eyond English-Centric Multilingual Machine Translation

ngela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, hmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael uli[†], rmand Joulin[†‡]
Facebook I, LORI

bstract

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 `h r` .

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 (rivazhagan 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 uthor. Email: `angelafan@fb.com`

†. Equal contribution. Order determined with coin flip.

‡. Holger Schwenk, Onur Celebi, Vishrav Chaudhary, hmed El-Kishky, ngela Fan, Edouard Grave, Zhiyi Ma, and Guillaume Wenzek worked on large-scale data mining, including improvements to fasttext, L SER, CC ligned, and CCMatrix. Siddharth Goyal worked on backtranslation. Michael uli, Mandeep Baines, Shruti Bhosale, Tom Birch, Sergey Edunov, ngela Fan, Naman Goyal, Siddharth Goyal, and Vitaliy Liptchinsky worked on model scaling and scaling infrastructure. Michael uli, Shruti Bhosale, Sergey Edunov, ngela Fan, Edouard Grave, rmand Joulin, Zhiyi Ma, and Holger Schwenk worked on model and experimental design. Michael uli, hmed El-Kishky, ngela Fan, Edouard Grave, rmand Joulin, Zhiyi Ma, and Holger Schwenk wrote the paper.

must be split between many languages (rivazhagan et al., 2019). This has been alleviated by increasing model capacity (haroni 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- entric* 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 (rtetxe 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; rora 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.

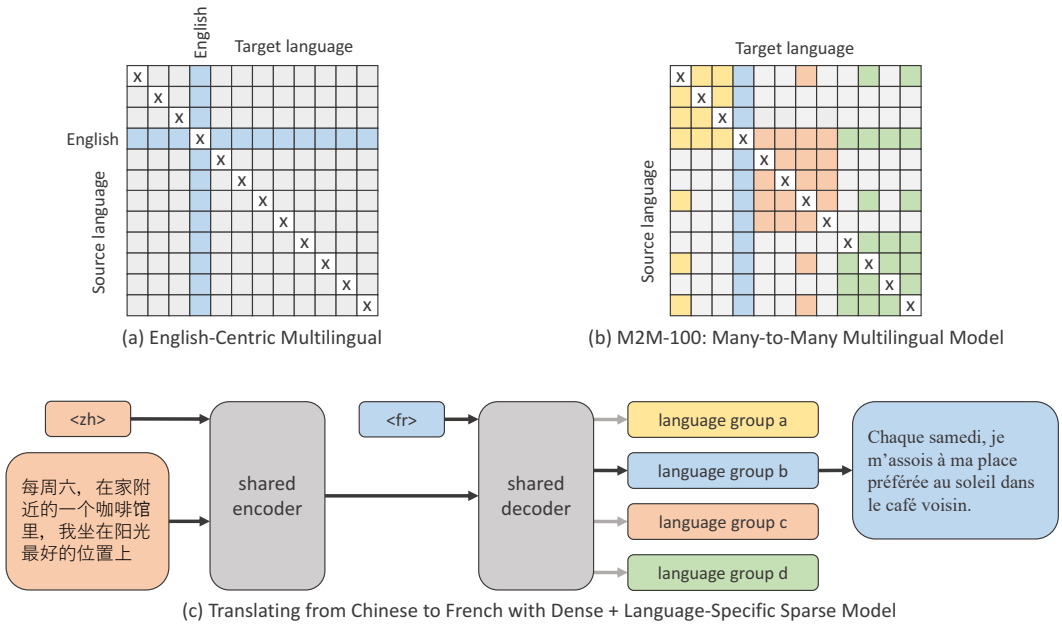

Figure 1: **Summary of our Many-to-Many dataset and multilingual model**. English-Centric data (top left) only contains training data to and from English, where as our Many-to-Many multilingual setting (top right) contains data directly through various different directions. Our proposed model, M2M-100, 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 $(w_1, \quad, w_S)$ and $(v_1, \quad, v_T)$. Most machine translation systems are trained by maximizing the probability of the target sequence, given the source sentence and the target language ℓ_t:

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

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. nother limitation of word-based systems are languages that are not naturally split into words, like Thai. n 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., $p_\ell = \frac{\ell}{i \ i}$. 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_ℓ^{-} 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 ppendix .

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 $W = (w_1, \quad , w_S)$ and the source language ℓ_s, and produces a sequence of embeddings $H = (h_1, \quad , h_S)$, which are then fed to the decoder with the target language ℓ_t to produce the sequence of target tokens $V = (v_1, \quad , v_T)$ sequentially, i.e.,

$$H = \texttt{encoder}(W, \ \ell_s), \tag{1}$$

$$\forall i \in [1, \quad , T], \ v_{i \ 1} = \texttt{decoder}(H, \ \ell_t, \ v_1, \quad , \ v_i) \tag{2}$$

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):

$$Z = \texttt{norm}\left(X + \texttt{self-attention}(X)\right), \tag{3}$$

$$Y = \texttt{norm}\left(Z + \texttt{feed-forward}(Z)\right) \tag{4}$$

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 dam 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 (rtetxe 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	frikaans	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- ryan	Eastern-Nagari
is	Icelandic	Germanic	Latin	gu	Gujarati	Indo- ryan	Gujarati
lb	Luxembourgish	Germanic	Latin	hi	**Hindi**	Indo- ryan	Devanagari
no	Norwegian	Germanic	Latin	kn	Kannada	Tamil	Kannada
sv	**Swedish**	Germanic	Latin	mr	Marathi	Indo- ryan	Devanagari
fy	Western Frisian	Germanic	Latin	ne	Nepali	Indo- ryan	Devanagari
yi	Yiddish	Germanic	Hebrew	or	Oriya	Indo- ryan	Odia
ast	sturian	Romance	Latin	pa	Panjabi	Indo- ryan	Gurmukhi
ca	Catalan	Romance	Latin	sd	Sindhi	Indo- ryan	Persian Devanagari
fr	**French**	Romance	Latin	si	Sinhala	Indo- ryan	Sinhala
gl	Galician	Romance	Latin	ur	Urdu	Indo- ryan	rabic
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	**rabic**	rabic	rabic
uk	Ukrainian	Slavic	Cyrillic	he	**Hebrew**	Semitic	Hebrew
et	Estonian	Uralic	Latin	ps	Pashto	Iranian	rabic
fi	**Finnish**	Uralic	Latin	fa	**Farsi**	Iranian	rabic
hu	**Hungarian**	Uralic	Latin	am	mharic	Ethopian	Ge'ez
lv	Latvian	Baltic	Latin	ff	Fulah	Niger-Congo	Latin
lt	**Lithuanian**	Baltic	Latin	ha	Hausa	fro- siatic	Latin
sq	lbanian	lbanian	Latin	ig	Igbo	Niger-Congo	Latin
hy	rmenian	rmenian	rmenian	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	zerbaijani	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 enchmarks

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).
- **W T** — The W T competition covers sian languages paired with English. We consider data for Burmese-English (Riza et al., 2016), which contains news articles. W T 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 W T 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.
- **utshumato** — utshumato[4] is an 11-way parallel dataset comprising 10 frican 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 frica; Department of rts and Culture, South frica
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 rabic we use the QCRI rabic 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 CC ligned[17] (El-Kishky et al., 2020). In the following, we describe our mining strategy and summarize the main ideas of CCMatrix and CC ligned. We refer the reader to the references for a detailed description of the approaches.

Mining parallel data with L SER. 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 (rtetxe and Schwenk, 2018b; Chen et al., 2020; Kvapilíková et al., 2020). We focus on the embeddings generated by the L SER encoder, which enables the comparison of sentences in 94 different languages (rtetxe and Schwenk, 2018a). We then retrieve parallel corpora efficiently using F ISS indexing (Johnson et al., 2019). L SER embeddings generalize to unseen languages, like sturian, 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/PyThaiN P/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. s 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 pproach. 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 F ISS (Johnson et al., 2019). Thus, we apply it to a selected subset of relevant pairs, as detailed in § 3.2.1.

CC ligned Mining pproach. CC ligned 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 L SER-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. nother 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 . 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 ridge 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. 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*. ll 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

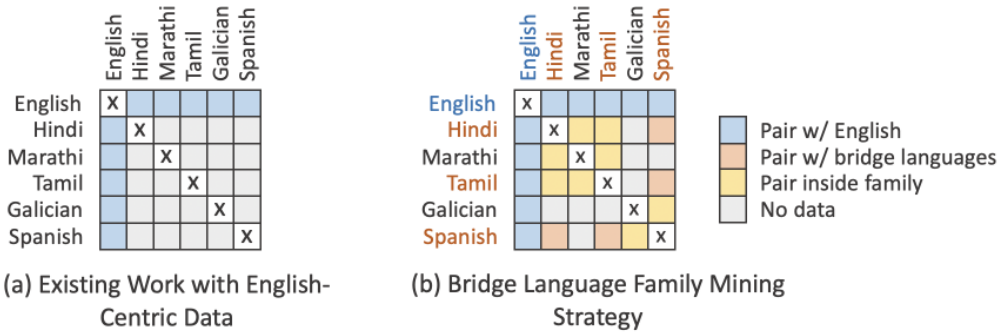

Figure 2: **Depiction of an English-Only data mining setting compared to the Bridge Language Mining Strategy**. We display a data matrix, where languages are shown on the X and Y axes. Data is mined in one direction (such as Hindi to Marathi) and used to train bidirectionally.

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- ryan family. ll 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.

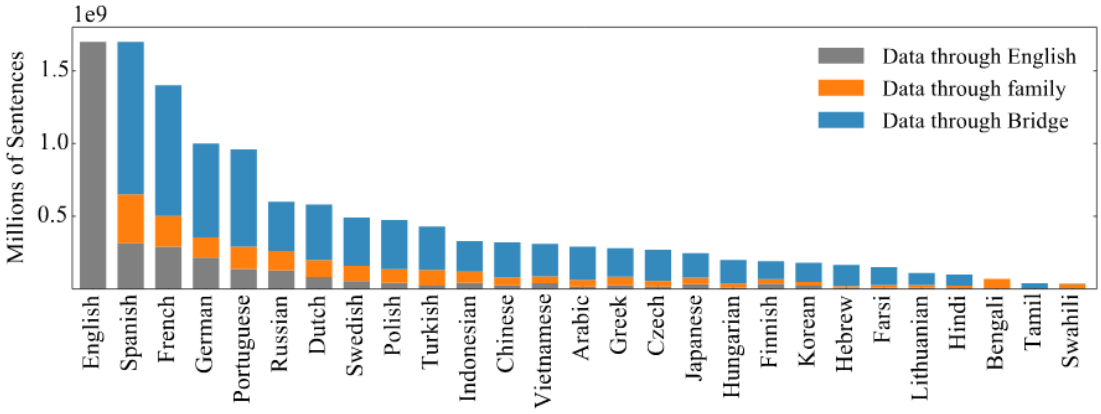

Figure 3: **Total mount of Data through Bridge Languages on our 100x100 Training Corpus**. We depict the amount of data through English (gray), amount of data through a bridge language not counting English (orange), and amount of data through the language grouping not counting bridge languages (blue).

Model	ll vg	Supervised Low	Mid	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**

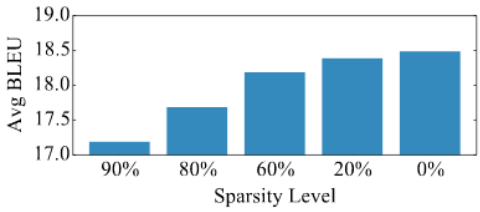

Table 2: **(left) Comparison of Sparse Mining Strategies**. We first hold the sparsity level fixed at 80% — compared to randomly selecting pairs to mine, the Bridge Language mining strategy performs very well. **(right) Bridge Language Strategy at Different Sparsity Levels**. We analyze different levels 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- entric + Random* strategy (Random 20% w/ En). In the Random strategy, mined pairs are randomly chosen, while in the *English- entric + 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 ugmenting 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 dditional 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

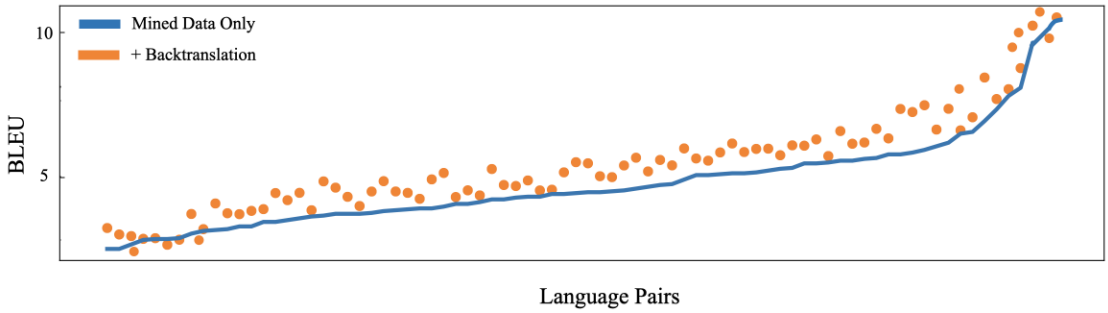

Figure 4: **Improvements from dding Backtranslated Data.** For each of the 100 language directions we explored by adding backtranslation. The blue line indicates the original model, where directions were selected if they had between 2 and 10 BLEU. The orange scatter indicates the effect of adding backtranslation.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

	Supervised				Zero-Shot	ll
Data sampling	**Low**	**Mid**	**High**	**vg**	**vg**	**vg**
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. s 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. standard solution is to rebalance each language independently with Temperature Sampling (rivazhagan et al., 2019), e.g. replacing the probability p_ℓ of a language by p_ℓ^{-}. 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 such that:

$$\max \text{ tr } (\quad \mathbf{Q}) \qquad \text{s.t.} \qquad 1_L = \mathbf{p}^{-}, \quad {}^{\top}1_L = \mathbf{p}^{-},$$

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 , 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 $\{0\ 1, 0\ 2, 0\ 3\}$.