scale Transformers and directly translate between 100 languages without pivoting through English at a performance that is competitive with bilingual models on many competitive benchmarks, including WMT. Figure 1 illustrates our data mining strategy as well as our model architecture. This paper is organized as follows: first, we introduce several standard components of modern machine translation and explain how they apply in the multilingual setting (§ 2), then describe our strategy to scale the number of language pairs to create a Many-to-Many dataset (§ 3). We then systematically compare this Many-to-Many dataset to an English-Centric approach (§ 4). Next, we incorporate increased model scale through both dense scaling and sparse mixture-of-experts (§ 5). Finally, we end with a thorough analysis, including human evaluation, of the quality of our 100x100 Many-to-Many translation system (§ 6).

2. Preliminaries

In this work, we investigate how we can best translate from 100 languages to 100 languages, or 9900 directions, using a single model. We describe our starting point in this section, and provide preliminary context on Transformer-based neural machine translation models.

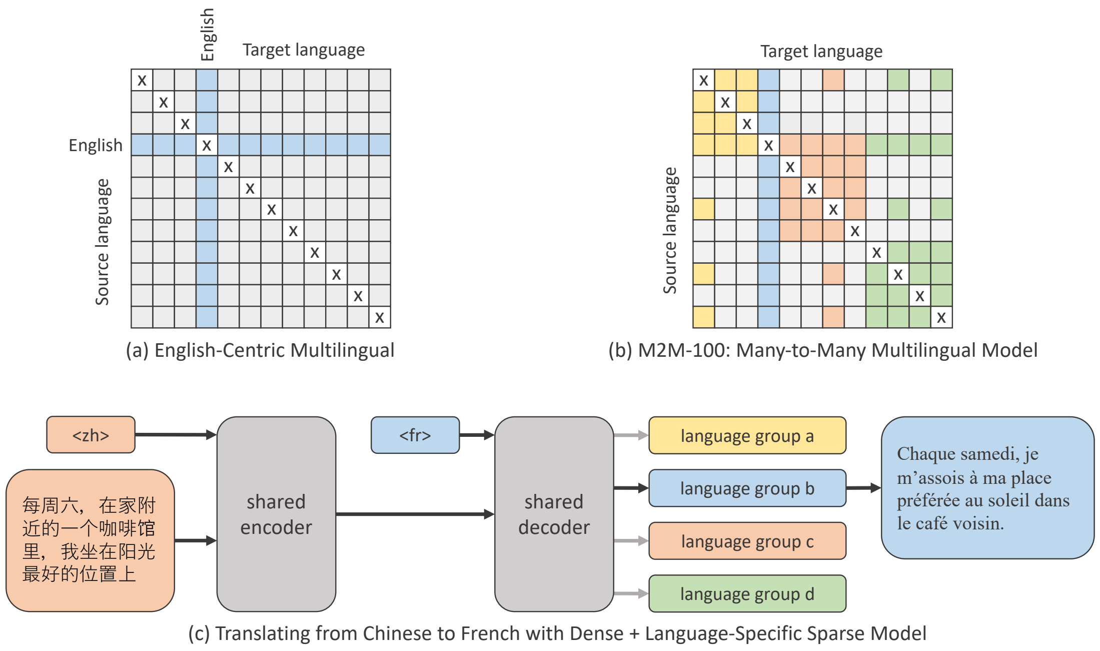

English
Target language
English
Source language
(a) English-Centric Multilingual
Target language
Source language
(b) M2M-100: Many-to-Many Multilingual Model
<zh>
每周六，在家附近的一个咖啡馆里，我坐在阳光最好的位置上
shared encoder
<fr>
shared decoder
language group a
language group b
language group c
language group d
Chaque samedi, je m'assois à ma place préférée au soleil dans le café voisin.
(c) Translating from Chinese to French with Dense + Language-Specific Sparse Model

Figure 1: **Summary of our Many-to-Many dataset and multilingual model**. English-Centric data (top left) only contains training data to and from English, where as our Many-to-Many multilingual setting (top right) contains data directly through various different directions. Our proposed model, M2M-100, combines dense and sparse language-specific parameters to translate directly between languages (bottom).

Sequence-to-sequence models are trained on pairs of sequences, conditioning on an input sequence to produce an output sequence. Each sentence is split into tokens, that can be words or characters, resulting in pairs of sequences $(w_1, \ldots, w_S)$ and $(v_1, \ldots, v_T)$. Most machine translation systems are trained by maximizing the probability of the target sequence, given the source sentence and the target language ℓ_t:

$$P(v_1,\ \ldots,\ v_T \mid w_1,\ \ldots,\ w_S,\ \ell_t)$$

Modern neural machine translation systems are based on several standard components, namely a subword segmentation method and an encoder-decoder architecture called a Transformer. We describe these components in the context of multilingual translation.

Segmentation with SentencePiece. The input and output of translation systems are sequences of tokens. These tokens are units from a dictionary built with the goal to reconstruct any sentence in any language. Using words as base units is challenging, as it leads either to vocabularies with poor coverage or to large vocabularies. This is especially true in the multilingual setting. Another limitation of word-based systems are languages that are not naturally split into words, like Thai. An alternative approach is to use *subword* units, which are learned directly from data (Sennrich et al., 2015; Kudo and Richardson, 2018). We use SentencePiece[1] as it was designed to work with languages with no segmentation, making it

1. `https://github.com/google/sentencepiece`

particularly suited to our setting. We train a model with 0.9995 character coverage to have sufficient representation of character-based languages.

Creating a Multilingual Dictionary. SentencePiece produces subword units depending on their frequency in the training dataset. Naively applying it to our corpora would result in low resource languages and languages written in less frequent scripts being underrepresented in the resulting dictionary. Randomly sampling data favors overrepresented languages because the probability of picking language ℓ is proportional to its number of sentences, D_ℓ, i.e., $p_\ell = \frac{D_\ell}{\sum_i D_i}$. We circumvent this problem by adding monolingual data for low resource languages and by using temperature sampling with $T = 5$. More precisely, the probability p_ℓ is rescaled to $p_\ell^{\frac{1}{T}}$ where the temperature T controls the distribution. For example, setting T to 1 gives the original data distribution. The resulting dictionary contains 128k unique tokens that are well distributed across languages, as shown in Appendix A.

2.1 Transformers

Our multilingual machine translation model is based on the Transformer sequence-to-sequence architecture, which is composed of two modules: the encoder and the decoder (Vaswani et al., 2017). The encoder transforms the source token sequence into a sequence of embeddings of the same length. Then, the decoder sequentially produces the target sentence, token by token, or autoregressively. More precisely, the encoder takes the sequence of tokens $W = (w_1, \ldots, w_S)$ and the source language ℓ_s, and produces a sequence of embeddings $H = (h_1, \ldots, h_S)$, which are then fed to the decoder with the target language ℓ_t to produce the sequence of target tokens $V = (v_1, \ldots, v_T)$ sequentially, i.e.,

$$\begin{aligned} H &= \texttt{encoder}(W,\ \ell_s), & (1)\\ \forall i \in [1, \ldots, T],\ v_{i+1} &= \texttt{decoder}(H,\ \ell_t,\ v_1,\ \ldots,\ v_i). & (2)\end{aligned}$$

Both the encoder and decoder are composed of the same type of layers, called Transformer layers. Each Transformer layer takes a sequence of vectors as input and outputs a sequence of vectors. In the encoder, transformer layers are composed of two sublayers, a self-attention and a feed-forward layer. These are applied sequentially and are both followed by a residual connection (He et al., 2015) and layer normalization (Ba et al., 2016):

$$\begin{aligned} Z &= \texttt{norm}\left(X + \texttt{self-attention}(X)\right), & (3)\\ Y &= \texttt{norm}\left(Z + \texttt{feed-forward}(Z)\right). & (4)\end{aligned}$$

The self-attention layer is an attention layer that updates each element of the sequence by looking at the other elements, while the feed-forward layer (FFN) passes each element of the sequence independently through a 2-layer MLP. In the decoder, there is an additional third sublayer, between the self-attention and the feed-forward, which computes attention over the output of the encoder. We refer the reader to Vaswani et al. (2017) for details of these layers.

Target language token. The Transformer architecture has been designed for the bilingual case, where the target language is fixed. In the case of multilingual machine translation, the target language is not fixed, and several strategies can be applied to condition the network to produce a sentence in the desired target language. Similarly to Ha et al. (2016) and Johnson et al. (2017), we add a special token in the encoder indicating the source language and a special token in the decoder indicating the target language.

Training. Our starting point for improving massively multilingual translation models is a large Transformer model, with 12 Encoder and 12 Decoder layers, with 8192 FFN size and 1024 embedding dimension. We share the weight matrices of the input and output embeddings. The total parameter count is 1.2B. We train with the Adam optimizer (Kingma and Ba, 2015) and warmup first for 4000 updates, with label smoothing 0.1 (Szegedy et al., 2015; Pereyra et al., 2017). For regularization, we tune the dropout parameter between $\{0.1, 0.2, 0.3\}$. To stabilize the training of deeper Transformers, we train with LayerDrop (Fan et al., 2019) 0.05 and pre-normalization (Nguyen and Salazar, 2019).

To train with billions of sentences, we split the training data into 256 different shards to manage memory consumption. However, directly dividing mid and low resource languages into shards would reduce the variability of each shard's data for mid or low resource languages. Imagine the case where there are only 100 sentences of a language direction per shard — the model would easily overfit. Thus, each language is divided into a different number of shards based on resource level, such that high resource languages have more shards and the lowest resource languages only have one shard. Subsequently, lower resource shards are replicated until the full number of shards is reached.

Generation. Unless otherwise specified: for all results, we report single models with no checkpoint averaging, use beam search with beam 5, and do not tune length penalty.

3. Building a Many-to-Many Parallel Dataset for 100 Languages

In this section, we provide an overview of our Many-to-Many setting: the selection of the 100 languages, the evaluation benchmarks, and the construction of a large-scale training set through data mining (Artetxe and Schwenk, 2018b) and backtranslation (Sennrich et al., 2016a) that provides training data thousands of directions.

3.1 Creating a Multilingual Benchmark

The first step of establishing a Many-to-Many dataset is to select 100 languages for which there already exist high-quality, annotated datasets that can be used for model evaluation.

3.1.1 Language Selection

We consider several factors to select which languages to focus on. First, we include widely-spoken languages from geographically diverse language families. We cover a diversity of scripts and resource levels (as shown in Table 1) to have high coverage of languages worldwide. Second, we use languages for which public evaluation data exists, as we must be able to quantify model performance. Lastly, we only use languages for which monolingual data is available, as monolingual data is a critical resource for large-scale mining. Combining these three criteria results creates our full list of 100 languages, summarized in Table 1.

ISO	Language	Family	Script	ISO	Language	Family	Script
af	Afrikaans	Germanic	Latin	ja	**Japanese**	Japonic	Kanji; Kana
da	Danish	Germanic	Latin	ko	**Korean**	Koreanic	Hangul
nl	**Dutch**	Germanic	Latin	vi	**Vietnamese**	Vietic	Latin
de	**German**	Germanic	Latin	zh	**Chinese Mandarin**	Chinese	Chinese
en	**English**	Germanic	Latin	bn	**Bengali**	Indo-Aryan	Eastern-Nagari
is	Icelandic	Germanic	Latin	gu	Gujarati	Indo-Aryan	Gujarati
lb	Luxembourgish	Germanic	Latin	hi	**Hindi**	Indo-Aryan	Devanagari
no	Norwegian	Germanic	Latin	kn	Kannada	Tamil	Kannada
sv	**Swedish**	Germanic	Latin	mr	Marathi	Indo-Aryan	Devanagari
fy	Western Frisian	Germanic	Latin	ne	Nepali	Indo-Aryan	Devanagari
yi	Yiddish	Germanic	Hebrew	or	Oriya	Indo-Aryan	Odia
ast	Asturian	Romance	Latin	pa	Panjabi	Indo-Aryan	Gurmukhi
ca	Catalan	Romance	Latin	sd	Sindhi	Indo-Aryan	Persian Devanagari
fr	**French**	Romance	Latin	si	Sinhala	Indo-Aryan	Sinhala
gl	Galician	Romance	Latin	ur	Urdu	Indo-Aryan	Arabic
it	Italian	Romance	Latin	ta	**Tamil**	Dravidian	Tamil
oc	Occitan	Romance	Latin	ceb	Cebuano	Malayo-Polyn.	Latin
pt	**Portuguese**	Romance	Latin	ilo	Iloko	Philippine	Latin
ro	Romanian	Romance	Latin	id	**Indonesian**	Malayo-Polyn.	Latin
es	**Spanish**	Romance	Latin	jv	Javanese	Malayo-Polyn.	Latin
be	Belarusian	Slavic	Cyrillic	mg	Malagasy	Malayo-Polyn.	Latin
bs	Bosnian	Slavic	Latin	ms	Malay	Malayo-Polyn.	Latin
bg	Bulgarian	Slavic	Cyrillic	ml	Malayalam	Dravidian	Malayalam
hr	Croatian	Slavic	Latin	su	Sundanese	Malayo-Polyn.	Latin
cs	**Czech**	Slavic	Latin	tl	Tagalog	Malayo-Polyn.	Latin
mk	Macedonian	Slavic	Cyrillic	my	Burmese	Sino-Tibetan	Burmese
pl	**Polish**	Slavic	Latin	km	Central Khmer	Khmer	Khmer
ru	**Russian**	Slavic	Cyrillic	lo	Lao	Kra-Dai	Thai; Lao
sr	Serbian	Slavic	Cyrillic; Latin	th	Thai	Kra-Dai	Thai
sk	Slovak	Slavic	Latin	mn	Mongolian	Mongolic	Cyrillic
sl	Slovenian	Slavic	Latin	ar	**Arabic**	Arabic	Arabic
uk	Ukrainian	Slavic	Cyrillic	he	**Hebrew**	Semitic	Hebrew
et	Estonian	Uralic	Latin	ps	Pashto	Iranian	Arabic
fi	**Finnish**	Uralic	Latin	fa	**Farsi**	Iranian	Arabic
hu	**Hungarian**	Uralic	Latin	am	Amharic	Ethopian	Ge'ez
lv	Latvian	Baltic	Latin	ff	Fulah	Niger-Congo	Latin
lt	**Lithuanian**	Baltic	Latin	ha	Hausa	Afro-Asiatic	Latin
sq	Albanian	Albanian	Latin	ig	Igbo	Niger-Congo	Latin
hy	Armenian	Armenian	Armenian	ln	Lingala	Niger-Congo	Latin
ka	Georgian	Kartvelian	Georgian	lg	Luganda	Niger-Congo	Latin
el	**Greek**	Hellenic	Greek	nso	Northern Sotho	Niger-Congo	Latin
br	Breton	Celtic	Latin	so	Somali	Cushitic	Latin
ga	Irish	Irish	Latin	sw	**Swahili**	Niger-Congo	Latin
gd	Scottish Gaelic	Celtic	Latin	ss	Swati	Niger-Congo	Latin
cy	Welsh	Celtic	Latin-Welsch	tn	Tswana	Niger-Congo	Latin
az	Azerbaijani	Turkic	Latin; Cyrillic Persian	wo	Wolof	Niger-Congo	Latin
ba	Bashkir	Turkic	Cyrillic	xh	Xhosa	Niger-Congo	Latin
kk	Kazakh	Turkic	Cyrillic	yo	Yoruba	Niger-Congo	Latin
tr	**Turkish**	Turkic	Latin	zu	Zulu	Niger-Congo	Latin
uz	Uzbek	Turkic	Latin; Cyrillic	ht	Haitian Creole	Creole	Latin

Table 1: **100 Languages grouped by family.** For each language, we display the ISO code, language name, language family, and script. Languages in bold are *bridge languages* (*Malayo-Polyn.* stands for *Malayo-Polynesian*).

3.1.2 Evaluation Benchmarks

We use publicly available evaluation datasets to evaluate the performance of all of our models. To cover our set of 100 languages and 2200 directions, we bring together data from a variety of sources. We describe each evaluation dataset below.

- **WMT** — The majority of language pairs from WMT go through English and the data is from the news domain. We consider data for 13 languages (Ondrej et al., 2017; Bojar et al., 2018; Barrault et al., 2019).

- **WAT** — The WAT competition covers Asian languages paired with English. We consider data for Burmese-English (Riza et al., 2016), which contains news articles. WAT contains many other evaluation directions, but many of those are covered by WMT or in a specific domain, so we focus on Burmese-English for WAT only.

- **IWSLT** — The IWSLT translation competition contains data from TED talks paired with English translations. We use data for 4 languages (Cettolo et al., 2017).

- **FLORES** — FLORES[2] (Guzmán et al., 2019) pairs two low resource languages, Sinhala and Nepali, with English in the Wikipedia domain.

- **TED** — The TED Talks dataset[3] (Ye et al., 2018) contains translations between more than 50 languages; most of the pairs do not include English. The evaluation data is n-way parallel and contains thousands of directions.

- **Autshumato** — Autshumato[4] is an 11-way parallel dataset comprising 10 African languages and English from the government domain. There is no standard valid/test split, so we use the first half of the dataset for valid and second half for test.

- **Tatoeba** — Tatoeba Challenge[5] covers 692 test pairs from mixed domains where sentences are contributed and translated by volunteers online. The evaluation pairs we use from Tatoeba cover 85 different languages.

We evaluate the quality of translations with BLEU (Papineni et al., 2002). We first detokenize all data, then apply standard tokenizers for each language before computing BLEU. For most languages, we use the `moses` tokenizer (Koehn et al., 2007).[6] For Chinese we use the SacreBLEU tokenizer (`tok zh`) and convert all traditional characters generated by the model to simplified characters using HanziConv[7] (Post, 2018),[8] for Indian languages

2. `https://github.com/facebookresearch/flores`
3. `https://github.com/neulab/word-embeddings-for-nmt`
4. `https://repo.sadilar.org/handle/20.500.12185/506`, CTexT® (Centre for Text Technology, North-West University), South Africa; Department of Arts and Culture, South Africa
5. `https://tatoeba.org/eng/`
6. `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl`
7. `https://github.com/berniey/hanziconv`
8. The evaluation datasets for Chinese usually contained simplified characters. However, our training data contains a mix of simplified and traditional characters, and thus the model could generate either. We convert the generated traditional Chinese characters to simplified for consistency.

we use the Indic NLP library (Kunchukuttan, 2020),[9] for Japanese we use Kytea,[10] for Thai we use PyThaiNLP (Phatthiyaphaibun et al., 2016),[11] for Arabic we use the QCRI Arabic Normalizer,[12] for Korean we use Mecab,[13] for Burmese we use the official segmentation tool provided by Ding et al. (2019), for Romanian we follow Sennrich et al. (2016b) and apply Moses tokenization, special normalization, and remove diacritics for Romanian texts,[14] and finally for Serbian we transliterate the output to Latin characters before computing BLEU.[15] We release the tokenization and evaluation scripts for reproducibility `here`[16]. We remove all data from all evaluation sets from our training sets.

3.2 Covering the Language Matrix by Mining Relevant Parallel Data

Supervised translation systems rely on large quantities of parallel sentences, which we refer to as bitext data, which are traditionally derived from human translations. Most existing bitext datasets go through English, with a few domain specific exceptions such as proceedings from international organizations (Koehn, 2005; Ziemski et al., 2016). These corpora are limited in size and domain, and an alternative is to *mine* parallel data (Resnik, 1999; Utiyama and Isahara, 2003) in large collections of monolingual data (Conneau et al., 2019; Wenzek et al., 2019). In this work, we leverage and extend the corpus provided by two of these mining projects: CCMatrix (Schwenk et al., 2019) and CCAligned[17] (El-Kishky et al., 2020). In the following, we describe our mining strategy and summarize the main ideas of CCMatrix and CCAligned. We refer the reader to the references for a detailed description of the approaches.

Mining parallel data with LASER. Mining parallel data consists of searching for sentences that could be potential translations in large monolingual corpora. This search requires a measure that captures the semantic similarity between sentences in different languages. Most recent methods build this similarity by comparing the embeddings from a neural network trained on multilingual data (Artetxe and Schwenk, 2018b; Chen et al., 2020; Kvapilíková et al., 2020). We focus on the embeddings generated by the LASER encoder, which enables the comparison of sentences in 94 different languages (Artetxe and Schwenk, 2018a). We then retrieve parallel corpora efficiently using FAISS indexing (Johnson et al., 2019). LASER embeddings generalize to unseen languages, like Asturian, allowing us to mine bitexts for 100 languages. The generic data mining pipeline consists of several steps: **(1)** a large corpus of text is preprocessed and divided into different languages, **(2)** candidate pairs of aligned sentences are embedded and stored in a index, **(3)** indexed sentences are compared to form potential pairs, **(4)** the resulting candidate pairs are filtered in post-processing.

9. `https://github.com/anoopkunchukuttan/indic_nlp_library`
10. `https://github.com/neubig/kytea`
11. `https://github.com/PyThaiNLP/pythainlp`
12. `http://alt.qcri.org/tools/arabic-normalizer/`
13. `https://pypi.org/project/python-mecab-ko/`
14. `https://github.com/rsennrich/wmt16-scripts/blob/master/preprocess/`
15. In Serbian, both Latin script and Cyrillic script are used, and often intermixed within a sentence in the evaluation data. As the target sentence could be in either script and it is not possible to predict the target script from the input, we transliterate before computing BLEU.
16. `https://github.com/pytorch/fairseq/tree/master/examples/m2m_100`
17. `http://www.statmt.org/cc-aligned`

CCMatrix Mining Approach. CCMatrix takes a global approach: all unique sentences in one language are compared with all unique sentences in another language. This *global mining* approach has the advantage of considering all possible documents when searching for the translation of a sentence. CCMatrix works on the large monolingual corpora in the 91 languages of CCNet (Wenzek et al., 2019), but at this scale, the global search is computationally demanding even with fast indexing from FAISS (Johnson et al., 2019). Thus, we apply it to a selected subset of relevant pairs, as detailed in § 3.2.1.

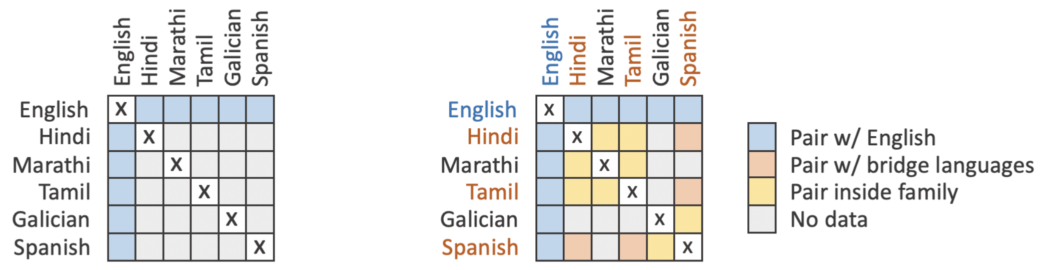
CCAligned Mining Approach. CCAligned avoids the scaling challenges of global mining by pre-selecting documents to compare. This *local mining* follows a hierarchical approach: first, document-level language identification along with various rules is applied to find whole documents that are likely to contain mutual translations (El-Kishky et al., 2020). Parallel sentences are then mined using LASER-based alignment within the paired documents only. Filtering (Chaudhary et al., 2019) is performed to remove unaligned data that exists because the original webpage did not have any parallel data, only partial parallel data, or other processing failures. One advantage of this approach is that it is very fast, scalable, and retrieves parallel sentences with high precision. Another is that each English document is aligned to many non-English documents — thus, mining non-English pairs can be quickly performed by joining non-English documents paired to the same source.

Postprocessing. We apply a filtering step to remove sentences of greater than 50% punctuation. The data is then deduplicated, and we remove any sentence that appears in any validation or test dataset – even if it is associated with another language pair. Finally, we apply length and language-specific filtering. The length filtering removes sentences that are too long – more than 250 subwords after segmentation with SPM – or with a length mismatch between the sentence and its translation – if the length ratio is greater than $3\times$. The language-specific filtering removes sentences that contain more than 50% of characters that have not been marked as core to the identified language – specifically, characters that are commonly used in the identified language with the exception of white space, numbers, punctuation, and Latin characters for non-Latin script languages.

3.2.1 Bridge Language Group Mining Strategy

Mining data for each and every language pair is prohibitive — previous work circumvents this issue by focusing only on the 99 pairs that go through English (Zhang et al., 2020). One alternative to the extensive computation required to mine all possible combinations of pairs is *sparse mining*, or mining only a select subset of pairs. A straightforward strategy is to *randomly* select pairs to mine, but this does not use any linguistic information on how languages are related and spoken around the world.

In this work, we propose an alternative based on language families and bridge languages that avoids exhaustively mining every possible pair. Our goal is to reduce the number of bitext pairs while preserving translation directions of practical interest. We first group all the 100 languages into 14 *language groupings*. All languages within a grouping are mined against each other. For instance, within the Indic language grouping, we mine all pairs of Bengali, Hindi, Marathi, Tamil, Urdu, and so on. The motivation for this strategy is two-fold. First, people living in areas that speak multiple languages in the same grouping tend to communicate a lot with each other and would benefit from high quality direct



(a) Existing Work with English-Centric Data

(b) Bridge Language Family Mining Strategy

Figure 2: **Depiction of an English-Only data mining setting compared to the Bridge Language Mining Strategy**. We display a data matrix, where languages are shown on the X and Y axes. Data is mined in one direction (such as Hindi to Marathi) and used to train bidirectionally.

translation. Second, systematically mining languages of the same grouping is helpful for training language-specific parameter models (see § 5.2).

For the most part, languages are grouped by linguistic similarity, e.g. Germanic, Slavic, or Malayo-Polynesian languages. However, the size of the resulting groupings varies greatly, resulting in less mined data for the languages in the smallest groupings. We further group languages by geographic and cultural proximity to reduce this discrepancy. For example, Uralic and Baltic languages are gathered into a single group to increase the quantity of mined data. The resulting groupings are shown in Table 1.

To connect languages across groupings, we define 1–3 *bridge languages* in each grouping, usually those with the most resources, such as Bengali, Hindi, and Tamil for the 12 languages in the Indo-Aryan family. All 26 bridge languages are highlighted in Table 1. These bridge languages are mined against all other bridge languages. Finally, all 100 languages are mined against English. We illustrate this mining strategy in Figure 2. On the left, we depict what many current approaches model: data only through English. On the right, we depict our Many-to-Many language matrix for several example languages. Compared to English-Centric, our dataset has far greater coverage of non-English, direct translation directions.

Training Data Statistics. In total, our final training dataset contains 7.5B parallel sentences, corresponding to 2200 directions. In Figure 3, we show all bridge languages and demonstrate how their associated training data is divided between translations with English, within a language grouping, or with bridge languages across language groupings. Of particular interest is the comparison between the additional Many-to-Many data and the data through English. We observe that 5–10 times more parallel data can be mined if using a Many-to-Many strategy, compared to an English-Centric one. This is particularly beneficial for mid- and low-resource languages.

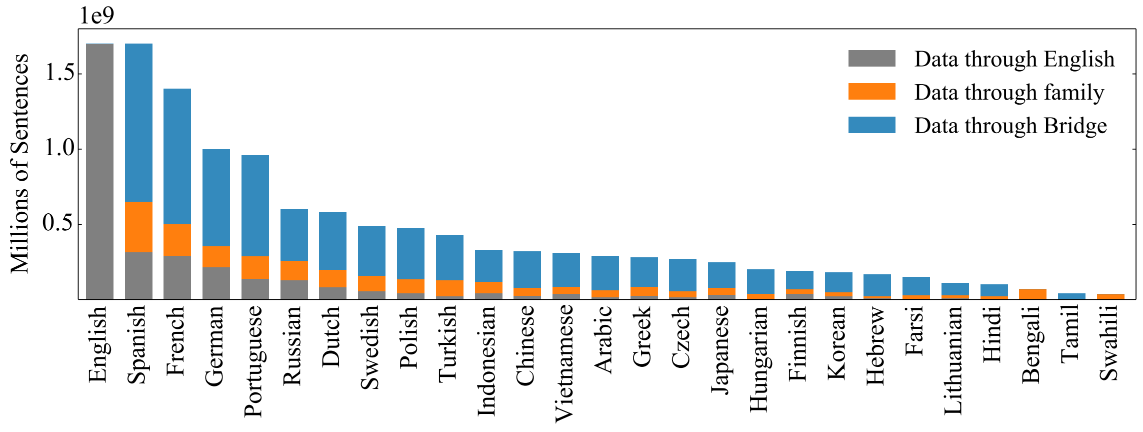


Figure 3: **Total Amount of Data through Bridge Languages on our 100x100 Training Corpus**. We depict the amount of data through English (gray), amount of data through a bridge language not counting English (orange), and amount of data through the language grouping not counting bridge languages (blue).

Model	All Avg	Supervised Low	Supervised Mid	Supervised High
Random 80%	11.9	3.6	16.1	31.5
Random 80% w/ En	16.3	8.9	22.4	36.6
Bridge Language, 80%	**17.2**	**10.4**	**23.2**	**37.4**

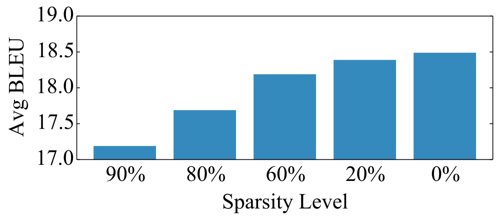


Table 2: **(left) Comparison of Sparse Mining Strategies**. We first hold the sparsity level fixed at 80% — compared to randomly selecting pairs to mine, the Bridge Language mining strategy performs very well. **(right) Bridge Language Strategy at Different Sparsity Levels**. We analyze different levels of sparsity in the language matrix to understand how many pairs to mine. Based on these results, we take 60% sparse as a tradeoff point between strong performance and reasonable quantity of mining. 0% indicates no sparsity, or a fully mined language matrix.

3.2.2 Results

We validate the impact of several decisions made during data construction. First, we study the impact of our bridge language strategy compared to English-Centric mining augmented by other random pairs, as well as fully random mining. Second, we investigate the impact of the level of sparsity chosen in our bridge strategy, focusing on a subset of 50 languages.

Bridge Language strategy versus Random and English-Centric Random. We experimentally evaluate the impact of our bridge language mining strategy on the performance of our baseline model in Table 2 (left). We consider two additional baselines, a fully random mining strategy (Random 20%) and a *English-Centric + Random* strategy (Random 20% w/ En). In the Random strategy, mined pairs are randomly chosen, while in the *English-Centric + Random* strategy, we retain all pairs through English and only select the remaining pairs randomly. We show that fully random mining has a substantial negative impact on performance, as a lot of high quality data is aligned through English, so sampling fully randomly eliminates a large portion of the potential training set. Random 20% w/ En is worse as well. Through examination, we find that randomly sampling pairs to mine often selects pairs that do not produce as much data, as the pairs may not include high resource languages. However, the bridge language strategy ensures that high resource languages are mined, and then focuses on mining languages in related families. This produces a large amount of bitext, and at the same time, covers many language directions.

Impact of Sparsity. We control the sparsity of our language matrix using the number of bridge languages. In Figure 2 (right), we show the impact of sparsity on the performance of our baseline model compared to a fully mined language matrix (0% sparse). We observe that increasing the amount of mined data to make the matrix less sparse is helpful, but fully mining the matrix is not substantially better. The main reason is that our mining strategy prioritizes frequently used pairs which are often associated with the largest bitext, while the discarded pairs are often associated with small bitext. For example, fully mining the matrix would mine a pair such as Icelandic to Chinese, but the amount of data produced by mining this pair is quite low. This case is representative of what occurs as the full matrix is mined — as increasingly more data is mined, the additional pairs begin to add less data which in turn leads to diminishing quality improvements.

3.3 Augmenting Bitext Data with Backtranslation

Backtranslation (BT) creates synthetic bitexts from unaligned monolingual data (Schwenk, 2008; Bojar and Tamchyna, 2011; Sennrich et al., 2016a; Edunov et al., 2018; Hoang et al., 2018). The core idea is to translate monolingual sentences in the backward direction, and add the obtained synthetic translations to the training set. More precisely, when training a model to translate from a source language to a target language, backtranslation generates additional data by translating monolingual target sentences into the source language. Using backtranslation thus requires the ability to translate in both directions, which fits well into the setting of multilingual machine translation (Zhang et al., 2020; Siddhant et al., 2020). However, generating these backtranslations is time consuming even for a single direction, which is compounded in the Many-to-Many case. We thus focus on applying backtranslation on specific pairs to supplement mining data where needed.

Selecting Backtranslation Pairs. Our goal is to translate between 100 languages and to provide good translation quality for as many translation directions as possible. To this end, we use BT to improve directions which have initially lower translation quality. We identify these language directions by measuring the quality of our 1.2B parameter multilingual model before applying BT. Since back-translation is computationally intensive, we focus on 100 directions with a BLEU score of between 2 and 10. For 50 of these directions, we do not have any bitext at all as we did not mine all 4,450 possible language pairs.

Training a Multilingual Model with Additional Backtranslations. For the selected pairs, we first generate synthetic translations that are added to the training set without upsampling. Following Caswell et al. (2019), we add a special encoder-side BT token to these translations to indicate to the model that they are synthetic. For each of the 100

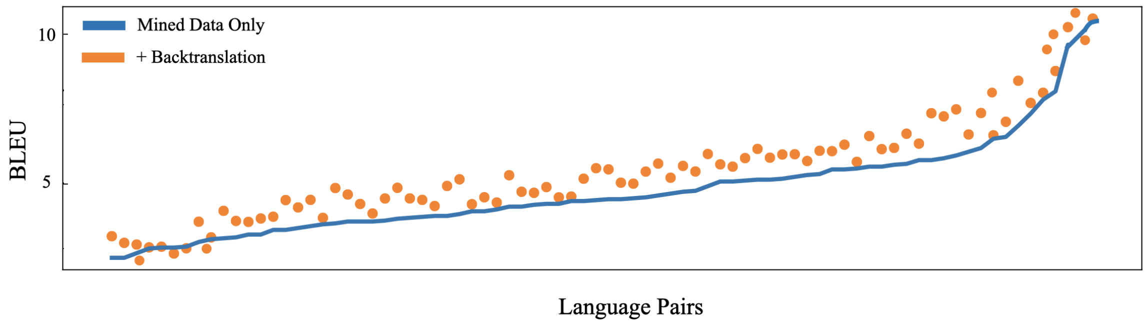


Figure 4: **Improvements from Adding Backtranslated Data.** For each of the 100 language directions we explored by adding backtranslation. The blue line indicates the original model, where directions were selected if they had between 2 and 10 BLEU. The orange scatter indicates the effect of adding backtranslation. Languages are ordered by their original BLEU scores before backtranslation.

	Supervised				**Zero-Shot**	**All**
Data sampling	**Low**	**Mid**	**High**	**Avg**	**Avg**	**Avg**
Uniform	6.1	20.4	38.4	19.0	11.8	15.7
Temperature Rescaling	10.2	23.7	38.0	21.8	13.0	18.1
Sinkhorn Temp. Rescaling	**10.9**	**24.1**	**38.3**	**22.2**	**13.5**	**18.6**

Table 3: **Comparison of Various Sampling Strategies.** We report BLEU on the validation set of our 1.2B base multilingual model trained with different data sampling schemes. Performance is broken down into different resource-setups (low, mid, high) where bitext data exists (supervised) or in the zero-shot setting for pairs without data.

target languages, we randomly sample 50 million unique monolingual sentences from the cleaned CommonCrawl corpus of Wenzek et al. (2019). The synthetic translations are then generated with our 1.2B MMT model. We use a beam search with beam of size 5 and fix all the hyper-parameters, including the length penalty, to the same values for all the directions. We apply the same filtering to the backtranslations as the original mined training data, which substantially reduces the size of the resulting synthetic bitexts.

Impact of Backtranslated Data. Results are shown in Figure 4, where we compare the original Many-to-Many model used to create the backtranslations (blue line) with the improvements after training a multilingual model with the backtranslation added (orange scatter). Backtranslation almost always improves performance for any direction, regardless of the original BLEU score. As the amount of data generated with BT correlates with the length of training time, we decide to focus on applying BT on directions with low performance (BLEU between 2 and 10) to improve our MMT system where it underperforms.

3.4 Balancing Languages in a Many-to-Many Setting

The data distribution produced by large-scale mining is not balanced between languages, so training a model would favor over-represented languages. A standard solution is to rebalance each language independently with Temperature Sampling (Arivazhagan et al., 2019), e.g. replacing the probability p_ℓ of a language by $p_\ell^{\frac{1}{T}}$. In the English-centric case, changing the probability of sampling a language changes the probability of the other languages only through the normalization. However, in the Many-to-Many case, language distributions are more interdependent. For example, some languages are only paired with a subset of other languages or to an overrepresented language. Thus, sampling them will affect the probability of these other languages they are paired with. This strategy thus has no guarantee to produce the target balanced distribution between languages. We describe *Sinkhorn Temperature Sampling*, which extends the temperature sampling strategy to the Many-to-Many setting.

Our goal is to design a sampling technique such that the distribution of languages on the source *and* target sides is equal to a given target distribution. Unfortunately, sequentially sampling the source language and then the target would not work, as some languages are only paired with a subset of languages — making it impossible to sample the target language according to a given distribution. Moreover, the sizes and distributions of bitexts greatly vary from a language to another. Instead, we propose directly sampling a pair of languages from a matrix of pair probabilities such that the marginal distributions of languages corresponds to our target distribution. In practice, this means that each row and column of the matrix should sum to the probability of the corresponding language. More precisely, we estimate a square matrix $\mathbf{P}^*$ such that:

$$\max_{\mathbf{P}} \ \mathrm{tr}\ (\mathbf{PQ}) \qquad \text{s.t.} \quad \mathbf{P}1_L = \mathbf{p}^{\frac{1}{T}},\ \mathbf{P}^\top 1_L = \mathbf{p}^{\frac{1}{T}},$$

where $\mathbf{p}$ is the vector stacking the probabilities of the L languages and $\mathbf{Q}$ is the matrix of pair probabilities. This problem can be solved exactly with the Sinkhorn-Knopp algorithm. The matrix $\mathbf{Q}$ has entries equal to 0 for pairs with no bitext and this algorithm preserves them in the solution $\mathbf{P}^*$, hence adding no probability mass to missing bitexts. We calculate this once before training and set the temperature T to 5. In Table 3, we show the benefits of this strategy over temperature sampling with a constant improvement of 0.5 in BLEU.

4. Many-to-Many Compared to English Centric

In this section, we first present an experiment to better understand the performance improvements of English-Centric systems and to compare them to our Many-to-Many setting.

Experimental Setting. We train our 1.2B model on the full 100 language Many-to-Many dataset and compare it to the same model trained only on data through English. We use the same vocabulary built with SentencePiece on the full dataset in both cases. Each model has a different dataset size and we train for 500K updates. This number of updates corresponds to one pass over the entire Many-to-Many dataset and 3.5 passes on the English-centric data. We tune the dropout rate for each model over the values $\{0.1, 0.2, 0.3\}$.

Setting	**To English**	**From English**	**Non-English**
Bilingual baselines	27.9	**24.5**	8.3
English-Centric	31.0	24.2	5.7
English-Centric with Pivot	—	—	10.4
Many-to-Many	**31.2**	24.1	**15.9**

Table 4: **Comparison of Many-to-Many and English-Centric Systems.** Many-to-Many matches the performance of English-centric on evaluation directions involving English, but is significantly better on non English directions.

Setting	**w/ bitext**	**w/o bitext**
En-Centric	5.4	7.6
En-Centric Piv.	9.8	12.4
M2M	**12.3**	**18.5**

Table 5: **Many-to-Many versus English-Centric on zero-shot directions.** We report performance on language pairs with and without bitext in the Many-to-Many training dataset.

Setting	**→En**	**En→**	**Non-En**
En-Centric	**26.4**	17.8	2.4
En-Centric Piv.	—	—	5.1
M2M	25.7	**18.1**	**9.4**

Table 6: **Many-to-Many versus English-Centric on one pass of data.** We report performance for models after a number of updates equivalent to the size of the English-centric dataset.

4.1 Main Result

In Table 4, we compare the performance of both models on different types of directions, namely, any language to English (To English), English to any language (From English), and all the directions not involving English (Non-English). Performance is aggregated over 150 directions for To English and From English, and over 2500 directions for Non-English. On the pairs including English, both models achieve similar performance, suggesting that a 1.2B model does not underfit even though the additional non-English data represents 98% of the directions and 74% of the data. For the non-English pairs, we consider two translation strategies for the English-Centric model: directly translating as if the model was trained on the pair – by using the corresponding language tokens – or by pivoting through English. Our model outperforms direct translation with the English-Centric model by 10.2 BLEU and when the English-Centric model uses pivoting by 5.5 BLEU. While this result is not surprising, it confirms that a purely English-Centric model has limited potential on non-English pairs, and there is a fundamental need for training on Many-to-Many data.

4.2 Understanding the Source of Improvement

The main impact of adding Many-to-Many data is on the directions that do not include English. In this section, we provide a detailed study of where we observe the largest improvements with the additional data.

Impact on Zero-shot. Many non-English pairs are not covered by our Many-to-Many model, and we can thus study if the improvements we observe originate primarily from

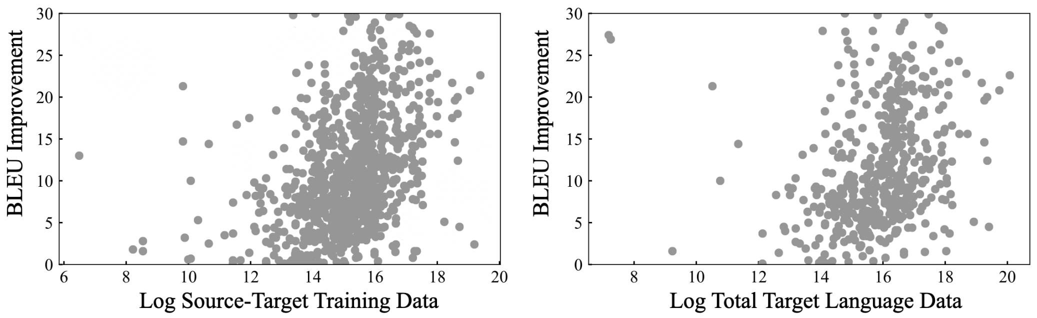


Figure 5: **Improvement of Many-to-Many over English-centric** with respect to the amount of mined training data (left) and the amount of target side language data (right). Improvement from a Many-to-Many model correlates with greater amounts of bilingual training data with Pearson correlation 0.38 (left) and greater amounts of target language data with Pearson correlation 0.32 (right).

directions associated with bitext data or if we observe the same improvement on directions where the Many-to-Many model generates translations in a zero-shot fashion. In Table 5, we show the performance if the evaluation is split between the Non-English pairs *with* and *without* bitext. On directions with bitext, the Many-to-Many model outperforms the English-Centric model by 7 BLEU for direct translation, and by 3.5 BLEU for English-Centric with pivoting. This shows the importance of diverse data. Not surprisingly, this gap is even bigger on pairs without bitext. Many-to-Many performs nearly 11 BLEU better than the English-Centric model for direct translation, and with pivoting the gain remains over 6 BLEU.

Impact of the quantity of training data. A hypothesis to explain the gain between English-Centric and Many-to-Many models is the effect of additional source and target side training data. Even if the Many-to-Many system has never seen a direction at training time, it benefits from additional source and target side data available through other training pairs. As mining non-English language pairs creates more training data compared to English-centric datasets, the Many-to-Many model benefits from a larger training set. In Table 6, we compare both models after seeing the same quantity of data. We train both models for one epoch. The English-Centric model performs better on To English directions, likely because it only has one output language to learn, but the Many-to-Many model outperforms on From English directions and Non-English directions.

Which Pairs Improve the Most? The main factor for improvement is the quantity of data associated with either a pair or a language. Pairs that have a large quantity of mined data, such as Spanish-Portuguese, greatly benefit from our Many-to-Many dataset. We show this effect in the left panel of Figure 5 (left). A second source of improvement is observed on languages for which the Many-to-Many dataset contains a large amount of data across many pairs. This data benefits the decoder-side language model in a way that is comparable with BT. In the right panel of Figure 5, we show the impact of this cumulative monolingual data on the average performance per language. Finally, we also observe a third type of improvements from the similarity in vocabulary and syntax from related languages. A striking example is the quality of translation between English and Belarusian, where the Many-to-Many model

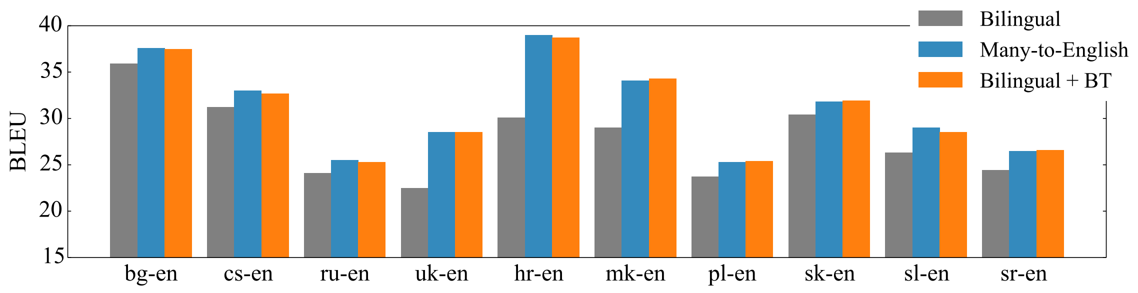


Figure 6: **Performance of many-to-English multilingual translation compared to bilingual baselines trained on mined data and bilingual + backtranslation.** The average performance of many-to-English is 25.1 BLEU compared to 25.2 BLEU for back-translation while the bilingual system achieves 23.1.

achieves 12.7 BLEU on the TED evaluation set, compared to 3.2 BLEU for a bilingual model. The number of bitexts for Belarusian is small, but Belarusian is related to Russian, and the Many-to-Many model transfers its knowledge from Russian to Belarusian.

4.3 Understanding the Performance of English-Centric Systems

In Table 4, we confirm an observation made in Arivazhagan et al. (2019) that an English-Centric model improves the most over bilingual models on the directions into English, while improvement in the other directions (From English) remain more modest. A hypothesis to explain this discrepancy between directions from and to English is that the decoder of an English-Centric model learns a better English language model by leveraging the aggregated English data across all through-English directions.

Result. We test this hypothesis with a controlled experiment where we compare a Many-to-English model with bilingual models using backtranslated English data (§ 3.3). The experiment is based on 11 Slavic languages and we backtranslate the exact same English data as was used to train the Many-to-English model so that both models are trained on the same English data. Figure 6 shows that backtranslation performs comparably to the Many-to-English approach. While this improves our understanding of Many-to-English translation, a multilingual approach nevertheless retains the advantage of combining many directions into a single model which greatly simplifies modeling.

5. Components for Scaling Multilingual Translation Models

Our goal is to build a single model capable of translating $9,900$ language directions covering 100 languages. This creates several challenges for models with insufficient capacity to capture that many languages and scripts adequately. To this end, previous MMT work has considered different types of large capacity models (Arivazhagan et al., 2019; Lepikhin et al., 2020). In this section, we investigate different ways to add capacity to an MMT model: we first investigate dense scaling, where we increase the depth and width of standard Transformer architectures. Then, we identify disadvantages of dense scaling, and propose an alternative to effectively add *language-specific* parameters and exploit the nature of language similarities within the task of multilingual machine translation.

5.1 Dense Scaling

5.1.1 Background: Model Parallel Training

During the training of a neural network, we need to fit its weights, activations, gradients, and optimizer state in memory. This restricts the maximum capacity of a network that we can train on a single accelerated device such as a GPU. In this section, we describe two directions to circumvent this limitation. The first direction focuses on fitting a larger model on single device by reducing the memory required by activations and optimizer states during the training process. The second direction focuses on efficient training of even larger models through model parallelism e.g. splitting a model across multiple devices. In this work, we pursue both techniques to densely scale the capacity of Transformers.

Reducing Memory Consumption on a GPU. To reduce the amount of memory, we consider optimizer state sharding and gradient checkpointing. Optimizer state sharding (Rajbhandari et al., 2019) divides the optimizer state across distributed data parallel workers so that each worker only needs to store a fraction of the optimizer state. We also apply gradient checkpointing, which saves memory by discarding intermediate activations before the forward pass finishes (Chen et al., 2016). During the backward pass, these activations are recomputed again as required. This trades time for memory. In the case of a Transformer based architecture, applying gradient checkpointing at pipeline parallel model partition boundaries reduces the memory used by activations by almost 50%.

Models Sharded across Multiple GPUs. Reducing the memory consumption enables fitting greater model capacity on a single GPU, but the physical limitations of a single device still apply. A solution is to split the model into separate components that are dispatched across different GPUs and trained in parallel. This type of solution scales model capacity with the number of GPUs. There are two broad paradigms to split a model: along the width or along the depth. Tensor parallelism (Shoeybi et al., 2019; Shazeer et al., 2018) splits by width, while pipeline parallelism (Huang et al., 2019; Kim et al., 2020) splits by depth, placing different layers on different GPUs. We use pipeline parallelism, but both methods work equally well with Transformers. We use the implementation from `fairscale`[18].

5.1.2 Training Large Dense Models

We investigate several strategies to increase the capacity of a sequence-to-sequence Transformer model in the context of multilingual machine translation.

How to Scale: Wide or Deep? We consider increasing the capacity of a Transformer by either increasing the number of layers (depth axis) or the dimensions of each layer, including the feedforward (width axis). On the left panel of Figure 7, we analyze which axis to prioritize by comparing models with different sizes, 1B, 2B, and 10B, obtained by growing their depth or width (see Appendix B for model configurations and dimensions). We report their performance in BLEU and their inference speed measured in words per second

18. `https://github.com/facebookresearch/fairscale`

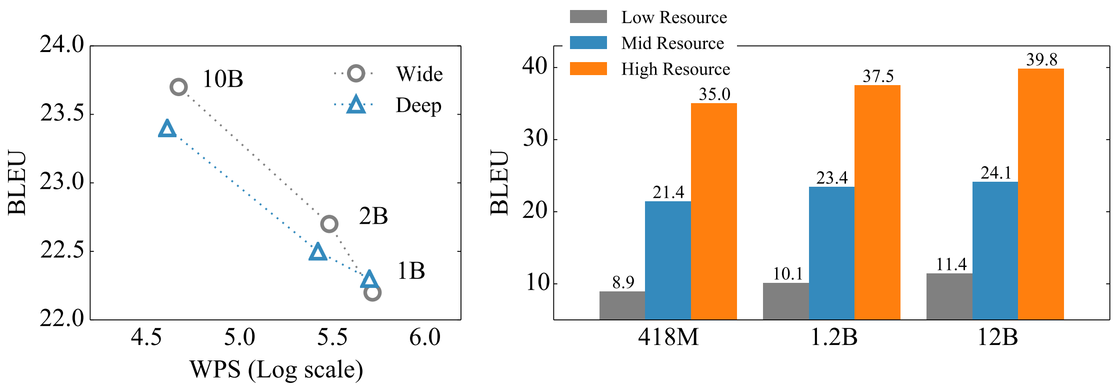


Figure 7: **(left) Comparison between deep versus wide models.** We compare the performance in BLEU for different wide and deep models as a function of their words per second (WPS) at training time, evaluating on a subset of 38 directions. **(right) Performance of wide models for different parameter sizes.** We compare the performance of different wide models on different pairs at low, mid, and high resource levels, evaluating on all supervised evaluation pairs. The white lines indicate comparisons between the different models at the same resource level.

(WPS). We train these models on a dataset that covers 80 languages and evaluate them on 38 different benchmark directions with more than 1k parallel sentences per direction. The main result of this study is that wider models scale better than deeper models in terms of performance and WPS. In the rest of this paper, we thus focus on wider models.

Performance as a function of scale. In the right panel of Figure 7, we compare the performance of wide models as we increase their capacity from 418M to 12B parameters. We train these models on the full set of 100 languages and evaluate them on all supervised evaluation pairs. We report their performance in BLEU for pairs with either low, mid or high resource training data. First, as we increase the number of parameters, we observe that the performance increases, *even on low-resource pairs*. This suggest that even a 12B parameter model could be underfitting with our many-to-many multilingual dataset. However, improvements increase roughly logarithmically in the number of parameters, and we need to scale model size by an order of magnitude to improve by a few BLEU points, e.g., +1.5 BLEU from 1.2B to 12B. As we scale models densely, their runtime and memory usage becomes too prohibitive to justify the gain in performance, and so, we consider alternatives to increase the capacity of our models more efficiently.

5.2 Scaling Model Capacity with Language-Specific Parameters

In this section, we introduce a layer whose parameters are split by language or language group based on similarity in vocabulary. Each translation direction only accesses a subset of these parameters, allowing the model capacity to scale without significantly affecting the training and inference time. The layer is trained with a novel re-routing scheme to improve generalization which we detail below. Compared to previous work (Wang et al., 2018; Bapna

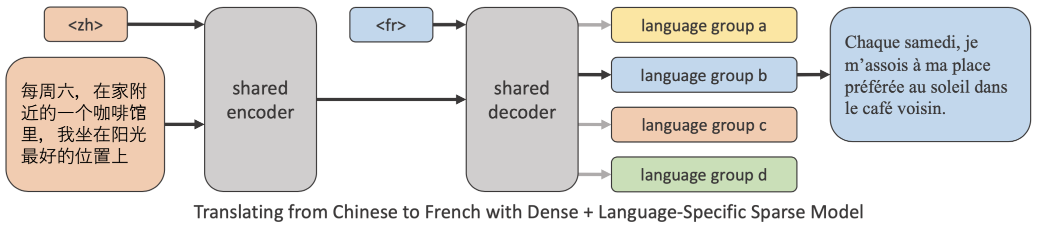

<zh>
每周六，在家附近的一个咖啡馆里，我坐在阳光最好的位置上
shared encoder
<fr>
shared decoder
language group a
language group b
language group c
language group d
Chaque samedi, je m'assois à ma place préférée au soleil dans le café voisin.
Translating from Chinese to French with Dense + Language-Specific Sparse Model

Figure 8: **Language-Specific Parameters** provide specialized capacity to an otherwise fully shared multilingual encoder and decoder.

et al., 2019; Zhang et al., 2020), we focus on allocating entire language-specific layers and using this to scale model size while maintaining training speed.

Parallel Transformer Layer. We follow the sequence-to-sequence Transformer architecture and replace some of its layers by a set of parallel Transformer layers, one for each pre-defined group of languages. More precisely, assuming we have split the languages into K fixed groups, this parallel layer is composed of K parallel Transformer sublayers, one per language group. For each translation, we then select the corresponding sublayer among the K possibilites depending on the language direction. If the parallel layer is in the encoder, we select the sublayer according to the source language, while if it is in the decoder, we select according to the target language. In practice, we only add these layers to either the encoder or decoder, not both. This enables us to split translations along with their sublayers per GPU, leading to faster training and efficient memory usage. Figure 8 shows an example of the resulting *trunk-and-branch* architecture when the parallel layer is in the decoder.

Grouping Languages by Frequency and Similarity. We group languages based on two criteria: the amount of training data and their vocabulary. The motivation for these criteria is that we can learn a specific layer for a language with enough data, and for the rest, overlapping vocabulary is a good proxy for similar languages. First, each language with more than 100M sentences forms its own group and hence has its own sublayer. We have 28 languages that fit this criteria: hu, ru, hi, ro, fr, nl, fi, pt, ar, el, vi, en, ms, tr, he, id, pl, cs, sv, fa, zh, bg, de, es, ko, ja, it, da. Second, we group the remaining languages by vocabulary overlap, leading to 18 additional groups. To create these groupings, we calculate the vocabulary overlap between the training data of different languages and cluster those that have high overlap together. Note that some low resource languages have their own script — such as Kannada — and are not clustered with any similar languages as the script is unique. However, to maintain balance between groups (Wang et al., 2020), we cluster the remaining languages together and roughly balance the amount of training data for each group. In total, we form 46 groups, each with its own sublayer in a language-specific layer.

Random Re-Routing between Sublayers. During training and inference, a sublayer is deterministically selected according to its language direction. This guarantees that our model always uses the same memory and time during inference, regardless of the translation pair. However, during training, this deterministic routing does not share information between similar languages if not associated with the same sublayer. For example, the sublayer associated with Ukrainian does not benefit from the large quantity of Russian training

Model	Params	BLEU
Language Specific Enc	540M	17.1
	920M	17.5
Language Specific Dec	540M	17.3
	920M	17.8

Table 7: **Comparing Language-Specific Encoders and Decoders**. We add parallel language-specific layers to either the encoder or decoder, with different sizes.

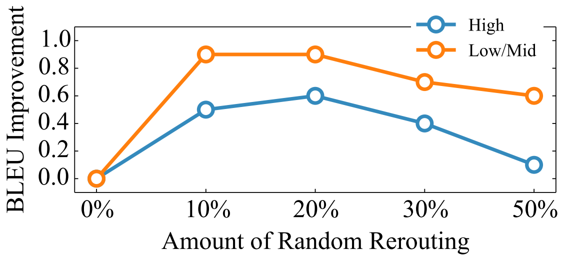


Figure 9: **Impact of Re-routing Rate** on the performance of high resource and low/mid resource languages.

data, since Russian has its own isolated sublayer. We mitigate this shortcoming by *random re-routing* of translations, i.e., randomly picking another sublayer instead of the designated one. This shares information between languages associated with different sublayers, benefiting low resource languages by training on similar high resource languages. The re-routing is completely random, though could be restricted to re-route only to similar languages.

Adding Language-Specific layers to Pre-Trained Transformers. We can integrate a language-specific layer into an already pre-trained Transformer by adding it either at the end of the decoder or at the beginning of the encoder. We can then freeze the parameters of the pre-trained Transformer and learn the language-specific components. These additional language-specific layers train rapidly as the rest of the model already has strong performance. This strategy means it is straightforward to adapt pre-trained networks to a new domain or language by training a small number of dedicated parallel layers, and could easily be extended to various other applications.

5.2.1 Evaluation of the Language-Specific Layer

We experiment with different scenarios by adding a language-specific layer to the encoder or decoder, or to a pre-trained densely scaled model. We demonstrate the importance of random re-routing. Finally, we validate this strategy by comparing it to scaling models densely.

Parallel layer in Encoder or Decoder? The trunk-and-branch architecture for language-specific layers is general and can be used to specialize capacity for any neural architecture. We explore adding language-specific capacity in the encoder or decoder using a smaller setting of 10 high-resource languages. Table 7 shows that language-specific parameters are generally more effective when applied to the decoder network. Recent studies show that encoders are more important for bilingual machine translation (Wu et al., 2019; Kasai et al., 2020), however, these studies are based on systems modeling only a single language direction compared to our setting. In our case, increasing the encoder or the decoder does not impact performance significantly, and we focus on decoder for the rest of this paper.

Random Re-routing. Figure 9 shows the impact of the re-routing strategy on performance as we increase the number of training samples routed to random parallel layers as opposed to their assigned layers. With a re-routing rate of 20%, an improvement of about 0.8 BLEU can be achieved over no re-routing for low and mid resource languages, without affecting

Model	Params	WPS	Supervised Low	Supervised Mid	Supervised High	All Avg
Dense Transformer	1.2B	40K	10.1	23.4	37.5	17.5
Dense Transformer	3B	20K	10.3	23.8	38.0	17.9
Dense Transformer	12B	3.5K	**11.8**	24.2	39.9	18.6
Dense Transformer 1.2B						
with 1 Language-Specific Layer	1.9B	38K	10.7	24.1	38.5	18.1
with 3 Language-Specific Layers	3.5B	34K	10.6	**24.7**	39.5	18.8
with 6 Language-Specific Layers	10B	26K	10.5	24.7	**40.3**	**19.2**

Table 8: **Scaling Model Size with Language-Specific Parameters**. We start with a 1.2B parameter baseline with 24 encoder layers and 24 decoder layers. We add increasingly more decoder layers to language specific layers. For example, in the case of 1 language-specific decoder layer, the decoder has 23 shared layers and 1 language-specific layer. We demonstrate the effect of using 1, 3, and 6 language specific layers. The additional parameters for language-specific layers are split across all language groups. We report WPS at training time holding the batch size fixed on 8 GPUs. The 12B baseline uses model parallel.

the performance of high resource languages. Too much stochasticity leads to performance similar to no random re-routing for high resource languages, but still improves mid to low resource performance compared to no re-routing.

Comparison with Large Dense Models. We compare adding language specific capacity with densely scaling model size in Table 8 on 100 languages. As language-specific layers add many parameters, we compare to baseline models at various sizes for multiple points of comparison. Our conclusion is that language-specific layers improve results compared to baselines of similar parameter size, particularly for mid and high resource languages where there is sufficient data to train the language-specific sublayers. Further, compared to dense scaling, sparse scaling only uses a fraction of the parameters in each forward pass, which maintains fast training speed despite large total model size.

Adding Language-Specific Layers to a Pre-Trained Model. We demonstrate the impact of adding language-specific layers to the decoder of a pre-trained 12B parameter Transformer in Figure 10. We show that adding language-specific layers for five languages improves results on the WMT evaluation datasets. The language-specific layer adds 3.4B parameters and we train it for 20K updates with the rest of the network frozen. The total size of this model is 15.4B parameters. For several directions, we observe gains of more than 1 BLEU, which validates this strategy. On average, we observe gains of 0.9 BLEU.

6. Bringing it all Together

We have explored the creation of a true many-to-many dataset for the multilingual translation of 100 languages, as well as how to effectively scale Many-to-Many models through a mix of dense and sparse scaling. In this section, we summarize our final results, compare to existing

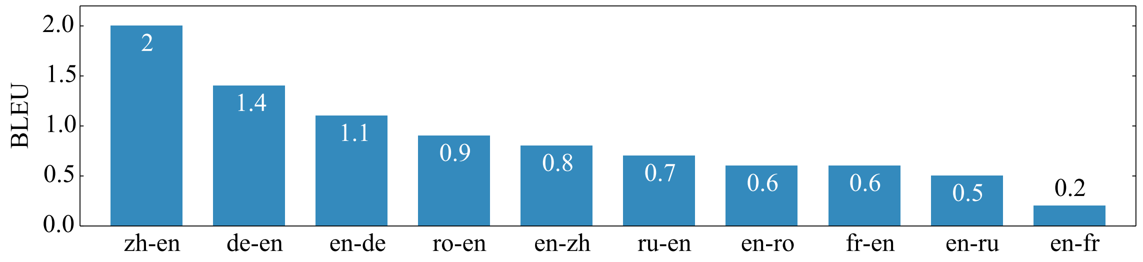


Figure 10: **BLEU Improvement using Dense + Sparse** over Dense alone. We display evaluation on a subset of pairs using the WMT evaluation datasets.

published work — both multilingual benchmarks and competitive directions in WMT — and end with a human evaluation of the overall quality of our translation quality.

6.1 Real-world Settings for Many-to-Many Translation

We highlight that there are several real-world usecases of translation directions not involving English. For example, many countries have official and regional languages that are not English, which would be natural candidates for direct translation. For example, it is intuitive to translate Kazakh directly to Russian in Kazakhstan. In Table 9, we compare English-Centric models to Many-to-Many on a variety of different non-English directions. We see that across the board, our M2M-100 model has drastically better performance and on average improves over 7 BLEU across these directions.

6.2 Comparison on Various Translation Benchmarks

Next, we compare our M2M-100 model to various existing work on different benchmarks. While the training data is not the same, we conduct this comparison to provide a reference point for the overall strength of our model. An important note is that for each of these benchmarks, there are various different tokenizers used which affect BLEU — we follow the tokenization and BLEU calculation of each of these benchmarks, rather than the evaluation methodology of our previous results. Thus, the numbers in this subsection are not comparable to the rest of the paper, as they use the tokenization of each benchmark. Further, this comparison was prepared in advance, so all sentences appearing in these evaluation sets were removed from the training data we used.

Comparison on WMT. First, we compare our Many-to-Many model to submissions to WMT, the premier translation competition. We display results on a variety of different language directions, some of which are standard natural language processing machine translation benchmarks, such as English-French, English-German, and English-Russian. Results are shown in Table 10. [19] Many submissions to the WMT shared task use ensembling, in-domain

19. **En $\leftrightarrow$ De/En $\leftrightarrow$ Ru:** we evaluated publicly available single model checkpoints prior to finetuning from Ng et al. (2019) on WMT2019. **En $\leftrightarrow$ Zh:** we report results from Li et al. (2019) which contains single model BLEU results on WMT2019. **En $\leftrightarrow$ Lt:** we report results from Pinnis et al. (2019) on WMT2019; both directions are the best single model systems which use unconstrained training data. **En $\rightarrow$ Fr:** we report results from Edunov et al. (2018). **Fr $\rightarrow$ En:** we report results from Johnson et al.

	Source	**Target**	**Test Set**	**BLEU**		
				English-Centric	M2M-100	Δ
India	Hindi	Bengali	TED	3.9	8.7	+4.8
	Hindi	Marathi	TED	0.4	8.4	+8.0
	Hindi	Tamil	TED	1.1	7.5	+6.4
South Africa	Afrikaans	Xhosa	Autshumato	0.1	3.6	+3.5
	Afrikaans	Zulu	Autshumato	0.3	3.6	+3.3
	Afrikaans	Sesotho	Autshumato	0.0	2.1	+2.1
	Xhosa	Zulu	Autshumato	0.1	3.6	+3.5
	Sesotho	Zulu	Autshumato	0.1	1.2	+1.1
Chad	Arabic	French	TED	5.3	20.8	+15.5
DR Congo	French	Swahili	Tatoeba	1.8	5.7	+3.9
Kazakhstan	Kazakh	Russian	TED	0.5	4.5	+4.0
Singapore	Chinese	Tamil	TED	0.2	8.0	+7.8
Austria	German	Croatian	TED	9.6	21.3	+11.7
	German	Hungarian	TED	11.3	17.4	+6.1
Belgium	Dutch	French	TED	16.4	25.8	+9.4
	Dutch	German	TED	18.1	26.3	+8.2
Belarus	Belarusian	Russian	TED	10.0	18.5	+8.5
Croatia	Croatian	Serbian	TED	22.4	29.8	+7.4
	Croatian	Hungarian	TED	12.4	17.5	+5.1
	Croatian	Czech	TED	15.2	22.5	+7.3
	Croatian	Slovak	TED	13.8	24.6	+10.8
Cyprus	Greek	Turkish	TED	4.8	12.6	+7.8
Czechia	Czech	Slovak	TED	9.5	28.1	+18.6
Finland	Finnish	Swedish	TED	7.9	19.2	+11.3
Italy	Italian	French	TED	18.9	28.8	+9.9
	Italian	German	TED	18.4	25.6	+7.2
Moldova	Romanian	Russian	TED	8.0	19.0	+11.0
	Romanian	Ukrainian	TED	8.7	17.3	+8.6
Montenegro	Albanian	Croatian	TED	3.0	20.7	+17.7
	Albanian	Serbian	TED	7.8	20.6	+12.8
Romania	Romanian	German	TED	15.0	24.7	+9.7
	Romanian	Hungarian	TED	11.0	16.3	+4.3
	Romanian	Turkish	TED	5.1	12.0	+6.9
	Romanian	Armenian	TED	0.4	8.2	+7.8
Russia	Bashkir	Russian	Tatoeba	0.1	4.3	+4.2
	Ukrainian	Russian	TED	18.0	23.7	+5.7
			Average	8.0	15.6	+**7.6**

Table 9: **Performance translating between official and official regional languages** of several nations, focusing on non-English directions.

Direction	Test Set	BLEU Published	BLEU M2M-100	BLEU Δ
Without Improvement				
English-Chinese (Li et al., 2019)	WMT'19	38.2	33.2	-5.0
English-Finnish (Talman et al., 2019)	WMT'17	28.6	28.2	-0.4
English-Estonian (Pinnis et al., 2018)	WMT'18	24.4	24.1	-0.3
Chinese-English (Li et al., 2019)	WMT'19	29.1	29.0	-0.1
With Improvement				
English-French (Edunov et al., 2018)	WMT'14	43.8	43.8	0
English-Latvian (Pinnis et al., 2017)	WMT'17	20.0	20.5	+0.5
German-English (Ng et al., 2019)	WMT'19	39.2	40.1	+0.9
Lithuanian-English (Pinnis et al., 2019)	WMT'19	31.7	32.9	+1.2
English-Russian (Ng et al., 2019)	WMT'19	31.9	33.3	+1.4
English-Lithuanian (Pinnis et al., 2019)	WMT'19	19.1	20.7	+1.6
Finnish-English (Talman et al., 2019)	WMT'17	32.7	34.3	+1.6
Estonian-English (Pinnis et al., 2018)	WMT'18	30.9	33.4	+2.5
Latvian-English (Pinnis et al., 2017)	WMT'17	21.9	24.5	+2.6
Russian-English (Ng et al., 2019)	WMT'19	37.2	40.5	+3.3
French-English (Edunov et al., 2018)	WMT'14	36.8	40.4	+3.6
English-German (Ng et al., 2019)	WMT'19	38.1	43.2	+5.1
English-Turkish (Sennrich et al., 2017)	WMT'17	16.2	23.7	+7.5
Turkish-English (Sennrich et al., 2017)	WMT'17	20.6	28.2	+7.6
	Average	30.0	31.9	**+1.9**

Table 10: **Comparison of Many-to-Many and public results on WMT datasets.** We compare M2M-100 to published work (best single models) on WMT. To identify previous work, we examine the WMT Shared Task proceedings for the top performing models and check reported results on `http://matrix.statmt.org/`. For these comparisons, we report detokenized BLEU with sacrebleu (Post, 2018) on the test set.

finetuning, or reranking methods, which are standard techniques to improve quality. As these could be added to our system at inference time as well, we focus instead on comparing single model results. To identify comparisons, we examine the WMT Shared Task proceedings as well as the submissions at `http://matrix.statmt.org/`.

As seen in Table 10, our M2M-100 system can achieve very competitive performance compared to bilingual models tuned especially for individual WMT translation directions. This shows that our model maintains strong translation quality on individual directions.

Next, we compare our models to other multilingual translation work. Table 11 displays several previously published results on different sets of benchmarks. Note that for each comparison, we follow the published setting in tokenization, evaluation, and whether or not the BLEU is tuned on the validation set to maximize comparability.

Bilingual Models. We first compare to mBART (Liu et al., 2020), which creates bilingual models based on finetuning a pretrained model on individual language directions. After pretraining as a denoising autoencoder, publicly available bitext data is used to create various

(2017) on WMT2014. **En ↔ Lv:** we report results from Pinnis et al. (2017) on WMT2017. **En ↔ Tr:** we report results from Sennrich et al. (2017) on WMT17. **En ↔ Et:** we report results from Pinnis et al. (2018) on WMT18. **En ↔ Fi:** we report results from Talman et al. (2019) on WMT17.

Benchmark	Model	BLEU
mBART	Previous Work (Liu et al., 2020)	23.9
	M2M-100	**24.6**
CCMatrix	Previous Work (Schwenk et al., 2019)	16.3
	M2M-100	**18.7**
OPUS100	Previous Work (Zhang et al., 2020)	14.1
	M2M-100	**18.4**

Table 11: **Comparison on various evaluation settings from previous work**. We display the best performing model from the published work and report average BLEU on the test set. For these comparisons, we use the tokenization and BLEU evaluation script used by each work for comparability. Liu et al. (2020) report Low/Mid resource directions into and out of English and High resource directions into English, we average across all. Schwenk et al. (2019) report the full matrix on 28 languages, we average across all. Zhang et al. (2020) report results on non-English directions, we average across all.

different bilingual models, one for each evaluation direction. Liu et al. (2020) tune the test set BLEU on the validation set. Following their setting, we tune the generation beam size between {5,10}, and length penalty between {0.5, 1.0, 1.5}, and the number of checkpoints to average between {1, 5, 10}. Our model provides +0.7 BLEU improvement.

We then compare to the bilingual baselines provided in CCMatrix (Schwenk et al., 2019), which trained individual models for each direction. As these models generate with no tuning, we generate on all pairs with beam size 5 and length penalty 1, using only the best checkpoint. Our one Many-to-Many multilingual model achieves a 2 BLEU point gain on average compared to training hundreds of individual models.

Multilingual Models. We next compare the performance of our multilingual system to other published multilingual systems. We compare to the English-Centric multilingual model from Zhang et al. (2020) on the OPUS100 corpus. Their model is trained with noisily aligned through-English data from OPUS (Tiedemann, 2012; Zhang et al., 2020), with online backtranslation to improve the performance of non-English pairs. Note that Zhang et al. (2020) train on 100 directions, but we only overlap a subset of directions. However, we fully cover their full set of non-English evaluation pairs. Finally, the OPUS100 non-English directions come only with a test set, so we generate with beam size 5, length penalty 1, and use the best checkpoint. As shown in Table 11, we improve by more than 4 BLEU.

6.3 Human Evaluation

We end with a human evaluation study to understand the quality of our model translations. We focus on 20 different directions, none of them involving English. We include languages commonly spoken in the same region, such as Japanese-Chinese, Hindi-Tamil, and Russian-Ukrainian, as well as directions that cross language families, such as Chinese-French, French-Arabic, and Russian-Spanish. We also include several very low resource directions, such as French-Wolof, Hindi-Marathi, and Japanese-Mongolian. All of our evaluators are native speakers in one of the languages and fluent in the other.

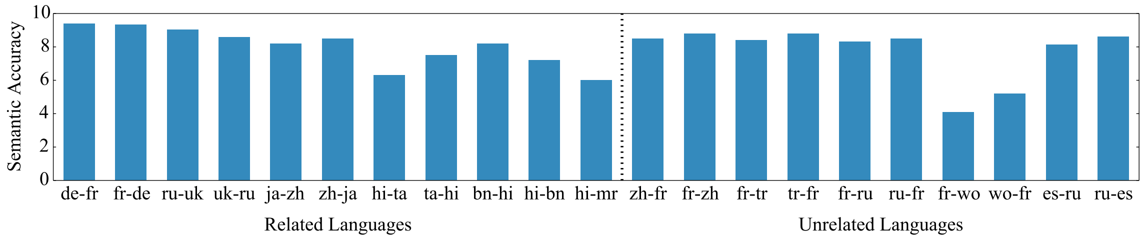


Figure 11: **Human Evaluation of Translation Accuracy of M2M-100 on Non-English Directions.** Evaluators are asked to score the semantic accuracy of translations on a scale of 1 to 10.

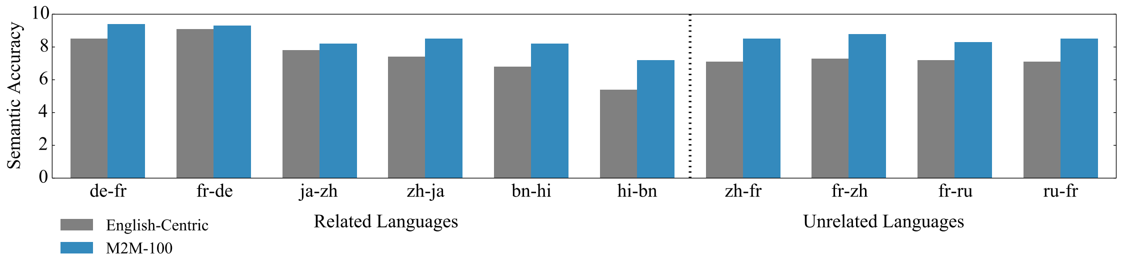


Figure 12: **Human Evaluation of Translation Accuracy of M2M-100 compared to English-Centric on 10 Non-English Directions.** Evaluators are asked to score the semantic accuracy of translations on a scale of 1 to 10.

Each evaluator rates 50 different translations for semantic accuracy on a scale of 1 to 10. Results are shown in Figure 11. On semantic accuracy, most of our evaluations score between 8.5 and 9.5 (with 10 being the best possible score). For lower resource directions, the scores remain reasonable. Hindi to Tamil and Wolof to French score around 7-8. The most challenging direction based on human evaluation is French into Wolof (fr-wo), likely because there is not sufficient target-side Wolof data.

Next, we compare our model with an English-Centric model on 10 directions in Figure 12. Each evaluator is asked to rate 100 sentences, 50 from each model, in a blind test. Across the board, we find that our Many-to-Many system scores better in translation accuracy - both for related and unrelated languages.

6.4 Discussion

Curating High-Quality Training Data. Creating high quality datasets to train translation models has been a long-standing area of research. For example, previous work has explored how to best filter noisy datasets (Koehn et al., 2018, 2019). Our use of large-scale mined training data presents large quantities of data to train multilingual models on, but brings challenges as well. For example, our mining methods mine both simplified and traditional Chinese text, tokenized and untokenized text, and many examples with code switching. We apply several data filtering methods, but the cleanliness and quality of alignment is critical for training high-quality translation systems. Further, multilingual translation can be affected by domain mismatch, as people in different parts of the world discuss different topics (Shen et al., 2019), which presents additional challenges for curating good training sets. Thus, we see the continued improvement of data quality as an important direction for multilingual translation systems, which require a lot of data to train well.

Improvements on Very Low-Resource Languages. Strong performance for low-resource languages remains a critical area for future improvement (Gu et al., 2018; Sennrich and Zhang, 2019). For many languages, our system still requires substantial improvements. Examples include African languages such as Xhosa and Zulu, European languages such as Catalan and Basque, and Southeast Asian languages such as Iloko and Cebuano. For many of these, even monolingual resources on the internet are limited, which strongly affects the quantity and quality of mined data. Using curated data, possibly supplemented by mining, may provide a starting point for future improvement. For example, several resources for African languages exist, including JW300 (Agić and Vulić, 2019) used in the MASAKHANE machine translation effort (∀ et al., 2020) and datasets for Nigerian Pidgin (Ahia and Ogueji, 2020), Wolof (Alla et al., 2020), Fon (Emezue and Dossou, 2020), Igbo (Ezeani et al., 2020), Amharic, Tigrigna, Afan-Oromo, Wolaytta, and Ge'ez (Abate et al., 2018). Other lines of work present resources for low-resource Asian languages, such as the ALT project (Riza et al., 2016; Ding et al., 2016), Mongolian, Uyghur, and Tibetian (Anonymous, 2020), or strategies for improvement on specific directions (Chen et al., 2019). Further research is required to bring together small datasets of higher quality translations, mined data, and monolingual resources to create improved translation systems for very low resource languages.

7. Conclusion

We introduced M2M-100, a new Many-to-Many multilingual translation model that can translate between the 9,900 directions of 100 languages. The underlying dataset was mined from CommonCrawl using a novel strategy which exploits language groupings to avoid mining every possible direction while maintaining good accuracy. Such a large dataset requires models with increased capacity and to this end we explored densely scaling the number of parameters as well as sparsely, through introducing language-specific parameters trained with a novel random re-routing scheme.

Results show that M2M-100 outperforms English-Centric multilingual models trained on data where either the source or target language is English. The system improves over 10 BLEU on average compared to an English-Centric baseline when translating directly between non-English directions. M2M-100 is competitive to bilingual models from WMT and improves over existing publicly available multilingual translation systems. Human judges indicate that our model translates fluently with high semantic accuracy.

Acknowledgments

We thank Yuqing Tang and Peng-Jen Chen for their work on the multilingual translation infrastructure in fairseq. We thank Xian Li, Chau Tran, Yuqing Tang, Peng-Jen Chen, and Marc'Aurelio Ranzato for insightful conversations. We thank our volunteer human evaluators for closely examining the translation quality of our models through various directions.

References

Solomon Teferra Abate, Michael Melese, Martha Yifiru Tachbelie, Million Meshesha, Solomon Atinafu, Wondwossen Mulugeta, Yaregal Assabie, Hafte Abera, Binyam Ephrem Seyoum, Tewodros Abebe, et al. Parallel corpora for bi-directional statistical machine translation for seven ethiopian language pairs. In Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing, pages 83–90, 2018.

Željko Agić and Ivan Vulić. JW300: A wide-coverage parallel corpus for low-resource languages. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3204–3210, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1310. URL `https://www.aclweb.org/anthology/P19-1310`.

Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. arXiv preprint arXiv:1903.00089, 2019.

Orevaoghene Ahia and Kelechi Ogueji. Towards supervised and unsupervised neural machine translation baselines for nigerian pidgin. arXiv preprint arXiv:2003.12660, 2020.

Lo Alla, Dione Cheikh Bamba, Nguer Elhadji Mamadou, Ba Sileye O Ba, and Lo Moussa. Using lstm to translate french to senegalese local languages: Wolof as a case study. arXiv preprint arXiv:2004.13840, 2020.

Anonymous. Syntactic relevance xlnet word embedding generation in low-resource machine translation. OpenReview, 2020.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. Massively multilingual neural machine translation in the wild: Findings and challenges. arXiv preprint arXiv:1907.05019, 2019.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. arXiv preprint arXiv:1802.06509, 2018.

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. In `https://arxiv.org/abs/1812.10464`, 2018a.

Mikel Artetxe and Holger Schwenk. Margin-based parallel corpus mining with multilingual sentence embeddings. arXiv preprint arXiv:1811.01136, 2018b.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv, abs/1607.06450, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Simple, scalable adaptation for neural machine translation. arXiv preprint arXiv:1909.08478, 2019.

Loïc Barrault, Ondřej Bojar, Marta R Costa-Jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. Findings of the 2019 conference on machine translation (wmt19). In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), pages 1–61, 2019.

Ondrej Bojar and Ales Tamchyna. Improving translation model by monolingual data. In Workshop on Statistical Machine Translation (WMT), 2011.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (WMT18). In Proceedings of the Third Conference on Machine Translation: Shared Task Papers, pages 272–303, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6401. URL `https://www.aclweb.org/anthology/W18-6401`.

Isaac Caswell, Ciprian Chelba, and David Grangier. Tagged back-translation. arXiv preprint arXiv:1906.06442, 2019.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Niehues Jan, Stüker Sebastian, Sudoh Katsuitho, Yoshino Koichiro, and Federmann Christian. Overview of the iwslt 2017 evaluation campaign. In International Workshop on Spoken Language Translation, pages 2–14, 2017.

Vishrav Chaudhary, Yuqing Tang, Francisco Guzmán, Holger Schwenk, and Philipp Koehn. Low-resource corpus filtering using multilingual sentence embeddings. WMT 2019, page 261, 2019.

Peng-Jen Chen, Jiajun Shen, Matt Le, Vishrav Chaudhary, Ahmed El-Kishky, Guillaume Wenzek, Myle Ott, and Marc'Aurelio Ranzato. Facebook ai's wat19 myanmar-english translation task submission. arXiv preprint arXiv:1910.06848, 2019.

Pinzhen Chen, Nikolay Bogoychev, Kenneth Heafield, and Faheem Kirefu. Parallel sentence mining by constrained decoding. 2020.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. arXiv, abs/1604.06174, 2016.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116, 2019.

Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. Similar southeast asian languages: Corpus-based case study on thai-laotian and malay-indonesian. In Proceedings of the 3rd Workshop on Asian Translation (WAT2016), pages 149–156, 2016.

Chenchen Ding, Hnin Thu Zar Aye, Win Pa Pa, Khin Thandar Nwet, Khin Mar Soe, Masao Utiyama, and Eiichiro Sumita. Towards Burmese (Myanmar) morphological analysis:

Syllable-based tokenization and part-of-speech tagging. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 19(1):5, 2019.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In Proc. of EMNLP, 2018.

Ahmed El-Kishky, Vishrav Chaudhary, Francisco Guzman, and Philipp Koehn. CCAligned: A massive collection of cross-lingual web-document pairs. In Proc. of EMNLP, 2020.

Chris Chinenye Emezue and Femi Pancrace Bonaventure Dossou. Ffr v1. 1: Fon-french neural machine translation. In Proceedings of the The Fourth Widening Natural Language Processing Workshop, pages 83–87, 2020.

Ignatius Ezeani, Paul Rayson, Ikechukwu Onyenwe, Chinedu Uchechukwu, and Mark Hepple. Igbo-english machine translation: An evaluation benchmark. arXiv preprint arXiv:2004.00648, 2020.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. arXiv preprint arXiv:1909.11556, 2019.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. arXiv preprint arXiv:1601.01073, 2016.

∀, Wilhelmina Nekoto, Vukosi Marivate, Tshinondiwa Matsila, Timi Fasubaa, Tajudeen Kolawole, Taiwo Fagbohungbe, Solomon Oluwole Akinola, Shamsuddee Hassan Muhammad, Salomon Kabongo, Salomey Osei, et al. Participatory research for low-resourced machine translation: A case study in african languages. arXiv preprint arXiv:2010.02353, 2020.

Ekaterina Garmash and Christof Monz. Ensemble learning for multi-source neural machine translation. In Proc. COLING, 2016.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In Proc. of ICML, 2017.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. Universal neural machine translation for extremely low resource languages. arXiv preprint arXiv:1802.05368, 2018.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. Improved zero-shot neural machine translation via ignoring spurious correlations. arXiv preprint arXiv:1906.01181, 2019.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. 2019.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. Toward multilingual neural machine translation with universal encoder and decoder. arXiv preprint arXiv:1611.04798, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In Proc. of CVPR, 2015.

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

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In Advances in neural information processing systems, pages 103–112, 2019.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. IEEE Transactions on Big Data, 2019.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. Transactions of the Association for Computational Linguistics, 5:339–351, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. arXiv, abs/2001.08361, 2020.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. arXiv, 2020.

Chiheon Kim, Heungsub Lee, Myungryong Jeong, Woonhyuk Baek, Boogeon Yoon, Ildoo Kim, Sungbin Lim, and Sungwoong Kim. torchgpipe: On-the-fly pipeline parallelism for training giant models. arXiv preprint arXiv:2004.09910, 2020.

DP Kingma and LJ Ba. Adam: A method for stochastic optimization. arXiv, 2015.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In MT Summit, 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, pages 177–180. Association for Computational Linguistics, 2007.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L Forcada. Findings of the wmt 2018 shared task on parallel corpus filtering. In Proceedings of the Third Conference on Machine Translation: Shared Task Papers, pages 726–739, 2018.

Philipp Koehn, Francisco Guzmán, Vishrav Chaudhary, and Juan Pino. Findings of the WMT 2019 shared task on parallel corpus filtering for low-resource conditions. In Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2), 2019.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226, 2018.

Anoop Kunchukuttan. The IndicNLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`, 2020.

Ivana Kvapilíková, Mikel Artetxe, Gorka Labaka amd Eneko Agirre, and Ondřej Bojar. Unsupervised multilingual sentence embeddings for parallel corpus mining. 2020.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv, 2020.

Bei Li, Yinqiao Li, Chen Xu, Ye Lin, Jiqiang Liu, Hui Liu, Ziyang Wang, Yuhao Zhang, Nuo Xu, Zeyang Wang, et al. The niutrans machine translation systems for wmt19. In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), pages 257–266, 2019.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. arXiv preprint arXiv:2001.08210, 2020.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook FAIR's WMT19 news translation task submission. In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1). Association for Computational Linguistics, 2019. URL `https://www.aclweb.org/anthology/W19-5333`.

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. arXiv preprint arXiv:1910.05895, 2019.

Bojar Ondrej, Rajen Chatterjee, Federmann Christian, Graham Yvette, Haddow Barry, Huck Matthias, Koehn Philipp, Liu Qun, Logacheva Varvara, Monz Christof, et al. Findings of the 2017 conference on machine translation (wmt17). In Second Conference onMachine Translation, pages 169–214. The Association for Computational Linguistics, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing 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, 2002.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. In International Conference on Learning Representations (ICLR) Workshop, 2017.

Wannaphong Phatthiyaphaibun, Korakot Chaovavanich, Charin Polpanumas, Arthit Suriyawongkul, Lalita Lowphansirikul, and Pattarawat Chormai. PyThaiNLP: Thai Natural Language Processing in Python, June 2016. URL `http://doi.org/10.5281/zenodo.3519354`.

Mārcis Pinnis, Rihards Krišlauks, Toms Miks, Daiga Deksne, and Valters Šics. Tilde's machine translation systems for WMT 2017. Association for Computational Linguistics, 2017. URL `https://www.aclweb.org/anthology/W17-4737`.

Mārcis Pinnis, Matīss Rikters, and Rihards Krišlauks. Tilde's machine translation systems for WMT 2018. Association for Computational Linguistics, 2018. URL `https://www.aclweb.org/anthology/W18-6423`.

Marcis Pinnis, Rihards Krišlauks, and Matīss Rikters. Tilde's machine translation systems for WMT 2019. Association for Computational Linguistics, 2019. URL `https://www.aclweb.org/anthology/W19-5335`.

Matt Post. A call for clarity in reporting BLEU scores. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages 186–191, 2018.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. ArXiv, 2019.

Philip Resnik. Mining the Web for Bilingual Text. In ACL, 1999. URL `http://www.aclweb.org/anthology/P99-1068`.

Hammam Riza, Michael Purwoadi, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Sethserey Sam, et al. Introduction of the asian language treebank. In 2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA), pages 1–6. IEEE, 2016.

Holger Schwenk. Investigations on large-scale lightly-supervised training for statistical machine translation. In IWSLT, pages 182–189, 2008.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, Armand Joulin, and Angela Fan. CCMatrix: Mining billions of high-quality parallel sentences on the web. arXiv preprint arXiv:1911.04944, 2019.

Rico Sennrich and Biao Zhang. Revisiting low-resource neural machine translation: A case study. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 211–221, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1021. URL `https://www.aclweb.org/anthology/P19-1021`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. Conference of the Association for Computational Linguistics (ACL), 2016a.

Rico Sennrich, Barry Haddow, and Alexandra 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, 2016b.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The University of Edinburgh's neural MT systems for WMT17. Association for Computational Linguistics, 2017. URL `https://www.aclweb.org/anthology/W17-4739`.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. Mesh-tensorflow: Deep learning for supercomputers. In Advances in Neural Information Processing Systems, pages 10414–10423, 2018.

Jiajun Shen, Peng-Jen Chen, Matt Le, Junxian He, Jiatao Gu, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. The source-target domain mismatch problem in machine translation. arXiv preprint arXiv:1909.13151, 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint arXiv:1909.08053, 2019.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. Leveraging monolingual data with self-supervision for multilingual neural machine translation. arXiv preprint arXiv:2005.04816, 2020.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567, 2015.

Aarne Talman, Umut Sulubacak, Raúl Vázquez, Yves Scherrer, Sami Virpioja, Alessandro Raganato, Arvi Hurskainen, and Jörg Tiedemann. The University of Helsinki submissions to the WMT19 news translation task. In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1). Association for Computational Linguistics, 2019. URL `https://www.aclweb.org/anthology/W19-5347`.

Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), pages 2214–2218, 2012.

Masao Utiyama and Hitoshi Isahara. Reliable Measures for Aligning Japanese-English News Articles and Sentences. In ACL, 2003. URL `http://www.aclweb.org/anthology/P03-1010`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. Balancing training for multilingual neural machine translation. In Proc. of ACL, 2020.

Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong. Three strategies to improve one-to-many multilingual translation. In Proceedings of the 2018 Conference on

Empirical Methods in Natural Language Processing, pages 2955–2960, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1326. URL `https://www.aclweb.org/anthology/D18-1326`.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. arXiv preprint arXiv:1911.00359, 2019.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In Proc. of ICLR, 2019.

Qi Ye, Sachan Devendra, Felix Matthieu, Padmanabhan Sarguna, and Neubig Graham. When and why are pre-trained word embeddings useful for neural machine translation. In HLT-NAACL, 2018.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. Improving massively multilingual neural machine translation and zero-shot translation. arXiv preprint arXiv:2004.11867, 2020.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. The United Nations parallel corpus v1.0. In LREC, May 2016.

Appendix A. Additional Information about Data

Figure 13 displays the dictionary coverage for each of our 100 languages.

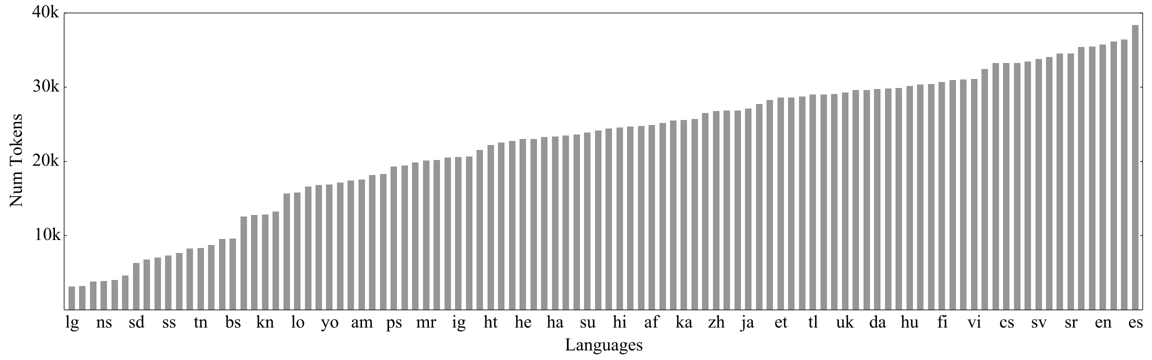


Figure 13: **Dictionary Coverage** per Language

Appendix B. Model Architectures

Table 14 shows the various model configurations considered in our experiments when scaling dense models.

Size	Embed	FFN	Layers
1B Wide	1024	16K	14
1B Deep	1024	4K	38
2B Wide	2048	16K	11
2B Deep	1024	8K	48
10B Wide	4096	16K	24
10B Deep	3072	12K	36

Figure 14: **Architecture of Wide and Deep Models.**

Appendix C. Exploiting Multilinguality at Inference Time with Multi-source Self-Ensembles

Throughout the paper, we explored how to improve the performance of single models, scaling the amount of data as well as the model size, but there remain numerous directions for future investigation of multilinguality. One direction is understanding how to exploit the nature of multilingual translation at inference time as well.

A known, effective strategy to improve accuracy is to ensemble multiple models at inference time. However, this requires training multiple models which substantially increases the training compute requirements. Instead, we suggest exploring self-ensembles, created by applying the multilingual model to the same source sentence in different languages. For

Model	**BLEU**
Multilingual	17.3
Multi-Model Ensemble	17.5
Pivoting with Multilingual	17.0
Multi-source Self-Ensemble	17.5

Table 12: **Results on zero-shot language pairs for Multi-Source Self-Ensemble** compared to various baselines. We report the average test BLEU score on 100 randomly sampled pairs.

example, if we wish to translate Galician to English, then instead of directly translating between the two, we ensemble the translation of Spanish to English with the translation of Galician to English, using the same multilingual model for both directions, and by averaging the predicted token log-probabilities, as for standard multi-model ensembles. The additional source is obtained by translating the input to another *intermediary* language. After this, we ensemble the translation of both sources to the target. This uses the same multilingual model for all steps.

We evaluate both pivoting and self-ensembling on zero-shot directions as these can benefit from better accuracy. We report results on 100 randomly sampled zero-shot translation directions which have at least 1000 examples in the validation and test set. Next, for each translation direction, we choose the intermediary language that resulted in the highest BLEU on the validation set; the same is done to choose the intermediary language for pivoting. We also tune a weight to balance the two language directions (Garmash and Monz, 2016). Table 12 shows that multi-source self-ensembling improves the single model result by 0.2 BLEU on average. It also performs as well as standard multi-model ensembling but requires training only a single model. This is particularly relevant for large models trained on vast quantities of data, which require a lot of compute to be able to perform standard ensembling.