Exploring Enhanced Code-Switched Noising for Pretraining in Neural Machine Translation

Vivek Iyer
School of Informatics
University of Edinburgh
v.iyer@sms.ed.ac.uk

Arturo Oncevay
School of Informatics
University of Edinburgh
aoncevay@ed.ac.uk

Alexandra Birch
School of Informatics
University of Edinburgh
a.birch@ed.ac.uk

Abstract

Multilingual pretraining approaches in Neural Machine Translation (NMT) have shown that training models to denoise synthetic code-switched data can yield impressive performance gains — owing to better multilingual semantic representations and transfer learning. However, they generated the synthetic code-switched data using non-contextual, one-to-one word translations obtained from lexicons - which can lead to significant noise in a variety of cases, including the poor handling of polysemes and multi-word expressions, violation of linguistic agreement and inability to scale to agglutinative languages. To overcome these limitations, we propose an approach called Contextual Code-Switching (CCS), where contextual, many-to-many word translations are generated using a \`base' NMT model. We conduct experiments on 3 different language families - Romance, Uralic, and Indo-Aryan - and show significant improvements (by up to 5.5 spBLEU points) over the previous lexicon-based SOTA approaches. We also observe that small CCS models can perform comparably or better than massive models like mBART50 and mRASP2, depending on the size of data provided. Lastly, through ablation studies, we highlight the major code-switching aspects (including context, many-to-many substitutions, code-switching language count etc.) that contribute to the enhanced pretraining of multilingual NMT models.

1 Introduction

Recent research in Neural Machine Translation (NMT) has focused on the pretraining of massively multilingual models (Aharoni et al., 2019; Siddhant et al., 2022; Costa-jussà et al., 2022) - due to their high scalability, easy deployability and state-of-the-art (SOTA) performances (Tran et al., 2021; Yang et al., 2021). One of the most common pretraining approaches trains a model to reconstruct (or \`\`denoise") a sentence noised using one or more mechanisms. Following masking (Song et al., 2019; Lewis et al., 2020), synthetic code-switching[1] has emerged as a more effective noising mechanism (Yang et al., 2020b; Pan et al., 2021). With the motive of moving the denoising task from language modeling to machine translation, these works propose to randomly code-switch input sentences, and then to train the MT model to denoise these sentences to the original monolingual ones.

Original sentence:	"If we don 't win , there will be some inquiries of why we haven't , " Graves told BBC Radio Leeds
AA-noised sentence:	" If noi annetada 't ויטוריה , そこ хочу jet sometime αιτήσεις seine kuna bize haven't , " Graves erzählte BBC Radio Leeds.

(a) Example sentences (Pan et al., 2021, Figure. 6)

Word	AA translation	GT translation	Word	AA translation	GT translation
annetada (Et)	don't	to donate	ויטוריה (He)	win	Vitoria
αιτήσεις (El)	inquiries	applications	some (En)	sometimes	some
хочу (Ru)	will	want	Bize (Tr)	to us	we

(b) Translation Errors in the noised sentence generated using AA. Parentheses indicate code-switching language.

Figure 1: Example of the errors induced by Aligned Augmentation (AA). GT refers to \`Ground Truth' translations, as provided by native speakers.

The most noteworthy system in this line of research is the massively multilingual model mRASP2 (Pan et al., 2021). It was pretrained using an algorithm called Aligned Augmentation (AA) - that constructs a synthetic code-switched sentence using dictionary-based word-level translations, followed by contrastive learning to semantically align this sentence with the reference sentence. Pan et al. (2021) showed that AA enables mRASP2 to achieve SOTA results across varied high, medium, and low-resource language pairs, and verified that this was due to improved multilingual semantic representations and enhanced cross-lingual transfer.

[1]In this paper (and related others), code-switching refers exclusively to \`synthetic' code-switching - which is employed as a pretraining approach for enhancing cross-lingual transfer learning. This is very different from the larger body of MT research that studies authentic, human-generated code-switching - as employed by bilingual speakers in informal contexts.

Despite being highly effective, a common weakness of this family of approaches is that they synthesize code-switched sentences using word lexicons - opening up several potential quality issues in the pretraining data. For instance, the SOTA algorithm AA as well as other related works (Lin et al., 2020; Li et al., 2022a) use non-contextual, one-to-one word translations obtained from MUSE lexicons (Lample et al., 2018) -- which can be problematic in a variety of cases. Firstly, failing to factor sentence-level context can cause violations in linguistic agreement, such as gender, case, tense and verb agreement. Secondly, AA cannot adequately handle contextual synonyms or polysemes (often estimated to constitute up to 80% of English words (Miller, 1998; Geeraerts, 1993)), with Lin et al. (2020) and Pan et al. (2021) assigning random word senses (and thus, translations) for polysemes -- regardless of context. Thirdly, one-to-one word translations create further issues, including the handling of multi-word expressions (eg. ``get out") and multi-word entities (eg. ``New York") -- with these problems aggravating for agglutinative languages. Finally, the bilingual MUSE lexicons themselves have been shown to be of dubious quality across a variety of languages (Kementchedjhieva et al., 2019), and using these for multilingual code-switching can propagate such errors manifold. This is illustrated in Figure 1, which shows an example of AA noising taken directly from Pan et al. (2021). With 6 errors in a sentence of 14 words (refer Appendix A.1 for a detailed examination of these errors), we contend that these limitations could cause significant corruption in the pretraining corpus.

We, thus, hypothesize that although AA has been effective in a variety of scenarios, the raised issues could lead mRASP2 to underperform. For this reason, we propose Contextual Code-Switching (CCS) - a novel approach for extracting contextual, many-to-many word translations, leveraging massive[2] NMT models, and then using these for noising the pretraining corpus. We conduct experiments on 3 different language families: Romance, Uralic, and Indo-Aryan, and report significant average improvements across the board, with gains of up to +5.5 spBLEU. We also find that CCS models narrow the gap with or outperform massively multilingual models like mBART50 (Tang et al., 2021) and mRASP2, despite using a tiny fraction of the data and compute. Lastly, we conduct ablation studies to analyze some of the most important factors to consider when synthesizing code-switched text for multilingual NMT pretraining - which constitutes another key, novel contribution of this work.

Our major contributions are, thus, as follows:

1. Firstly, we show that improving the quality of synthetic code-switching can significantly enhance pretraining of multilingual NMT models across various high, medium, low-resource and agglutinative language pairs (3.4.1).
2. Secondly, we demonstrate how massively multilingual NMT models can be harnessed to pretrain smaller models that yield comparable or better performance -- all while using a fraction of the training data and compute (3.4.2).
3. Thirdly, we empirically analyze and discuss some of the key factors that can enhance NMT pretraining on code-switched data - including context, many-to-many substitutions, code-switching language count, and fine-tuning - furthering scientific understanding (3.5)
4. Finally, for greater scalability of our approach, we propose useful variations of CCS that could alleviate potential resource dependencies (3.4.3) and increase efficiency (8.1) -- all while maintaining comparable performance.

2 Approach

2.1 Definitions

Given a set of N languages $L = L_1, L_2 \dots L_N$, multilingual NMT is defined as the task of learning a many-to-many mapping function θ from source language L_a to target language L_b. Code-switching refers to the phenomenon of shifting between two or more languages in a sentence. This work explores functions C that can synthetically code-switch corpora for pretraining multilingual NMT models.

2.2 Aligned Augmentation

Aligned Augmentation constructs synthetically code-switched datasets using multilingual lexicons. These lexicons are generated by interlinking bilingual MUSE dictionaries through a pivot language, English. Given a sentence S, a code-switched sentence $C_{AA}(S)$ is created by looking up word translations in the lexicon and, if available, substituting with replacement ratio r. A bilingual lexicon is used to code-switch parallel corpora and the multilingual one for monolingual data, with $r = 0.9$.

[2]`massive' in this work signifies the size of pretraining data, while `large' refers to the large Transformer architecture (with 12 encoders and 12 decoders)

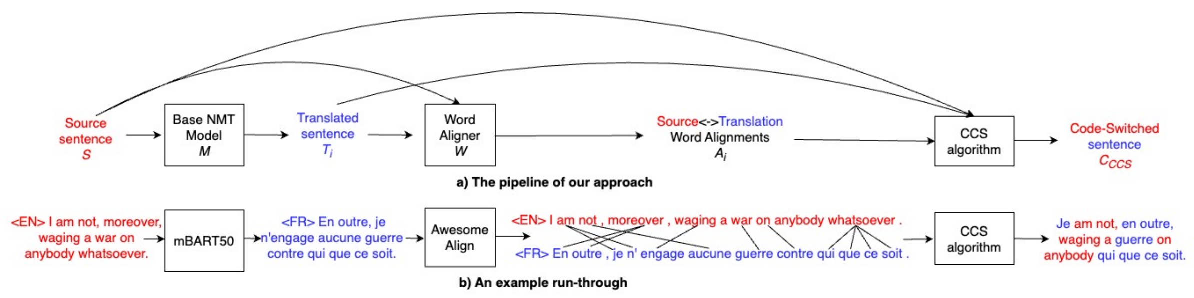

Source sentence S
Base NMT Model M
Translated sentence T_i
Word Aligner W
Source<->Translation Word Alignments A_i
CCS algorithm
Code-Switched sentence C_{CCS}
a) The pipeline of our approach
<EN> I am not, moreover, waging a war on anybody whatsoever.
mBART50
<FR> En outre, je n'engage aucune guerre contre qui que ce soit.
Awesome Align
<EN> I am not , moreover , waging a war on anybody whatsoever .
<FR> En outre , je n' engage aucune guerre contre qui que ce soit .
CCS algorithm
Je am not, en outre, waging a guerre on anybody qui que ce soit.
b) An example run-through

Figure 2: a) The pipeline of, and b) an example illustrating our approach. Alignments between punctuation marks have been omitted for ease of illustration. Color coding signifies language in the code-switched sentence $C_{CCS}(S)$. $C_{CCS}(S)$ is later fed to a Transformer and trained using Cross Entropy and Contrastive Loss (Section 2.4).

2.3 Contextual Code-Switching

Contextual Code-Switching (CCS) seeks to obtain contextual, many-to-many word translations, later used for code-switching parallel and monolingual corpora. Given a source sentence S, we generate the code-switched sentence $C_{CCS}(S)$ as follows:

1. Use a `base' NMT model M to translate S in n languages ($n \geq 1$) to obtain translations $\{T_1 \dots T_n\}$. Here, $M_{L_i}(S) = T_i$ where $M_{L_i}(S)$ is translation of S by M to L_i
2. Use word aligner W to align S with translations $\{T_1 \dots T_n\}$ and obtain word alignments $\{A_1 \dots A_n\}$; where $W(S, T_i) = A_i$
3. Generate the `noised' code-switched sentence $C_{CCS}(S)$ using the CCS algorithm, with S, $\{T_1 \dots T_n\}$ and $\{A_i \dots A_n\}$ as inputs

The CCS algorithm works by generating ``connected components'' of aligned words. For a given translation T_i and word alignment A_i, we iterate over each source word w_i in S, extract target words from T_i which are aligned with w_i (as specified by A_i), and then iteratively extract the source words aligned to these target words and vice versa, until convergence. This yields all possible many-to-many word alignment combinations, from which code-switching is carried out through random, iterative substitutions in S until the replacement ratio is reached -- yielding the final sentence $C_{CCS}(S)$. $C_{CCS}(S)$ could be code-switched in one ($n = 1$) or more ($n \geq 1$) languages, which we term Bilingual (BLCS) and Multilingual Code-Switching (MLCS) respectively. Although mRASP2 uses MLCS, we show in Section 3.5 that BLCS mostly performs better and is also more efficient. We illustrate our approach in Figure 3 and provide an example where the many-to-many substitution enables CCS to correctly translate the word moreover as en outre. Finally, we provide pseudo-code (containing finer technical details) in Algorithm 1 of Appendix A.2.

2.4 Training

To ensure a fair comparison, we replicate the training conditions proposed by Pan et al. (2021) while training our AA and CCS models. Our training dataset D consists of shuffled parallel and monolingual sentences, noised using the respective code-switching approaches. These code-switched sentences are input to the encoder. Meanwhile, the target sentences are the reference sentences for parallel corpora and the denoised (original) sentences for monolingual corpora. A special token indicating language ID is prepended to all source and target sentences. Finally, the model is trained using a loss function $\mathcal{L}$ that jointly optimizes Contrastive Loss $\mathcal{L}_{\text{CON}}$ and Cross Entropy Loss $\mathcal{L}_{CE}$ as follows:

$$\mathcal{L} = \mathcal{L}_{CE} + |s| * \mathcal{L}_{CON}$$

$$\text{where: } \mathcal{L}_{CE} = \sum_{x,y \in D} P_\theta(y|x), \text{and}$$

$$\mathcal{L}_{\text{CON}} = -\sum_{x,y \in D} \log \frac{e^{\text{sim}^+(\mathcal{E}(x),\mathcal{E}(y))/\tau}}{\sum_{a,b \in B} e^{\text{sim}^-(\mathcal{E}(x),\mathcal{E}(b))/\tau}}$$

Here, $\mathcal{E}$ denotes the average pooled encoder output, `sim' computes positive and negative semantic similarity for a pair of sentences, as denoted by sim^+ and sim^- respectively. Temperature τ controls the strength of penalties during contrastive learning and is set to 0.1. B denotes the mini-batch in dataset D that (x, y) belong to. As shown by Pan et al. (2021), Contrastive Learning aligns semantic representations of source and reference sentences (x, y) while pushing away all `negative' targets -- approximated to other reference sentences in the mini-batch for convenience. Finally, $|s|$ is the average sentence length (token count) that balances the token-level cross entropy loss and sentence-level contrastive loss.

3 Experiments

In this section, we seek to answer the following Research Questions:

1. How does CCS perform against the SOTA non-contextual algorithm, AA, and how does this vary across language pairs? (3.4.1)
2. How do small CCS models compare against SOTA massively multilingual models? (3.4.2)
3. How can CCS alleviate its potential resource dependencies and scale beyond? (3.4.3)
4. What are the key factors to consider when pretraining on code-switched text? (3.5)

As part of 3 Case Studies, we evaluate CCS on 3 different language families - namely, the high-resourced Romance, the agglutinative Uralic, and the low-resourced Indo-Aryan - in order to test its efficacy under different scenarios. For each family, we train from scratch small multilingual AA and CCS baselines on its languages. Small models have the benefit of minimizing negative interference, while also satisfying our resource constraints.

3.1 Corpora

Tables 1a and 1b show the training corpora statistics used per language family. We use this data for training all AA and CCS models in this work. For a fair comparison, only languages from each family that are present in the training set of mRASP2 and mBART50, and contain MUSE dictionaries are chosen for training. Portuguese (Pt) is taken as the zero-shot case, explored in Section 3.4.3, so no parallel corpus is used. The datasets constituting the training corpora are detailed in Section A.4.3.

For validation, we choose the last 1000 sentences from the bitext for each language. For testing, we use the latest available WMT test sets for each language into and out of English. Table 2 specifies these test sets, which are used for all experiments in this work. Regarding metrics, following related work, we use spBLEU-101[3] (Goyal et al., 2022) to evaluate all baselines in this work, but also provide ChrF++ (Popović, 2017) and COMET (Rei et al., 2020) results in Appendix A.5.1. We observe that these metrics largely agree with each other.

3.2 Preprocessing (CCS)

In order to apply the CCS algorithm (Section 2.3), we first generate `base' model translations and word alignments using the fine-tuned mBART50 and awesome-align (Dou and Neubig, 2021) models respectively. For the former, we use the corresponding multilingual 1-n, n-1, and n-n models (based on the language pair) and generate translations with a beam size of 5. For the latter, we fine-tune awesome-align with a subset (300K parallel sentences) from our training corpus using the Translation Language Modeling and Self-Training objectives, as suggested by Dou and Neubig (2021). Where possible, we attempt to have this subset uniformly distributed across all languages (except in low-resourced Indo-Aryan, where 12K En-Gu and 288K En-Hi sentences are used). This setup of `base' and word alignment models is used for training all CCS baselines in this work unless otherwise specified (such as in Section 3.4.3).

[3]referred to as spBLEU in this work for brevity

Romance				**Uralic**		**Indo-Aryan**	
En	7.5M	**It**	7.5M	**En**	20M	**En**	20M
Es	7.5M	**Ro**	7.5M	**Fi**	16M	**Hi**	6.25M
Fr	7.5M	**Pt**	7.5M	**Et**	8.1M	**Gu**	650K

(a) Monolingual data

Romance				**Uralic**		**Indo-Aryan**	
En-Es	1.8M	**En-It**	1.7M	**En-Fi**	4M	**En-Hi**	1.6M
En-Fr	1.8M	**En-Ro**	364K	**En-Et**	2.3M	**En-Gu**	12K

(b) Parallel data

Table 1: Statistics of training corpora used in this work

3.3 Experimental Settings

We use the vanilla Transformer (Vaswani et al., 2017) with 6 encoder and 6 decoder (6e6d) layers to train all models in this work, except in Table 3 - where `large' CCS baselines with 12 of each (12e12d) are used for fair comparison against massively multilingual models. We use a batch size of 4000 and a learning rate of 0.0001, with a polynomial decay scheduler and 5000 warm-up updates. We use an Adam optimizer with $\epsilon = 1e{-}6$. For regularization, we use dropout of 0.1 and weight decay of 0.001. We use automatic mixed precision and an update frequency of 4 to speed up training. We conduct validation every 1000 updates and use a patience value of 10 for early stopping. We train each model only once since a random seed of 0 is set everywhere. For tokenization, we use sentencepiece (Kudo and Richardson, 2018). Sentencepiece models using a unigram language model are trained on the corresponding corpora with a vocabulary size of 32000 and character coverage of 1.0. Following Pan et al. (2021), we use a replacement ratio of 0.9 in AA models while for CCS, we use 0.55, 0.75, and 0.1 for the Romance, Uralic and Indo-Aryan families after grid-search optimization. We detail infrastructure and training costs in Section A.4.

3.4 Results

3.4.1 Comparison with AA

In this section, we compare AA baselines with vanilla CCS models, when trained from scratch under identical conditions. Since Pan et al. (2021) already showed AA is the SOTA pretraining algorithm, we do not recreate other denoising approaches here - however, we do compare against two of the best-performing massive models from related work, mBART50 and mRASP2, in Section 3.4.2. Table 2 shows results for all 3 Case Studies.

We note some interesting trends. Firstly, while significant improvements are observed across all language families, the margin of gain varies. The highest gains are observed for the high-resourced Romance and the agglutinative Uralic families -- while the latter benefits greatly from many-to-many substitutions (Table 6), the former is quite noteworthy given that for Romance languages, MUSE lexicons are available in all directions (not just English-centric) so AA models would be strongest here. Meanwhile, the low-resourced Indo-Aryan family, which is non-agglutinative and suffers from low-quality `base' model translations (as observed from inspection of mBART50 translations by native speakers), the margin is relatively lower. Nonetheless, on average, CCS still outperforms the SOTA approach AA and, as we shall see in Section 3.4.2, even massive models like mBART50 and mRASP2 - despite using far lesser data overall. Moreover, we show in Table 7 how techniques like BiLingual Codeswitching (BLCS) can further boost CCS by +1 to +2 spBLEU, making the total gap against AA +2 to +3 spBLEU points for the Indo-Aryan family.

Secondly, we observe that CCS mostly performs equally well for En-X and X-En pairs (with few exceptions), but AA varies considerably and performs better for the latter. This could be because X-En is generally an easier translation task than En-X, owing to the abundance of high-quality English target-side data. CCS bridges the gap between these two tasks and improves consistency - likely due to the higher quality codeswitching systematically benefiting cross-lingual representations overall.

Lastly, we note that the spBLEU gains in Table 2 translate to comparably large improvements in ChrF++ and COMET (Table 10)

3.4.2 Comparison with Massively Multilingual Models

We now proceed to compare our CCS systems against two SOTA massively multilingual models mBART50 (Tang et al., 2021), trained on 50 languages, and mRASP2 (Pan et al., 2021), trained on 32 languages; in Table 3. Table 3a contains ratios of the monolingual and parallel data used by the massive models w.r.t. ours, per language family, and can help in a meaningful interpretation of the results in Table 3b. While we use significantly lesser data for the Romance family, we attempt to match the parallel data used by mRASP2 for the Uralic and Indo-Aryan families. This data is comparable to or slightly more[4] than that used for fine-tuning mBART50. As baselines, we use the AA and CCS models from Table 2 and their large (12e12d) variants. We also include a fine-tuned (ft) version of CCS (large) - created by pretraining on the code-switched monolingual data but leaving the parallel data unnoised for subsequent multilingual fine-tuning. We explore the impact of fine-tuning in greater detail in Section 3.5.

However, considering mBART50 and mRASP2 are massive models designed to scale to a much larger set of languages, we emphasize that Table 3 is *not* intended to be a head-to-head comparison; rather it is meant to position our work against the wider, popular SOTA. Our motive here is two-fold: a) to provide a way to harness massive models for pretraining and perform comparably or better, and b) to estimate the potential impact of scaling up CCS models - thus suggesting a worthy new direction of future exploration for pretraining better massive multilingual models. The first purpose could be especially useful for academics with fewer resources, while the second is likely better suited to groups capable of training massive models, eg. large industrial labs.

We achieve the first purpose by showing that the CCS (large) model performs comparably to mBART50 despite using substantially lesser data (such as in the Romance family), though mRASP2 retains a larger gap. But, when comparable data is used, as in the other families, CCS (large) models consistently outperform massive models by significant margins, despite using lesser monolingual data. The only exception is X-En where mBART50

[4]Relatively speaking here. Note that the overall data gap is still heavily biased towards mBART50, with up to 90x more monolingual data than our CCS models

	En-Es wmt13		En-Fr wmt14		En-It wmt09		En-Ro wmt16		Avg		Δ	
	→	←	→	←	→	←	→	←	→	←	→	←
AA	25.0	26.2	28.8	28.7	23.8	26.8	18.7	24.1	24.1	26.5	-	-
CCS	**30.7**	**29.1**	**33.1**	**30.9**	**29.1**	**29.0**	**25.4**	**30.4**	**29.6**	**29.9**	**+5.5**	**+3.4**

(a) Case Study 1: Romance languages

	En-Fi wmt19		En-Et wmt18		Avg		Δ	
	→	←	→	←	→	←	→	←
AA	15.6	19.3	20.5	23.3	18.05	21.3	-	-
CCS	**21.2**	**21.2**	**25.6**	**25.7**	**23.4**	**23.45**	**+5.35**	**+2.15**

(b) Case Study 2: Uralic languages

	En-Hi wmt19		En-Gu wmt18		Avg		Δ	
	→	←	→	←	→	←	→	←
AA	**28.4**	**24.6**	10.2	11.5	19.3	18.05	-	-
CCS	28.0	24.0	**12.9**	**12.9**	**20.45**	**18.45**	**+1.15**	**+0.4**

(c) Case Study 3: Indo-Aryan languages

Table 2: Pair-wise spBLEU results for each conducted case study. → stands for En-X while ← means X-En. `Avg' indicates average spBLEU, and Δ signifies spBLEU improvements over AA.

	Romance	**Uralic**	**Indo-Aryan**
Monolingual data ratios			
mBART50 (ft)	x91	x68	x90
mRASP2	x7	x4	x4
AA/CCS	x1	x1	x1
Parallel data ratios			
mBART50 (ft)	x9	x0.5	x0.8
mRASP2	x8	x1	x1
AA/CCS	x1	x1	x1

(a) Data Ratios

	Romance		**Uralic**		**Indo-Aryan**	
	En-X	X - En	En - X	X-En	En-X	X-En
Massively Multilingual Models						
mBART50 (large, ft)	32.25	35.63	23.10	**29.35**	**13.45**	**23.20**
mRASP2 (large)	**36.00**	**37.13**	**25.20**	27.00	5.75	15.10
Our Models						
AA	24.08	26.45	18.05	21.30	19.30	18.05
AA (large)	29.18	29.53	21.25	23.55	20.20	18.65
CCS	29.58	29.85	23.40	23.45	20.45	18.45
CCS (large)	31.30	31.10	27.60	27.25	23.30	22.00
CCS (large, ft)	**31.30**	**31.13**	**27.70**	**28.05**	**25.30**	**23.50**

(b) Results (large = 12e12d, ft = fine-tuned)

Table 3: Data and performance comparison with massively multilingual models. mBART50 (Tang et al., 2021) and mRASP2 (Pan et al., 2021) were taken and evaluated on the given pairs. Our Models were trained and tested only on languages from specific families (4 for Romance, 2 for Uralic and Indo-Aryan) on a much smaller dataset (Table 3a).

performs better, likely due to the wide gap in English monolingual and target-side data. However, the fine-tuned CCS model does close the gap with fine-tuned mBART50, performing comparably or better. We address the second purpose by noting that except for the low-resourced Indo-Aryan case, mRASP2 - which is essentially a scaled-up version of AA (large) trained on more languages - routinely improves over the latter. While `forgetting' low-resource languages can lead to some performance decline, performance, in general, can be observed to improve on scaling up - likely boosted by improved cross-lingual transfer. Now, when using comparable data, CCS (large) beats *both* AA (large) and mRASP2. This suggests that scaling up CCS to train a massive model like mRASP2 could reasonably be expected to yield improvements over the latter. We leave this exploration to future work.

Given CCS leverages massive models to pretrain small, high-performing ones, we also explore another interesting auxiliary application of CCS, Knowledge Distillation (KD), in Section A.5.2, and show how it outperforms traditional KD baselines.

3.4.3 Alleviating Resource Dependencies

	Romance		**Uralic**		**Indo-Aryan**	
	En-X	X-En	En-X	X-En	En-X	X-En
CCS (*base*=ft. mBART50)	29.58	29.85	23.4	23.45	20.45	18.45
CCS (*base*=from-scratch)	30.05	29.40	23.1	22.95	20.25	19.55

Table 4: CCS with different `base' model choices. A model trained `from-scratch' can be used as a substitute for fine-tuned mBART50 with comparable performance.

Scaling beyond mBART50 While the above experiments use mBART50 as the `base' model, an important question to consider is how to scale to languages beyond the ones included in the same (or, more challengingly, any available massive model). We conduct an alternative set of experiments, where we train up small (6e6d) Transformer multilingual models ``from-scratch'' on our parallel data (Table 1b) and show in Table 4 that CCS baselines using these as `base' models consistently perform comparably to those using mBART50 as the `base'. This suggests that although massive models are readily available and convenient to use, CCS performance is not dependent on their existence, and small models trained from scratch can be a good substitute.

Zero-Shot Translation: A follow-up question to the previous solution of training models from scratch on parallel data is the zero-shot scenario: i.e. what happens if there is no parallel data available, such as for low-resource languages or non-English centric pairs? Table 5 addresses this scenario. Starting from a random `from-scratch' baseline trained on the Romance corpus in Table 1b (i.e. no En-Pt data provided), we observe that CCS baselines using the former as a `base' model introduce large gains over the same and over AA. The latter is a particularly strong baseline since it uses ground-truth MUSE lexicons from Pt to every other Romance language for code-switching. However, the enhanced multilingual code-switching of CCS can potentially achieve superior alignment of the Pt vocabulary with that of the other languages -- making CCS a better alternative than AA even if parallel data is unavailable for a given pair.

	En-Pt	Pt-En
From-scratch	1.30	3.40
AA	3.20	10.30
CCS (*base*=from-scratch)	**4.80**	**11.00**

Table 5: Zero-shot Translation

3.5 Analysis

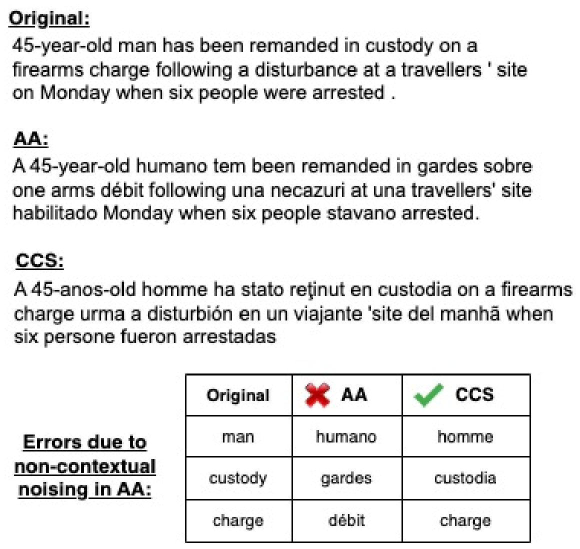
We now empirically discuss some key factors that enhance code-switched pretraining and hope these would serve as useful pointers for future work.

Impact of many-to-many substitutions Table 6 studies the impact of many-to-many substitutions through an ablation study, comparing against a CCS baseline where only 1-1 aligned words are chosen for substitution. We note a consistent decline in performance across all language pairs, with the largest being for Uralic (about 1.5-2 spBLEU points on average). This is likely due to agglutination in the Uralic family. For instance, 1 Finnish word jauhelihakeitto aligns to 3 English words: minced meat soup. This is correlated statistically - the average word count per sentence in our Fi corpus is 10.66 while for En it is almost double (20.2). Many-to-many substitutions are, thus, crucial for code-switching such agglutinative languages.

	Romance		**Uralic**		**Indo-Aryan**	
	En-X	X-En	En-X	X-En	X-En	En-X
CCS (1-1)	28.53	28.98	21.50	21.95	18.95	18.00
CCS	**29.58**	**29.85**	**23.40**	**23.45**	**20.45**	**18.45**

Table 6: Ablation study investigating the role of many-to-many substitutions

Impact of contextual translations Through an example lifted from the Romance corpus, Figure 3 shows how CCS improves pretraining data quality through contextual substitutions. While man means humano (human) in certain contexts, the French word homme is correct here. A similar argument holds for charge and gardes (garde could mean either guard or custody, based on context).



Original:
45-year-old man has been remanded in custody on a firearms charge following a disturbance at a travellers ' site on Monday when six people were arrested .

AA:
A 45-year-old humano tem been remanded in gardes sobre one arms débit following una necazuri at una travellers' site habilitado Monday when six people stavano arrested.

CCS:
A 45-anos-old homme ha stato reţinut en custodia on a firearms charge urma a disturbión en un viajante 'site del manhã when six persone fueron arrestadas

	Original	✗ AA	✓ CCS
Errors due to non-contextual noising in AA:	man	humano	homme
	custody	gardes	custodia
	charge	débit	charge

Figure 3: CCS produces more contextual substitutions. In this example taken from the Romance corpus, the sentences codeswitch between En, It, Fr, Es, Pt, and Ro

Impact of code-switching language count Pan et al. (2021) use Multilingual Code-Switching (MLCS)[5] to noise sentences, meaning a sentence could be code-switched in multiple languages. While we reproduce this in vanilla CCS and AA baselines, in Table 7 we explore Bilingual Code-Switching (BLCS), where 1 sentence switches between only 2 languages. Intuitively, this could be easier for the model to denoise. We observe our intuitions are mostly correct -- except for Romance, BLCS consistently improves performance. For the Romance languages, a likely explanation is that these have high lexical similarity (about 80-90% (Eberhard et al., 2022)) and more shared vocabulary, so the denoising task is not as complex and MLCS encourages greater transfer. In contrast, the lexical similarity for Uralic languages is lower than 50% (Jorgensen, 2020), while Hindi and Gujarati

[5]While this term is coined for ease-of-use in this work, it is a slight misnomer, and we discuss this in Appendix A.3

	Romance		Uralic		Indo-Aryan	
	En-X	X-En	En-X	X-En	En-X	X-En
CCS (MLCS)	**29.6**	**29.9**	23.4	23.45	20.45	18.45
CCS (BLCS)	28.2	28.58	**23.9**	**23.7**	**21.45**	**20.65**

Table 7: Comparison between MultiLingual (MLCS) and BiLingual Code-Switching (BLCS)

use different scripts -- leading to reduced vocabulary sharing and increased complexity. This might also explain why BLCS gives the Indo-Aryan family the highest gains, followed by Uralic. We note BLCS is also more time-efficient in Section 8.1, so we suggest future work to use the same when code-switching lexically dissimilar languages.

Impact of fine-tuning While the default variant of CCS follows Pan et al. (2021) and mixes parallel and monolingual code-switched data for pretraining, we explore if pretraining on only the latter and leaving the parallel data `unnoised' for fine-tuning, might be a better alternative (as is common in other related works). Table 8 confirms this enhances performance significantly, likely due to monolingual data being abundant enough for achieving the cross-lingual transfer desired during pretraining, and fine-tuning more closely resembling the final translation task. Secondly, we note that Multilingual Fine-Tuning (MLFT) beats Bilingual Fine-Tuning (BLFT) in code-switched pretraining too, complementing the findings of Tang et al. (2021) in masked pretraining.

	Romance		Uralic		Indo-Aryan	
	En-X	X-En	En-X	X-En	X-En	En-X
CCS	29.58	**29.85**	23.40	23.45	20.45	18.45
CCS + BLFT	28.65	28.43	23.55	23.80	16.35	14.50
CCS + MLFT	**30.00**	29.68	**25.20**	**25.85**	**23.55**	**22.35**

Table 8: Improvements yielded by fine-tuning

4 Related Work

Denoising-based pretraining: Various noising mechanisms have been proposed for denoising-based pretraining of NMT models in recent times. Inspired by BERT (Devlin et al., 2019), earlier models including MASS (Song et al., 2019) and BART (Lewis et al., 2020) were pretrained on masked monolingual corpora, followed by fine-tuning on large parallel datasets (Tang et al., 2021). In an effort to shift the denoising objective from language modeling to translation, subsequent works adopted code-switched noising. ALM (Yang et al., 2020a) introduced this concept by using statistical phrase tables to code-switch parallel datasets and training bilingual MT models that showed small improvements for the high-resourced, linguistically similar En-De and De-En pairs. Next, CSP (Yang et al., 2020b) proposed using probabilistic lexicons for code-switching in order to train bilingual models on both monolingual and parallel data. RAS (Lin et al., 2020) extended this trend to multilingual NMT, utilizing MUSE lexicons to code-switch and pretrain the massive NMT model mRASP on parallel corpora from 32 languages. Its successor, Aligned Augmentation (Pan et al., 2021) used a `multilingual' lexicon (formed by heuristically chaining bilingual MUSE lexicons) to code-switch both monolingual and parallel corpora and pretrained the mRASP2 model on these using contrastive learning. They reported SOTA scores, beating mRASP and many other strong baselines across a variety of language pairs and tasks. CeMAT (Li et al., 2022b) showed that BART-like masking can complement lexicon-based code-switching.

Different from all these works that only attempt one-to-one, non-contextual code-switching, the key novel contribution of our work is to carefully explore and analyze the performance gains offered by enhanced code-switching that factors context, many-to-many substitutions, code-switching language count, etc. We show how modern NMT models can be utilized to achieve these goals and achieve comparable or better performance while using a tiny fraction of the data and compute.

5 Conclusion

We explore a noising mechanism called Contextual Code-Switching (CCS) that extracts contextual, many-to-many word translations for code-switched pretraining in multilingual NMT. Our experiments, conducted on 3 different language families, show that CCS consistently beats the previous SOTA approach, Aligned Augmentation and also performs comparably or better than mBART50 and mRASP2, based on the quantity of training data provided. We analyse the impact of some major factors responsible for enhancing code-switched pretraining through examples and ablation studies. We hope the findings of this work will be useful to researchers studying NMT pretraining, as well as to academic and industry peers who may be looking for a way to fruitfully leverage massive NMT models, or conversely, to jump-start the training of even larger and better-performing ones.

6 Acknowledgements

This work was funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee 10039436. The experiments in this work were conducted using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service[6], Sulis Tier 2 HPC platform hosted by the Scientific Computing Research Technology Platform at the University of Warwick, and the Baskerville Tier 2 HPC service[7] operated by Advanced Research Computing at the University of Birmingham. Finally, we would also like to acknowledge Guillem Ramirez for his help in selecting interesting examples for Figure 3.

7 Ethical Considerations

There has been significant concern recently over massive multilingual NLP models learning racial and gender biases during pretraining (Tan and Celis, 2019; Bender et al., 2021). A technique like CCS that leverages massive NMT models could be at risk of propagating any such biases present in the `base' model. While such a limitation is not unique to CCS and could apply to any technique harnessing large models (such as Knowledge Distillation approaches), it is an important ethical concern since model biases that are propagated this way could be harder to detect and control - as compared to data biases. In such a situation, it could be worthwhile to invest effort into curating more `unbiased' data, and then using models trained from scratch on this data as the base models for CCS (see Table 4) - giving a greater degree of control than massive models like mBART50 but potentially yielding comparable performance.

8 Limitations

We now discuss some limitations of our work and suggest some ways to mitigate them.

8.1 Cost

One of the advantages of AA is that it is relatively inexpensive to code-switch using lexicons. CCS, on the other hand, requires translating training data into multiple different languages followed by computing word alignments, which can be very expensive, particularly on scaling up. So, we suggest some ways of reducing the cost, while potentially maintaining comparable performance. One effective way would be to use the BLCS variant, since it only needs one translation and one set of word alignments per sentence. Another way to reduce costs is to use smaller (6e6d) models trained from-scratch as a faster substitute for larger models like mBART50 (Table 4). The effectiveness of these techniques is shown in Table 9, which depicts the total preprocessing costs (for the entire pipeline in Figure 3) for code-switching a 4M En-Fi parallel corpus on a single GPU node (with 3 A100 GPUs). It is worth remembering that the word alignment costs are minimal here (about 30 minutes), so the costs are primarily due to generating translations (with a beam size of 5). We, thus, encourage using standard techniques to improve MT efficiency, like using lower beam size, shortlisting, quantization etc. to further reduce costs.

Algorithm	Time
CCS-MLCS (*base*=mBART50)	8h 37m
CCS-BLCS (*base*=mBART50)	4h 30m
CCS-MLCS (*base*=from-scratch)	4h 42m
CCS-BLCS (*base*=from-scratch)	2h 37m

Table 9: Total Preprocessing (base translation+word alignment+CCS) time costs for En-Fi 4M corpus, while using 1 GPU node of 3 A100 GPUs

We will also release the code-switched corpora constructed in this research as part of the camera-ready version of this paper, to ensure greater reuse of the expenditure in our time and resources.

8.2 Resource Dependencies

The CCS models in our work function using mBART50 as the `base' model and the word alignment model, awesome-align. Greater resource dependencies are, thus, another limitation of CCS and it is important to think of viable alternatives in case of non-availability of these. Awesome-align uses representations from mBERT (Devlin et al., 2019) so, it could scale to the languages the latter is pretrained on. For other languages, such as very low-resource pairs, it could be worth exploring low-resource word aligners (Asgari et al., 2020; Poerner et al., 2018) - though we leave the exploration of the same as part of future work. As for the `base' model, we could use models trained from scratch as a viable alternative (see Table 4) and potentially obtain comparable performance. In case of non-availability of parallel data, this approach can scale

[6]www.csd3.cam.ac.uk

[7]https://www.baskerville.ac.uk/

well to zero-shot translation when parallel corpora from related language pairs are available. (Table 5)

8.3 Low-Resource Scenarios

As we saw in Table 10, CCS appears to yield relatively lower gains for low-resource languages. While this does beat many SOTA models including mBART50 and mRASP2, further research is needed to adapt CCS better for data-scarce and low-resource scenarios in general. Based on related work, one useful solution could be to leverage data from high-resourced languages and families (eg. mixing Romance language data with Indo-Aryan languages) in a more multilingual and scaled-up iteration of our work. Another way would be to filter out low-quality translations from the `base' model using its confidence scores, and only use the high-quality ones for code-switching. While we are unable to explore these within the scope of this work, they could make for interesting future directions.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874--3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1--88, Online. Association for Computational Linguistics.

Ehsaneddin Asgari, Masoud Jalili Sabet, Philipp Dufter, Christopher Ringlstetter, and Hinrich Schütze. 2020. Subword sampling for low resource word alignment. *arXiv preprint arXiv:2012.11657*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. 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, Florence, Italy. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610--623.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171--4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112--2128, Online. Association for Computational Linguistics.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2022. *Ethnologue: Languages of the World*, 25 edition. SIL International. Online version: http://www.ethnologue.com.

Dirk Geeraerts. 1993. Vagueness's puzzles, polysemy's vagaries.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 Evaluation Benchmark for Low-Resource and Multilingual Machine Translation. *Transactions of the Association for Computational Linguistics*, 10:522--538.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Paul Jorgensen. 2020. How similar are finnish and estonian? https://langfocus.com/language-features/how-similar-finnish-and-estonian. Accessed: 2022-10-18.

Yova Kementchedjhieva, Mareike Hartmann, and Anders Søgaard. 2019. Lost in evaluation: Misleading benchmarks for bilingual dictionary induction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3336--3341, Hong Kong, China. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of machine translation summit x: papers*, pages 79--86.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66--71, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871--7880, Online. Association for Computational Linguistics.

Pengfei Li, Liangyou Li, Meng Zhang, Minghao Wu, and Qun Liu. 2022a. Universal conditional masked language pre-training for neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6379--6391, Dublin, Ireland. Association for Computational Linguistics.

Pengfei Li, Liangyou Li, Meng Zhang, Minghao Wu, and Qun Liu. 2022b. Universal conditional masked language pre-training for neural machine translation.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649--2663, Online. Association for Computational Linguistics.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 244--258, Online. Association for Computational Linguistics.

Nina Poerner, Masoud Jalili Sabet, Benjamin Roth, and Hinrich Schütze. 2018. Aligning very small parallel corpora using cross-lingual word embeddings and a monogamy objective. *arXiv preprint arXiv:1811.00066*.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612--618.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Divyanshu Kakwani, Navneet Kumar, et al. 2022. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages. *Transactions of the Association for Computational Linguistics*, 10:145--162.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685--2702, Online. Association for Computational Linguistics.

Aditya Siddhant, Ankur Bapna, Orhan Firat, Yuan Cao, Mia Xu Chen, Isaac Caswell, and Xavier Garcia. 2022. Towards the next 1000 languages in multilingual machine translation: Exploring the synergy between supervised and self-supervised learning. *arXiv preprint arXiv:2201.03110*.

Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the eu bookshop corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1850--1855.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926--5936.

Yi Chern Tan and L Elisa Celis. 2019. Assessing social and intersectional biases in contextualized word representations. *Advances in Neural Information Processing Systems*, 32.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2021. Multilingual translation from denoising pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3450--3466, Online. Association for Computational Linguistics.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021. Facebook AI's

WMT21 news translation task submission. In *Proceedings of the Sixth Conference on Machine Translation*, pages 205--215, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Philip Williams and Barry Haddow. 2021. The elitr eca corpus.

Jian Yang, Shuming Ma, Haoyang Huang, Dongdong Zhang, Li Dong, Shaohan Huang, Alexandre Muzio, Saksham Singhal, Hany Hassan, Xia Song, and Furu Wei. 2021. Multilingual machine translation systems from Microsoft for WMT21 shared task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 446--455, Online. Association for Computational Linguistics.

Jian Yang, Shuming Ma, Dongdong Zhang, ShuangZhi Wu, Zhoujun Li, and Ming Zhou. 2020a. Alternating language modeling for cross-lingual pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9386--9393.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020b. CSP:code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624--2636, Online. Association for Computational Linguistics.

A Appendix

A.1 Examining the Code-Switching Errors in Figure 1

In this subsection, we dissect Figure 1 and detail the mentioned errors, along with their causes[8]. Pre-training a multilingual model like mRASP2 that uses non-English corpora requires code-switching in non English-centric directions as well. Since the MUSE dictionaries (like most lexicons) are largely available in English-centric directions, Pan et al. (2021) attempt to generate a multilingual dictionary by recursively linking the bilingual MUSE lexicons through a pivot language (English). In theory, this would allow dictionary entries from language X to language Y using only X-En and En-Y dictionaries. However, such recursive linking would propagate any existing quality issues even further. We illustrate this point using examples from Figure 1. The Estonian word annetada and the Hebrew word ויטוריה are used as substitutions for don't and win respectively -- but they actually mean donate and Vittoria (a city in Spain), both of which have relatively less edit distance with don't and victory. A closer look at the Estonian[9] and the Hebrew [10] MUSE dictionaries, as well as the multilingual dictionary[11] constructed by Pan et al. (2021) confirms that this has been caused by the process of linking noisy bilingual dictionaries. The latter error, for instance, was caused by linking win to victoria (Portuguese), which was then aligned with Vittoria (Italian) and then Vittoria (English) - completely altering the meaning. A similar explanation can be drawn for how the English word some is incorrectly substituted with the English word sometimes (Figure 1b), despite there being no English-English dictionary.

A.2 The CCS algorithm

The pseudo-code of the CCS algorithm is shown in Algorithm 1, along with finer details we were unable to describe previously.

A.3 A discussion on the MLCS terminology

Multilingual Code-Switching (MLCS), as described in Section 2, is a misnomer. In the work of Pan et al. (2021), code-switching is carried out using a bilingual (English-centric) lexicon for the parallel corpora, and a multilingual dictionary for the monolingual corpora. Thus, they use MLCS in a monolingual corpus with the multilingual dictionary, but only Bilingual Code-Switching (BLCS) in a parallel corpus. They do not explain the reason for this choice. In our work, we attempt to shed some light on this and explore the efficacy of BLCS, which is far more efficient for CCS (refer Section 8.1) and also performs comparably or better (Tables 2 and 3). We use the term MLCS in our AA and CCS baselines, therefore, to contrast with BLCS and for ease of use. It is worth noting, however, that the parallel corpus is still bilingually code-switched in the MLCS baselines, following Pan et al. (2021).

A.4 Experimental Settings

A.4.1 Computational Infrastructure

Due to expiry and low availability of GPU hours, we are forced to conduct our experiments on 3 dif-

[8]We did not include this in the main work since this is auxilliary to our paper's focus on contextual and many-to-many substitutions

[9]https://dl.fbaipublicfiles.com/arrival/dictionaries/et-en.txt

[10]https://dl.fbaipublicfiles.com/arrival/dictionaries/he-en.txt

[11]https://lf3-nlp-opensource.bytetos.com/obj/nlp-opensource/acl2021/mrasp2/synonym_dict_raw_dep3

Algorithm 1: Contextual Code-switching of a sentence using the CCS algorithm

```
Input  :  Sentence $S$; translations $T_1, T_2 \dots T_n$; alignments $A_1, A_2 \dots A_n$
Output :  Code-switched sentence $C_{CCS}(S)$
GenerateCCSCandidates $(S, T, A)$
    $Visited \leftarrow \varnothing$                    // Keeps track of words that have already been aligned
    $Candidates \leftarrow \varnothing$
    foreach word $w_i \in S$ do
        if $w_i \in Visited$ then
            continue
        end
        $SrcWords, TgtWords \leftarrow \{w_i\}, \varnothing$
        $PrevSrcWords, PrevTgtWords \leftarrow \varnothing, \varnothing$
        /* Generates many-to-many connected components of words */
        while true do
            /* Adds target words aligned to source words */
            $TgtWords \leftarrow TgtWords \cup A[w_j \forall w_j \in SrcWords]$
            if $PrevSrcWords == SrcWords$ or $PrevTgtWords == TgtWords$ then
                continue                                    // Convergence condition
            end
            $PrevSrcWords, PrevTgtWords \leftarrow SrcWords, TgtWords$
            /* Adds source words aligned to target words */
            $SrcWords \leftarrow SrcWords \cup A[w_j \forall w_j \in TgtWords]$
        end
        $Visited \leftarrow Visited + \{w_j \forall w_j \in SrcWords\}$
        $Candidates \leftarrow Candidates + (SrcWords, TgtWords)$
    end
    return $Candidates$
$CCSCandidates \leftarrow GenerateCCSCandidates(S, T_i, A_i) \forall (T_i, A_i)$
$C_{CCS}(S), Swaps \leftarrow "", \varnothing$
$Visited \leftarrow \varnothing$                    // Keeps track of words that have already been code-switched
while $|Visited|/|S| < ReplacementRatio$ do
    /* Randomly choose word(s) for substitution */
    $SrcWords, TgtWords = Random.Choice(CCSCandidates)$
    if $\exists w_i \in SrcWords\{w_i \in Visited\}$ then
        continue
    end
    $Swaps \leftarrow Swaps + (SrcWords, TgtWords)$
    $Visited \leftarrow Visited + \{w_j \forall w_j \in SrcWords\}$
end
$C_{CCS}(S) = S.Swap(SrcWords, TgtWords) \forall (SrcWords, TgtWords) \in Swaps$
return $C_{CCS}(S)$
```

ferent GPU clusters. To maintain comparability, however, we ensure that we use the same cluster for each case study - thus the training and evaluation of all baselines (be it CCS or AA), including the evaluation of the massive models, for a particular language family is always conducted on the same cluster. Additionally, we also ensure that the same parameters are used across all machines, and libraries with the same versions are installed.

Specifically, for the Romance family Case Study we use Skylake CPU nodes with the maximum memory of 300GB and Ampere GPU nodes with 1000GB RAM and 4 Nvidia A100 GPUs per node, running CentOS8. For the Uralic family, we use nodes with 2 AMD EPYC 7742 (Rome) CPUs and 512GB RAM, while for GPUs we use 3 Nvidia A100 GPUs with 40 GB RAM. For the Indo-Aryan family, we use Intel® Xeon® Platinum 8360Y CPU nodes with 512 GB RAM and GPU nodes with 4x NVIDIA A100 40GB GPUs.

A.4.2 Computational costs

Total preprocessing costs for the En-Fi 4M pair are given in Section 8.1 and, based on the technique used, can be roughly interpolated given our training corpora size (Table 1) to calculate total GPU hours. In practice, the real-world time was much lesser since we: a) used GPU clusters (as mentioned above) to simultaneously process multiple language pairs by submitting multiple jobs , and b) divided large corpora into smaller halves that were simultaneously preprocessed. For example, the 20M English monolingual corpus used in the Uralic family (Table **??**) was divided into 2 halves of 10M that were submitted as part of 2 separate jobs.

For training, we only used 1 GPU node (1 Slurm job) per language family with 3 or 4 GPUs, depending on the cluster. Training costs for the Romance model took about 50 hours till convergence, 32 hours for Uralic and 61 hours for the Indo-Aryan models. The discrepancy in time is likely due to the fact that each of these experiments had to be run on separate clusters (as mentioned in A.4.1) with different architectures and different batch sizes, especially given each model took very similar number of steps until convergence (270K-290K updates in total).

A.4.3 Datasets used

All the datasets we use are publicly available, distributed freely with the CC0 license. For all case studies, News Crawl (Akhbardeh et al., 2021) is chosen to make up the monolingual corpus. For parallel corpora, we use different sources for each language family. For the Romance family, we use the Europarl corpus (Koehn, 2005) as our parallel corpus. For Uralic, we use the WMT (Barrault et al., 2019), EUBookshop (Skadiņš et al., 2014) and the ELITR-ECA (Williams and Haddow, 2021) corpora. Finally, for Indo-Aryan, we use the Samanantar (Ramesh et al., 2022) corpus.

Except for Samanatar, all of these datasets belong to the news domain. While Samantar is a collection of datasets from various domains, given the test set belongs to the news domain, we sample news datasets from this corpus for inclusion in our training data.

A.5 Additional Results

A.5.1 Results on other metrics

We summarize the results of our key models in Table 10, using spBLEU, ChrF++ and COMET metrics. While we were unable to include the same in our main work, we observe that the metrics agree with each by and large and detail it in this section for completeness.

A.5.2 CCS for Knowledge Distillation

While the results in the main work mostly focused on the efficacy of CCS in its primary role as a pretraining mechanism, Table 11 indicates how it could also be effective as a better Knowledge Distillation (KD) technique, with minimal computational overload. We compare against the vanilla KD baseline (Hinton et al., 2015) that trains a small (6e6d) student model to mimic the teacher model (mBART50). CCS models of the same size routinely outperform KD, with the sole exception of X-En (Romance). It is interesting to note that it takes similar computational resources to preprocess and train CCS (BLCS), as it does for the KD baseline: given a translation generated by a teacher model, it appears it is better to use the translation to noise (code-switch) the source sentence and train it using the CCS mechanism, as opposed to using it as a target. The only overhead for CCS would be that of extracting word alignments, and in practice we find that it is relatively small - about 1/16th the time taken for translation generation.

	En-X			X-En		
	spBLEU	ChrF++	COMET	spBLEU	ChrF++	COMET
Massively Multilingual Models						
mBART50 (large, ft)	32.25	54.60	0.59	35.63	57.43	0.55
mRASP2 (large)	**36.00**	**57.18**	**0.69**	**37.13**	**58.50**	**0.59**
Language family-specific baselines						
AA (MLCS)	24.08	46.75	0.13	26.45	51.28	0.20
AA (MLCS) (large)	29.18	51.65	0.39	29.53	53.38	0.35
CCS (BLCS)	28.20	51.83	0.38	28.58	53.13	0.31
CCS (MLCS)	29.58	52.40	0.44	29.85	53.78	0.36
CCS (MLCS, large)	31.30	**53.93**	**0.51**	31.10	**54.88**	**0.43**
CCS (MLCS, large, ft)	**31.30**	53.90	**0.51**	**31.13**	54.53	**0.43**

(a) Case Study 1: Romance languages

	En-X			X-En		
	spBLEU	ChrF++	COMET	spBLEU	ChrF++	COMET
Massively Multilingual Models						
mBART50 (large, ft)	23.10	46.95	0.72	**29.35**	**52.35**	**0.52**
mRASP2 (large)	**25.20**	**48.55**	**0.75**	27.00	50.75	0.47
Language family-specific baselines						
AA (MLCS)	18.05	41.30	0.19	21.30	46.00	0.20
AA (MLCS) (large)	21.25	44.45	0.43	23.55	47.95	0.32
CCS (BLCS)	23.85	46.45	0.64	23.70	47.70	0.35
CCS (MLCS)	23.40	46.05	0.61	23.45	47.45	0.34
CCS (MLCS, large)	27.60	49.35	**0.80**	27.25	50.65	0.48
CCS (MLCS, large, ft)	**27.70**	**49.70**	0.77	**28.05**	**51.70**	**0.51**

(b) Case Study 2: Uralic languages

	En-X			X-En		
	spBLEU	ChrF++	COMET	spBLEU	ChrF++	COMET
Massively Multilingual Models						
mBART50 (large, ft)	**13.45**	**25.00**	**-0.17**	**23.20**	**47.85**	**0.43**
mRASP2 (large)	5.75	22.90	-0.99	15.10	35.60	-0.15
Language family-specific baselines						
AA (MLCS)	19.30	36.50	0.18	18.05	42.30	0.19
AA (MLCS) (large)	20.20	37.10	0.23	18.65	41.90	0.14
CCS (BLCS)	21.45	38.55	0.27	20.65	44.35	0.24
CCS (MLCS)	20.45	37.05	0.23	18.45	41.55	0.18
CCS (MLCS, large)	23.30	40.20	0.40	22.00	45.90	0.31
CCS (MLCS, large, ft)	**25.30**	**42.00**	**0.54**	**23.50**	**47.90**	**0.38**

(c) Case Study 3: Indo-Aryan languages

Table 10: Average spBLEU, ChrF++ and COMET scores for all 3 case studies

	Romance		**Uralic**		**Indo-Aryan**	
	En-X	X-En	En-X	X-En	X-En	En-X
mBART50 (ft)	32.25	35.63	23.10	29.35	13.45	23.20
KD	28.90	**30.65**	18.05	20.30	5.05	16.80
CCS (MLCS)	**29.58**	29.85	23.40	23.45	20.45	18.45
CCS (BLCS)	28.20	28.58	**23.85**	**23.70**	**21.45**	**20.65**

Table 11: CCS v/s Knowledge Distillation (KD)