Synthetic Data Generation and Joint Learning for Robust Code-Mixed Translation

Anonymous EMNLP submission

Abstract

The widespread online communication in a modern multilingual world has provided opportunities to blend more than one language (*aka* code-mixed language) in a single utterance. This has resulted a formidable challenge for the computational models due to the scarcity of annotated data and presence of noise. A potential solution to mitigate the data scarcity problem in low-resource setup is to leverage existing data in resource-rich language through translation. In this paper, we tackle the problem of code-mixed (Hinglish and Bengalish) to English machine translation. First, we synthetically develop `HINMIX`, a parallel corpus of Hinglish to English, with ~$5M$ sentence pairs. Subsequently, we propose `JAMT`, a robust perturbation based joint-training model that learns to handle noise in the real-world code-mixed text by parameter sharing across clean and noisy words. Further, we show the adaptability of `JAMT` in a zero-shot setup for Bengalish to English translation. Our evaluation and comprehensive analyses qualitatively and quantitatively demonstrate the superiority of `JAMT` over state-of-the-art code-mixed and robust translation methods.

1 Introduction

Recent explosion of digital communication around the world has been marked by the growing use of informal language in online conversations. These conversations often feature the *use of words and phrases from multiple languages back and forth into a single utterance*: a phenomenon referred to as code-mixing (CM) or code-switching (Myers-Scotton, 1993, 1997; Duran, 1994). *Code-mixing* has become a standard practice both as a form of speech and text in multilingual communities such as Hindi-English, Spanish-English, Cantonese-Sanghaiese, etc., where people subconsciously alter between languages. Building upon this prominent use, it is imperative to model NLP systems for code-mixed technologies.

Traditionally, researchers have investigated the linguistic properties and grammatical structures of code-mixed languages (Poplack, 1978; Pfaff, 1979; Joshi, 1982). However, a few recent studies explored computational models for code-mixed languages in various domains such as Automatic Speech Recognition (ASR), Text to Speech (TTS), Sentiment Analysis, etc. (luo; Sitaram et al., 2019; Patwa et al., 2020). Due to the unavailability of annotated data, code-mixing in the domain of text remains vastly unexplored. With no official references of CM text in books and articles, online social networks (OSNs) remain the only source of mixed data collection. Further, the real-world unstructured text is highly susceptible to typographical errors and misspellings. These mistakes become more prevalent when languages written in non-romanized scripts such as Hindi, Japanese, etc. are adopted to code-mixed scenarios as each word in the originating script can be mapped to multiple probable transliterations. The problem is exacerbated by the multilingual nature of online code-mixed content, making it essential to understand CM concerning a common language.

In order to circumvent all these challenges, we propose robust code-mixed translation using a joint learning model, named **J**oint **A**dversarial **M**achine **T**ranslation (`JAMT`). Neural Machine Translation (NMT) models have become state-of-the-art in sequence-to-sequence tasks (Sutskever et al., 2014; Bahdanau et al., 2015). At the root of this advancement are two interrelated issues: (i) NMT models need a vast amount of parallel data for satisfactory performance; and (ii) NMT models are brittle to even a slight amount of input noise (Belinkov and Bisk, 2018). First, to handle the scarcity of code-mixed parallel data, we construct a synthetic Hinglish-English dataset by leveraging a bilingual Hindi-English (Hi-En) corpus. For this, we identify various grammatical and semantic patterns in the continuous switching of two languages and formulate a general pipeline for creating a *synthetic code-mixed corpus*. The generated parallel data is then passed through an *adversarial module* that injects different types of naturally occurring adversarial perturbations to generate a source-side noisy version of the code-mixed dataset. Inspired by multilingual NMT models, we train a joint model for translation of clean and noisy CM text to make the code-mixed translation robust to noisy input. Our experiments show that by *jointly training* both noisy and clean text in a multilingual setting, the model can encode diverse lexical variations of code-mixed words into the shared representation space; thereby, substantially improving the translation quality. Additionally, the need of a parallel CM corpus for every new language pair limits the applicability of NMT models for code-mixed translation. Further, the availability and accuracy of language specific POS-taggers, translation dictionaries, filtering tools become pivotal for building a synthetic CM corpus. To ease this challenge, we propose *zero-shot* code-mixed translation, where a bilingual Bengali-English (Bn-En) parallel corpus is trained along with a code-mixed Hindi-English parallel corpus. This way, the model learns to adapt to the multilingual scenario and translate Bengali CM text to English.

Precisely, the contributions of our work are summarized below:

- We propose a novel `JAMT` model for effectively translating real-world noisy code-mixed sentences to English.
- We release `HINMIX`, the first large-scale **Hin**glish Code-**Mix**ed parallel corpus consisting of ~$5M$ parallel sentences.
- We manually annotate 2787 gold standard CM sentences for the evaluation.
- We explore *Zero-Shot* Code-Mixed Translation for Bengali code-mixed to English translation without any parallel corpus.
- Through experiments and analysis, we demonstrate that `JAMT` significantly outperforms the previous state-of-the-art CM and robust MT approaches.

2 Related Work

In the past, various linguists (Verma, 1976; Joshi, 1982; Singh, 1985) studied the phenomena of CM and intra-sentential code-switching. The ubiquitous usage of CM in day-to-day spoken conversations and online written content coupled with the success of large supervised NLP systems in downstream classification and sequence generation tasks such as POS tagging, sentiment analysis, speech recognition, and translation brings up the necessity to generate labeled CM datasets. In 2018, Dhar et al. (2018) initiated the effort to create a 6K pair gold-standard Hindi-English CM dataset. Following this, synthetic CM data generation methods by utilizing parse trees (Pratapa et al., 2018), alignment learning (Rizvi et al., 2021) and copy mechanism (Winata et al., 2018) were proposed. Recently, Gupta et al. (2020, 2021) explored the linguistic properties to automatically generate CM sequence without parallel corpus by employing NMT models such as pointer generator (See et al., 2017) and pretrained mBERT (Devlin et al., 2019).

The presence of annotated code-mixed data does not ease the target task due to the extensive amount of typos, slang, and phonetic variations in the data; thus, making it implausible to overlook the robustness against noise of existing solutions. Several approaches (Belinkov and Bisk, 2018; Karpukhin et al., 2019; Passban et al., 2020) have studied the robustness of the model with respect to the dataset and training procedure. Cheng et al. (2018, 2020) adopted an adversarial stability training objective to build a perturbation-invariant encoder. Some of the recent works (Sato et al., 2019; Park et al., 2020) also adopted the regularization procedure for the adversarial effectiveness of NMT models. Although these schemes satisfy the robustness criteria of an NMT model, the nature of noise in the CM language largely remains unexplored.

Our proposed work is motivated by the gap in research to build an all-inclusive code-mixed translation system that handles the diverse switching nature in CM communities and is robust to any kind of CM noise. Furthermore, `JAMT` can translate multiple languages without the necessity to create individual CM datasets. The following section elaborates upon the methodology adopted to build the dataset and satisfy the mentioned criterion.

3 Dataset

In this section, we describe the pipeline used to create `HINMIX` utilizing IITB English-Hindi parallel corpus (Kunchukuttan et al., 2018). `HINMIX` consists of Hindi-English CM parallel pairs generated using two strategies – alignment-based and translation-based.

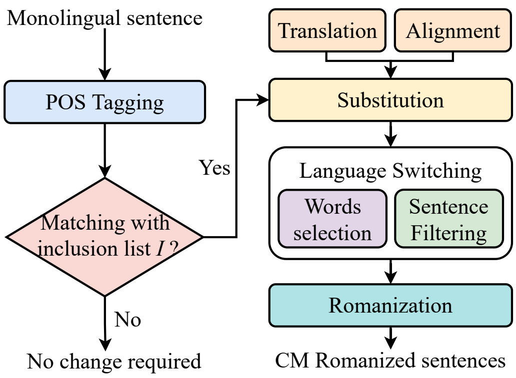

Figure 1: Pipeline of code-mixed data generation.

Code-Mixed Generation: Matrix Language Frame (MLF) model (Myers-Scotton, 1997) argues that the syntactic and morphological structure of any code-switch utterance comes from a Matrix Language (L_m) which borrows words from the Embedded Language (L_e). Following this theory, we characterize the asymmetric (Joshi, 1982) nature of intra-sentential code-mixing in Indian languages. After performing a linguistic study on a large number of CM tweets collected from Twitter, we conclude that the regional language acts as the base language L_m, and words are borrowed from English L_e for switching in the urban usage of hybrid text in Indian languages. Given a source-target sentence pair $S \parallel T$, we generate the synthetic code-mixed data by substituting words in the matrix language sentence with the corresponding words from the embedded language sentence. Figure 1 explains the code-mixed data generation pipeline.

Candidate Word Selection: We select *proper nouns* (NNP, NNPC, NNPS), *common nouns* (NN, NNC, NNS), *adjectives* (JJ), and *quantifiers* (QC, QCC, QO) to be part of an inclusion list I. All words whose POS tag belongs to the inclusion list are potential candidates for code-switching (c.f. appendix for detail).

Building Substitution Dictionary: Once the corpus is POS-tagged and candidate words are shortlisted, the substitute words from L_e need to be determined. We propose two approaches to build a substitution dictionary:

1. Translation Based: In any code-switch community, there is a code choice that is more favorable than other potential choices (Myers-Scotton, 1997). For example, a regular Hindi user would routinely use the English word "*help*" than the word "*assist*" due to its common usage. Moreover, NMT models show a similar property of memorizing commonly seen words in the corpus (Luong et al., 2015). Utilizing this correlation, we prepare a dictionary by training an `Hi-En` NMT model followed by context-independent word-by-word translation using the trained model. This method ensures a prevalent and consistent code-mixed vocabulary in the dataset.
2. Alignment Based: In this approach, an alignment model is trained between a source and target corpus to learn word-level correspondence between each parallel sentence. We use the fast-align (Dyer et al., 2013) symmetric alignment model to obtain the source-target alignment matrix. Next, a substitution dictionary for each sentence is obtained, consisting of only words with one-to-one source-target mapping. This approach allows us to deal with the word-sense ambiguity problem by substituting context-dependent foreign words in each sentence, thereby forming a diverse set of code-mixed vocabulary in the corpus.

For each sentence $S \parallel T$ in corpus, two substitution dictionaries are formed corresponding to the two approaches.

Language Switching: It might appear that the decision to switch a word is a binary choice and that every word in L_m can be replaced from the set of potential substitute words. However, the switching paradigm in a code-mixed utterance depends upon a range of factors such as the lexical information available with the speaker, their relative fluency and cognitive control in either language, speaker's intention to switch, and most importantly, the intrinsic structure of involved languages (Kroll et al., 2008). Hence, instead of substituting every candidate word and generating a single code-mixed sentence, we follow a randomized word-selection and filtering method to obtain multiple CM combinations of a single source sentence. Table 1 shows multiple generated Hindi code-mixed ($\texttt{Hi}_\texttt{c}$) sentences for a single sample using translation (T) and alignment (A) based approaches. To illustrate the need for sentence filtering, we rank from 1 to 5 (higher is better) to evaluate the quality of these CM sentences.

- *Word Selection:* Given that there can be $2^r - 1$ CM combinations in a sentence of r candidate words – computationally expensive for large r,

En Hi	The tendency to give physical training to the whole society resulted in many disastrous consequences. समस्त समाज को शारीरिक प्रशिक्षण देने के कारण बहुत से बुरे परिणाम हुए।	Rank ↑
A	whole समाज को physical training देने के कारण बहुत से बुरे परिणाम हुए।	3
A	whole society को physical training देने के कारण बहुत से बुरे consequences हुए।	5
T/A	समस्त society को physical training देने के कारण बहुत से बुरे परिणाम हुए।	5
T	all society को शारीरिक training देने के cause बहुत से evil results हुए।	2
T	समस्त society को physical training देने के कारण बहुत से बुरे results हुए।	4

Table 1: Sample of generated Hindi code-mixed ($\texttt{Hi}_c$) sentences using translation (T) and alignment (A) approach. Rank (↑) defines the quality assessment by humans.

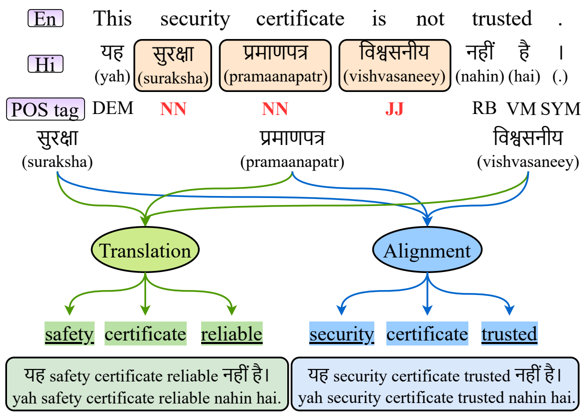

Figure 2: An example showing the process of code-mixed sentence generation using both method.

we adopt a set of heuristics (details in appendix) to limit the CM sentences to be generated.

- *Sentence Filtering:* To further narrow down the selection pool and incorporate language structures of bilingual languages into synthetic CM sentences, we use a combination of probabilistic and deterministic NLP evaluation metrics.
    1. We use an unsupervised cross-lingual XLM (Conneau and Lample, 2019) model to calculate the perplexity of CM sentences. We observe a good correlation between the fluency of the CM sentence and its perplexity, even when provided with Devanagari Hindi and English text in a single CM sentence.
    2. We employ code-mixed specific measures such as Code-Mixing Index (CMI) (Gambäck and Das, 2016) and Switch Point Fraction (SPF) (Gupta et al., 2020) to select sentences between a certain threshold, details of which are discussed in Section 5.

Figure 2 shows the generated CM sentences from both methods for a single sample. This forms our two code-mixed parallel datasets CTRANS and CALIGN from translation and alignment methods respectively with Hindi (Devanagari)-English CM pairs: $\texttt{Hi}_c$-En. Finally, for each case, we use Google Transliterate API[1] to produce the romanized version r of the CM parallel corpora – $\texttt{Hi}_{cr}$-En. In total, we obtain ~4.9M and ~4.2M parallel sentences using the translation and alignment strategies, respectively. A detailed statistics of the dataset is presented in appendix.

4 Joint Code-Mixed Translation

In this section, we describe our approach for robust translation of code-mixed sentences to English. We apply a language-free SentencePiece[2] tokenizer with a unigram subword model (Kudo, 2018) to generate a vocabulary directly from the raw text. The obtained synthetic CM text is then passed through an adversarial module to generate a noisy CM corpus. Subsequently, the clean and noisy corpora are simultaneously trained using the proposed JAMT model. A high-level architectural diagram of JAMT is illustrated in Figure 3.

Architecture: Inspired by the success of multilingual models, we leverage a sequence-to-sequence joint learning framework to translate code-mixed sentences to English. Unlike NMT models trained on a single language pair for one direction, the joint model consists of a single encoder and a decoder for different corpora (code-mixed/romanized/noisy) and directions allowing them to simultaneously learn useful information across language boundaries. For training the joint model from multiple sources to multiple targets (many-to-many), a proxy token for the target language is inserted at the beginning of the source sentence, indicating the intended target at the decoding stage as shown in Figure 3.

Training Objective: The joint model is trained to optimize the sum of categorical cross-entropy (CE) loss with label smoothing (Szegedy et al.,

[1] https://developers.google.com/transliterate/v1/getting_started

[2] https://github.com/google/sentencepiece

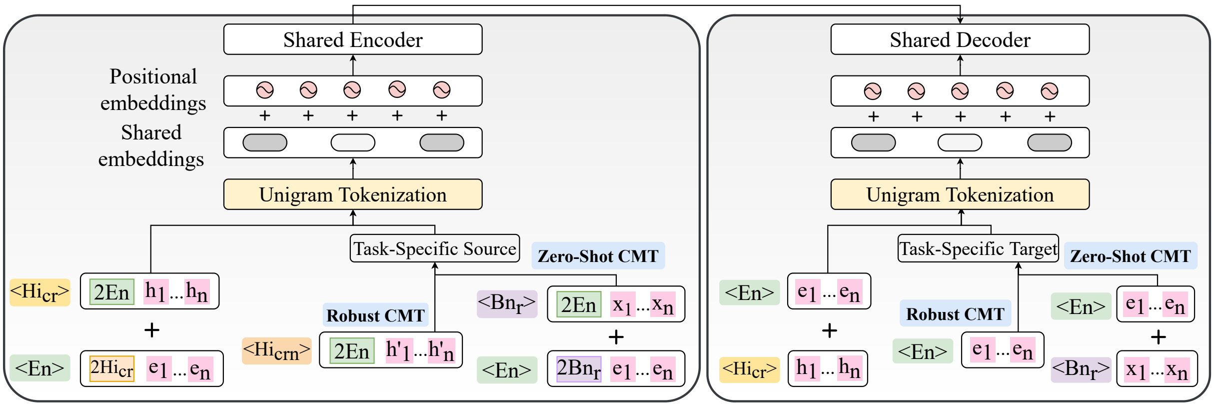

Figure 3: Architecture of our proposed `JAMT` model. Here, `Hi`, `En`, and `Bn` represent Hindi, English, and Bengali language, respectively. The subscripts `c`, `r`, and `n` are used to denote codemix, romanized, and noisy version of a dataset. The first token $[2T]$ in the encoder input indicates the intended target language T followed by tokens in the source language S. The target tokens are passed to the decoder sequentially for model training.

2016) across all language pairs. As our code-mixed datasets are synthetically prepared by replacing words using the matrix language framework (Myers-Scotton, 1997), learning the model directly using the CE loss would tend to memorize the labels for incorrect source tokens and degrade the model performance. Therefore, we adopt label smoothing to train our proposed model.

Adversarial Module: The transliteration of non-roman languages depends upon the phonetic transcription of each word, varying heavily with the writer's interpretation of involved languages. With no consistent spelling of a word, it becomes crucial to simulate the real-world variations and noise for the practical application of any CMT model. Hence, we propose to learn robust contextual representations by distorting the available clean corpora with word-level adversarial perturbations as follows (c.f. appendix for detail):

- **Switch:** "*t r a n s f e r*" vs "*t r a s n f e r*"
- **Omission:** "*a m a z i n g*" vs "*a m z n g*"
- **Proximity typo:** "*m o b i l e*" vs "*m o v i l e*"
- **Random Shuffle:** "*l a p t o p*" vs "*l o p t a p*"

We inject 30% switch, 12% omission, 12% typo, and 5% shuffle noise to $\texttt{Hi}_{\texttt{cr}}$ to produce a 60% word-level noisy code-mixed corpus $\texttt{Hi}_{\texttt{crn}}$-`En`. Both clean ($\texttt{Hi}_{\texttt{cr}}$-`En`) and noisy ($\texttt{Hi}_{\texttt{crn}}$-`En`) corpora are further used to train a joint model, which is described in the next subsection.

4.1 Robust Code-mixed MT (RCMT)

To capture the context-dependent lexical variations between the noisy and clean corpora, we formulate the cross-lingual translation setting to the code-mixed scenario, referred to as Robust Code-Mixed Translation (RCMT). For this, we jointly train a transformer model in three directions (RCMT_1) – bidirectional Hindi-English using *clean* code-mixed romanized corpus ($\texttt{Hi}_{\texttt{cr}} \rightleftharpoons \texttt{En}$) and Hindi to English using *noisy* code-mixed romanized corpus ($\texttt{Hi}_{\texttt{crn}} \rightarrow \texttt{En}$), where `c`, `r`, and `n` represent the code-mixed, romanized, and noisy versions of a dataset, respectively.

When a pair of a sentence from $\texttt{Hi}_{\texttt{cr}}$ and $\texttt{Hi}_{\texttt{crn}}$ are tokenized through the unigram model, the subwords tokens of both sentences would contain substantial amount of overlap due to the joint vocabulary. Any noise due to lexical, phonetic, or orthographic variations only perturbs the word at the character level, thereby obtaining similar subwords to some extent. Further, when translating two different sentences to the same target language, the joint model would learn the relationship between those subwords by utilizing their same syntactic and semantic properties. Therefore, the non-canonical nature of noisy text would benefit from the strong implicit supervision of clean sentences even when they are morphologically dissimilar.

Since both noisy and clean corpora follow the same origin (Devanagari Hindi), we also experiment with the robustness capabilities of `JAMT` by adding two non-romanized code-mixed directions in RCMT_1, representing it as RCMT_2: Devanagari $\texttt{Hi}_{\texttt{c}} \rightleftharpoons \texttt{En}$. This modification would enable `JAMT` to better handle the dependencies among Devanagari and romanized characters besides minimizing the morphological ambiguity across sentences.

4.2 Zero-shot Code-mixed MT (ZCMT)

The previous robust CMT approach uses the linguistic and lexical similarity of the corpora to learn robust representations effectively. However, to adapt code-mixed machine translation for any other language pair (e.g., Bengalish $\rightleftharpoons$ English), we need a code-mixed parallel corpus for the same, which is often unavailable. Therefore, to negate the limitation of data scarcity, we propose a zero-shot transfer learning approach for code-mixed translation for another language pair. In this approach, we use the previously generated CM corpora to exploit the transfer learning characteristic of cross-lingual models for code-mixed translation in an unseen pair. The idea is to utilize the existing non-code-mixed parallel corpus of language l_1 and a code-mixed parallel corpus of language l_2 for the translation of code-mixed sentences of l_1. To this end, we train JAMT with Bengali-English (Bn-En) and Hinglish-English ($\mathtt{Hi_{cr}}$-En) parallel corpora. Subsequently, the trained model is employed to convert a Bengalish sentence to English. We argue that the trained model would be able to transfer the code-mixing behaviour onto the network activations in a zero-shot way. We choose Bengali (Bn) due to the availability of both Bn-En large parallel-corpora (Hasan et al., 2020) and Bengali code-mixed test set $\mathtt{Bn_c}$-En (Gupta et al., 2021). The following language pairs are used to train the Zero-shot CM Translation (ZCMT) model:

- Code-mixed Hindi to English: Devanagari $\mathtt{Hi_c}\rightleftharpoons$En, romanized $\mathtt{Hi_{cr}}\rightleftharpoons$En, noisy romanized $\mathtt{Hi_{crn}}\rightarrow$En.
- Bengali to English: romanized $\mathtt{Bn_r}\rightleftharpoons$En and Eastern-Nagari Bn$\rightleftharpoons$En.

5 Experiments and Results

Depending upon the dataset and language pair, we evaluate JAMT on different tasks and configurations. Due to the unavailability of gold-standard code-mixed parallel test data, we limit our evaluation to two languages: Hindi (Hi) and Bengali (Bn), described as follows: **Hi-En:** We utilize the test and dev sets from WMT 2014 En-Hi shared task (Bojar et al., 2014). Two annotators who were fluent bilingual speakers of Hindi and English were asked to annotate the Hindi sentences to Hinglish manually. This gold-standard Hi-En code-mixed data consists of 280 (dev) and 2507 (test) CM utterances. **Bn-En:** For testing our ZCMT model, we make use of the Spoken Tutorial Bn-En CM test set released by Gupta et al. (2021). This data[3] consists of 28K utterances transcribed from courses related to engineering, programming, etc. We randomly selected 500 and 2000 sentences as the dev and test sets, respectively. We compute **SacreBLEU** (Ott et al., 2019) and **METEOR** (Banerjee and Lavie, 2005) to evaluate the quality of the translation.

Baselines: We conduct experiments with multiple CM and robust MT baselines for fair comparison of our JAMT approach: • **TFM:** We employ a vanilla Transformer with the same hyperparameters as JAMT for each configuration. • **FCN:** Following Gehring et al. (2017), we adapt seq2seq fully convolutional network for Robust CMT task. • **mT5:** Xue et al. (2021) put forward a "span-corruption" objective to pre-train a massive multilingual masked LM for sequence generation. • **mBART:** Liu et al. (2020b) used a seq2seq denoising-based autoencoder pre-trained on a large common-crawl corpus. • **MTNT:** Vaibhav et al. (2019) proposed to enhance the robustness of MT on the noisy text by pre-training an LSTM model with a clean corpus and fine-tuning it on noisy artificial data. • **MTT:** Zhou et al. (2019) presented a Multi-task Transformer for robust MT that uses dual decoders, one to generate the clean text and another to provide the translation given the noisy input. • **AdvSR:** Park et al. (2020) introduced an adversarial subword regularization scheme for on-the-fly selection of diverse subword segmentation in a sequence resulting in character-level robustness of an NMT model.

Code-mixed MT Results: Seq2Seq models such as transformers (TFM) and convolutional attention networks (FCN) have become the de-facto standard to evaluate MT systems (Liu et al., 2020a; Wu et al., 2019). Following their competitive performance in code-mixed translation tasks (Nagoudi et al., 2021; Appicharla et al., 2021; Dowlagar and Mamidi, 2021), we train individual models in each direction ($\mathtt{Hi_c}\rightarrow$En, $\mathtt{Hi_{cr}}\rightarrow$En, $\mathtt{Hi_{crn}}\rightarrow$En) for both the CTRANS and CALIGN datasets. Table 2 shows the superior performance of the transformer (TFM) over FCN with an average improvement of +2.47 & +2.68 BLEU scores across CM ($\mathtt{c}, \mathtt{c+r}$) and robust CM (c+r+n) translation models, respectively.

[3] https://github.com/shruikan20/Spoken-Tutorial-Dataset

Model	CTRANS						CALIGN					
	c		c + r		c + r + n		c		c + r		c + r + n	
	B	M	B	M	B	M	B	M	B	M	B	M
TFM	9.35	36.2	9.18	35.0	5.46	27.3	9.97	39.7	10.02	36.2	9.70	37.4
FCN	6.62	27.8	6.04	27.4	4.10	22.6	7.89	33.2	8.07	33.1	5.69	27.5
mT5	4.30	23.4	3.83	23.5	2.06	16.6	4.27	22.6	4.28	25.9	2.80	19.5
mBART	6.72	34.3	5.51	30.1	2.80	22.0	5.38	29.5	7.07	35.7	3.19	21.7
MTT	-	-	-	-	8.93	34.0	-	-	-	-	10.44	38.0
MTNT	-	-	6.76	29.8	4.26	22.3	-	-	8.48	35.1	5.92	28.0
AdvSR	-	-	6.64	30.5	2.62	19.1	-	-	9.63	36.7	7.28	32.7
RCMT$_1$	-	-	**12.91**	**43.0**	**10.25**	**37.7**	-	-	13.58	45.7	**11.54**	**41.5**
RCMT$_2$	**13.07**	**44.0**	12.83	43.0	9.79	36.9	13.81	46.2	**13.72**	45.7	11.3	40.8

Table 2: Baseline comparison of RCMT$_1$ and RCMT$_2$ from Hindi to English on CTRANS and CALIGN datasets. Here, c, r, and n denote codemix, romanized, and noisy version of a dataset. (B: SacreBLEU and M: METEOR)

Moreover, a substantial gain of +3.31B, +7.25M score (on avg.) over TFM is observed on the noisy corpus ($\mathtt{Hi_{crn}} \rightarrow \mathtt{En}$) when it is trained simultaneously with the clean corpora ($\mathtt{Hi_{cr}} \rightleftharpoons \mathtt{En}$) in RCMT$_1$. Furthermore, the inclusion of Devanagari code-mixed ($\mathtt{Hi_{c}} \rightleftharpoons \mathtt{En}$) in RCMT$_2$ improves code-mixed performance; however, it does not provide additional support in the robustness of the system. Also, for $\mathtt{Hi_c} \rightarrow \mathtt{En}$, JAMT shows stronger results than the TFM model even when the Devanagari subwords are not shared with any other pair. We hypothesize that training on a common target En enables the encoder to learn overlapping representations for all inputs ($\mathtt{Hi_c}, \mathtt{Hi_{cr}}, \mathtt{Hi_{crn}}$), thereby reducing the effect of script variation and reinforcing the same family correlation.

Previous works in CMT have primarily relied on large-scale multilingual models such as mBART and mT5 (Xue et al., 2021; Liu et al., 2020b; Gautam et al., 2021; Jawahar et al., 2021). For comparison, we adopt the existing approach by finetuning mT5-small and mBART models on our CM datasets. Table 2 (row-3 and row-4) highlights the code-mixed performance on these finetuned models. Surprisingly, the romanized code-mixed MT ($\mathtt{c} + \mathtt{r}$) demonstrates comparable METEOR performance with +1.35% improvement over its Devanagari counterpart (c), even though the romanized Hindi text is seen only during finetuning. Conclusively from Table 2, these transfer learning approaches still lag behind JAMT, especially in robust CMT as the pre-trained procedure did not involve any kind of code-mixed data. However, it gives us a direction to explore by incorporating such CM data in the pre-training steps.

Robust MT Results: In order to corroborate the robustness capabilities of RCMT models, we test three categories of noise-robust MT models as baselines, namely MTT, MTNT, and AdvSR. Among these, MTT proves to be most resilient to synthetic noise with 1.21 BLEU decrease from RCMT$_1$ as it uses a dual decoding scheme to jointly maximize clean text and the translated text. Yet, this improvement comes at the cost of increased model size to allocate parameters for second decoder module. On the other hand, JAMT has the capability to adapt to any number of pairs without increasing the model size. The AdvSR model, trained exclusively on noisy corpus, yields better performance on CALIGN dataset than the MTNT model, which is trained on clean corpus $\mathtt{Hi_{cr}} \rightarrow \mathtt{En}$ and finetuned on the noisy corpus $\mathtt{Hi_{crn}} \rightarrow \mathtt{En}$. However, in CTRANS, AdvSR reports inferior performance against MTNT, possibly due to the on-the-fly segmentation method that is unable to handle lexical differences for similar code-mixed words in source sentences when translated in different ways depending on the context. In comparison, without changing the training procedure or scaling the parameters, JAMT achieves the best robustness to noise with an avg BLEU score of 10.89 against 9.68 of the best baseline (MTT).

Zero-shot MT Results: A good way to leverage the cross-lingual transfer property of multilingual models is to incorporate CM behaviour learned from one code-mixed language to an unseen code-mixed language. Table 3 shows the effectiveness of zero-shot CM translation ($\{\mathtt{Bn_c}, \mathtt{Bn_{cr}}\} \rightarrow \mathtt{En}$) by training a joint model using a bilingual Bn-En corpus and our synthetic code-mixed Hi-En corpus in the fol-

Model			Hi		Bn	
			B	M	B	M
CTRANS	MMT	c	10.8	41.9	**13.84**	45.1
		c + r	9.41	40.2	12.65	43.3
		c + r + n	5.50	29.3	-	-
	ZCMT	c	**11.95**	**43.4**	12.81	**45.5**
		c + r	11.45	42.5	11.96	44.0
		c + r + n	7.41	33.2	-	-
CALIGN	MMT	c	13.59	45.0	**15.66**	47.7
		c + r	13.05	44.1	13.83	44.3
		c + r + n	8.31	34.2	-	-
	ZCMT	c	**14.00**	**46.7**	15.41	**49.8**
		c + r	13.69	46.1	14.01	47.6
		c + r + n	10.79	40.4	-	-

Table 3: Performance of ZCMT model for Hindi (Hi), Bengali (Bn) to English translation on CTRANS and CALIGN dataset. c, r, n denote the code-mixed, romanized, noisy version of a dataset.

Source	$\mathrm{Hi}_{\mathrm{cr}}$	Is **thought** ko sabhi places par support nahin mila.
Target	En	The **concept** is not a universal hit.
CTRANS	En	This idea was not supported at all places.
CALIGN	En	This thought did not support at all the places.
Source	$\mathrm{Hi}_{\mathrm{cr}}$	Yah aapke relatives aur loved ones ke liye ek **complete** gift hai.
Target	En	It is **perfect** gift for your relatives and loved ones.
CTRANS	En	This is a whole gift for your relatives and loved ones
CALIGN	En	This is a complete gift for your relatives and loved ones

Table 4: Sample translation of code-mixed ($\mathrm{Hi}_{\mathrm{cr}}$) sentences to English (En) by translation (CTRANS) and alignment (CALIGN) of proposed RCMT_1 model.

lowing directions: $\{\mathrm{Hi}_{\mathrm{c}}, \mathrm{Hi}_{\mathrm{cr}}, \mathrm{Bn}, \mathrm{Bn}_{\mathrm{r}}\} \rightleftharpoons \mathrm{En} + \mathrm{Hi}_{\mathrm{crn}} \rightarrow \mathrm{En}$. For the baseline model, we test Bn code-mixed translation without training on CM text in a multilingual manner (MMT), i.e., $\{\mathrm{Hi}, \mathrm{Hi}_{\mathrm{r}}, \mathrm{Bn}, \mathrm{Bn}_{\mathrm{r}}\} \rightleftharpoons \mathrm{En} + \mathrm{Hi}_{\mathrm{rn}} \rightarrow \mathrm{En}$. Interestingly, MMT demonstrates appreciable performance on the Bn test set with ZCMT obtaining 3.25 improvement of METEOR scores over the MMT model. A possible reason for this can be the nature of the spoken tutorial test set, which mostly contains technical words and proper nouns as English (L_e) words in Bengali (L_m) code-mixed text.

Another surprising benefit of our ZCMT model is observed in Hindi CM translation in both Devanagari and romanized texts of CALIGN dataset outperforming RCMT_1 and RCMT_2 scores in Table 2. This indicates that adding languages from the same family (Indo-Aryan) can sometimes improve the code-mixed translation quality despite varying scripts (Devanagari vs. Eastern-Nagari).

Qualitative Analysis: Table 4 shows the difference in outputs of CALIGN and CTRANS datasets for the RCMT_1 model. JAMT trained on CALIGN learns to match the words in source and target – the word "*thought*" is translated as it is from the source sentence; whereas, in CTRANS, it gets mapped to a commonly used word "*idea*". Similar behaviour can be seen in the second example where the word "*complete*" takes a new meaning "*whole*" in the CTRANS prediction. Interestingly, the translations in both samples are semantically very different from the ideal target even when they represent a coherent and accurate translation. This highlights the shortcomings of precision-recall based metrics such as B, M, etc. A simple but correct translation would result in a low score when evaluated against a vocabulary-rich complex translation.

Human Evaluation: To quantitatively assess the quality of our synthetic CM sentences, we perform a human evaluation on 50 randomly selected Hinglish samples from CTRANS and CALIGN datasets. Three bilingual speakers proficient in English and Hindi were asked to rate the adequacy and fluency of each sample on a 5-point scale. Fluency measures whether the generated code-mixed sentence is syntactically fluent independent of its meaning, whereas adequacy compares if the meaning of the original Hi sentence is adequately conveyed in the target sentence. The annotators report the average adequacy score for CALIGN and CTRANS as 4.76 and 4.18, respectively. Moreover, they report 4.44 and 4.12 average fluency scores on the two datasets. The superiority of CALIGN over CTRANS in adequacy and fluency also aligns with better CMT results in Table 2. However, both methods are prone to errors, some of them are discussed in appendix.

6 Conclusion

In this work, we proposed a two-phase strategy to translate the real-world code-mixed sentences in multiple languages to English. First, a linguistically informed pipeline was introduced to generate a large-scale HINMIX code-mixed corpora synthetically. Next, we created a perturbed corpus by passing the clean code-mixed corpus to an adversarial module – both of which are simultaneously trained in a joint learning mechanism to learn robust CM representations. Finally, we showed the effectiveness of zero-shot learning on code-mixed MT in Bengali language. Our evaluation showed satisfying performance for both robust Hindi code-mixed and zero-shot Bengali code-mixed translation.

References

Ramakrishna Appicharla, Kamal Kumar Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2021. IITP-MT at CALCS2021: English to Hinglish neural machine translation using unsupervised synthetic code-mixed parallel corpus. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 31–35, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR, San Diego, CA, US.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1766, Melbourne, Australia. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 32:7059–7069.

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.

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling code-mixed translation: Parallel corpus creation and MT augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Suman Dowlagar and Radhika Mamidi. 2021. Gated convolutional sequence to sequence based learning for English-hingilsh code-switched machine translation. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 26–30, Online. Association for Computational Linguistics.

Luisa Duran. 1994. Toward a better understanding of code switching and interlanguage in bilinguality: Implications for bilingual instruction. *The journal of educational issues of language minority students*, 14(2):69–88.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855, Portorož, Slovenia. European Language Resources Association (ELRA).

Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. CoMeT: Towards code-mixed translation using parallel monolingual sentences. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 47–55, Online. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi. 2021. Training data augmentation for code-mixed translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5760–5766, Online. Association for Computational Linguistics.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online. Association for Computational Linguistics.

Ganesh Jawahar, El Moatez Billah Nagoudi, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2021. Exploring text-to-text transformers for English to Hinglish machine translation with synthetic code-mixing. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 36–46, Online. Association for Computational Linguistics.

Aravind K. Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, Hong Kong, China. Association for Computational Linguistics.

Judith F. Kroll, Susan C. Bobb, Maya Misra, and Taomei Guo. 2008. Language selection in bilingual speech: Evidence for inhibitory processes. *Acta Psychologica*, 128(3):416–430. Bilingualism: Functional and neural perspectives.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. 2020a. Very deep transformers for neural machine translation. *CoRR*, abs/2008.07772.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Carol Myers-Scotton. 1993. Common and uncommon ground: Social and structural factors in codeswitching. *Language in Society*, 22(4):475–503.

Carol Myers-Scotton. 1997. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

El Moatez Billah Nagoudi, AbdelRahim Elmadany, and Muhammad Abdul-Mageed. 2021. Investigating code-mixed Modern Standard Arabic-Egyptian to English machine translation. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 56–64, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Jungsoo Park, Mujeen Sung, Jinhyuk Lee, and Jaewoo Kang. 2020. Adversarial subword regularization for robust neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1945–1953, Online. Association for Computational Linguistics.

Peyman Passban, Puneeth S. M. Saladi, and Qun Liu. 2020. Revisiting robust neural machine translation: A transformer case study. *CoRR*, abs/2012.15710.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790, Barcelona (online). International Committee for Computational Linguistics.

Carol W. Pfaff. 1979. Constraints on language mixing: Intrasentential code-switching and borrowing in spanish/english. *Language*, 55(2):291–318.

Shana Poplack. 1978. *Syntactic structure and social function of code-switching*, volume 2. Centro de Estudios Puertorriqueños,[City University of New York].

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, Melbourne, Australia. Association for Computational Linguistics.

Mohd Sanad Zaki Rizvi, Anirudh Srinivasan, Tanuja Ganu, Monojit Choudhury, and Sunayana Sitaram. 2021. GCM: A toolkit for generating synthetic code-mixed text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 205–211, Online. Association for Computational Linguistics.

Motoki Sato, Jun Suzuki, and Shun Kiyono. 2019. Effective adversarial regularization for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 204–210, Florence, Italy. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rajendra Singh. 1985. Grammatical constraints on code-mixing: Evidence from hindi-english. *Canadian Journal of Linguistics/Revue canadienne de linguistique*, 30(1):33–45.

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W. Black. 2019. A survey of code-switched speech and language processing. *CoRR*, abs/1904.00784.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Los Alamitos, CA, USA. IEEE Computer Society.

Vaibhav Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. 2019. Improving robustness of machine translation with synthetic noise. 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 1916–1920, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Shivendra K Verma. 1976. Code-switching: Hindi-english. *Lingua*, 38(2):153–165.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Code-switching language modeling using syntax-aware multi-task learning. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 62–67, Melbourne, Australia. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Shuyan Zhou, Xiangkai Zeng, Yingqi Zhou, Antonios Anastasopoulos, and Graham Neubig. 2019. Improving robustness of neural machine translation with multi-task learning. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 565–571, Florence, Italy. Association for Computational Linguistics.

A Appendix

Candidate Word Selection: First, we select words to substitute in the Hindi (L_m) sentence based on their POS tag. Given a source sentence $S = \{s_1, s_2, \ldots, s_n\} \in L_m$ and a target sentence $T = \{t_1, t_2, \ldots, t_m\} \in L_e$, we obtain POS tags for each word in S. Next, we make the select candidate words based on their POS tags:

1. Named entities such as *person*, *location*, *organization*, etc., are represented as *proper nouns* (NNP, NNPC, NNPS). These are typically present in an ambiguous manner where the root word does not change, but multiple spelling variations can be found due to its modern adaptation. For example, "*sitambar*" vs "*september*", "*captaan*" vs "*captain*".
2. *Common nouns* (NN, NNC, NNS), *adjectives* (JJ), and *quantifiers* (QC, QCC, QO) are frequently translated with their L_e counterparts. These words do not change the grammatical structure of L_m and form the basis of widespread Hinglish usage.

Based on these switching constraints, we form an inclusion list (I) containing the POS tags to be included for code-switching. Subsequently, we shortlist the candidate words $S' = s_i$ such that their corresponding tags $p_i \in I$.

Heuristic for candidate word selection for language switching: Given that there can be $2^r - 1$ CM combinations in a sentence of r candidate words, we adopt the following selection rule depending upon the length of sentences to narrow down the possible sample space:

1. Use all combinations for r<=4. For example, an n-word sentence with 3 candidate words will have $2^3 - 1$=7 CM sentences.
2. Use $r - 3$ to r candidate word combinations for 5<=r<7. For example, an n-word sentence with 5 candidate words will have ${}^5C_2$+${}^5C_3$+${}^5C_4$+${}^5C_5$=26 CM sentences.
3. Use $0.6r$ to $0.7r$ candidate word combinations for r>=7. For example, an n-word sentence with 15 candidate words will have ${}^{15}C_9$+${}^{15}C_{10}$=8008 CM sentences.

Adversarial Module: The transliteration of non-roman languages depends upon the phonetic transcription of each word, varying heavily with the writer's interpretation of involved languages. With no consistent spelling of a word, it becomes crucial to simulate the real-world variations and noise for the practical application of any CMT model. Hence, we propose to learn robust contextual representations by distorting the available clean corpora with word-level perturbations as follows[4]:

- **Switch:** The adjacent characters inside the word are randomly switched to reproduce the typos due to the fast entry of keys. For example, "$t\ r\ a\ \underline{n}\ \underline{s}\ f\ e\ r$" vs "$t\ r\ a\ \underline{s}\ \underline{n}\ f\ e\ r$".
- **Omission:** A single character inside a word is randomly omitted to add noise. This error is usual when using short words during informal communication on OSNs. This also occurs in cases when characters are excluded while typing due to the phonetically similar pronunciation of the correct and incorrect spellings. For example, "$a\ m\ \underline{a}\ z\ \underline{i}\ n\ g$" vs "$a\ m\ z\ n\ g$".
- **Proximity typo:** While typing a character, a neighboring key is pressed mistakenly, thereby completely distorting the word. To replicate this error, we randomly select a character from the word followed by random neighboring key replacement corresponding to the QWERTY keyboard. For example, "$m\ o\ \underline{b}\ i\ l\ e$" vs "$m\ o\ \underline{v}\ i\ l\ e$".
- **Random Shuffle:** Sometimes, the non-adjacent letters are swapped erroneously. Although this does not happen frequently, we inject this noise by randomly shuffling the word to make our model robust to any word-level noise. For example, "$l\ \underline{a}\ p\ t\ \underline{o}\ p$" vs "$l\ \underline{o}\ p\ t\ \underline{a}\ p$"

We inject 30% switch, 12% omission, 12% typo, and 5% shuffle noise to $\texttt{Hi}_{\texttt{cr}}$ for producing a 60% word-level noisy code-mixed corpus $\texttt{Hi}_{\texttt{crn}}$-`En`. Both clean ($\texttt{Hi}_{\texttt{cr}}$-`En`) and noisy ($\texttt{Hi}_{\texttt{crn}}$-`En`) corpora are further used to train a joint model, which is described in the next subsection.

Statistics: The detailed statistics of the synthetic and gold-standard annotated code-mixed datasets are provided in Table 5. `CTRANS` on an average, contains 19% more number of ways in which a single Hindi sentence is represented into multiple CM sentences, calculated by the ratio of total sentences to unique sentences than `CALIGN`. The higher number of `Hi` (src) tokens in `CALIGN` is justified by the fact that the dataset has lower Code-Mixing Index (CMI) (27.9% vs 35.9%) than `CTRANS` suggesting a less percentage of code-mixing. Due to this, a relatively lesser number of words are substituted by their English counterparts. Despite a

[4] All noise is added between the first and last character of a word keeping both characters intact.

Statistics	CTRANS	CALIGN	Dev	Test
	Train			
#Total Sent	4.9M	4.2M	280	2507
#Unique Sent	0.67M	0.71M	280	2507
CMI	35.6	27.9	32.6	32.4
SPF	47.7	44.3	47	45.5
Token-level statistics				
#Hi (src)	0.19M	0.25M	711	4194
#En (src)	0.08M	0.11M	667	5923
#En (tgt)	0.17M	0.19M	1392	11255
#Total (src-tgt)	0.45M	0.52M	2533	18827
Char-level sentence length				
Mean	84.73	100.9	65.6	124.9
Median	74	88	64	111
Word-level sentence length				
Mean	15.7	18.24	12.17	22.8
Median	14	16	12	20

Table 5: Statistics of CTRANS and CALIGN code-mixed datasets. Here, src and tgt represent source ($\mathtt{Hi}_{\mathtt{c}}$) and target (En) sentences.

Source	$\mathtt{Hi}_{\mathtt{r}}$	Pati ki **prerana** se unhonne sanskrut men likhit ramayan ka bangla men **sankshipt** rupantar kiya.
Target	En	At her husband's **persuasion** she translated into Bengali an **abridged** version of the Ramayana from Sanskrit.
CTRANS	$\mathtt{Hi}_{\mathtt{cr}}$	Husband ki inspiration se unhonne sanskrit men written ramayana ka bangla men brief rupantar kiya.
CALIGN	$\mathtt{Hi}_{\mathtt{cr}}$	Husband ki persuasion se unhonne sanskrit men likhit ramayan ka bangla men abridged rupantar kiya.
Source	$\mathtt{Hi}_{\mathtt{r}}$	Hum khane ke baad **aam khate** the
Target	En	We **ate mangoes** after lunch
CTRANS	$\mathtt{Hi}_{\mathtt{cr}}$	Hum khane ke baad common account the
CALIGN	$\mathtt{Hi}_{\mathtt{cr}}$	Hum khane ke baad mangoes ate the

Table 6: Samples of generated code-mixed ($\mathtt{Hi}_{\mathtt{cr}}$) sentences using translation (CTRANS) and alignment (CALIGN) approaches.

lower CMI, we can see that CALIGN dataset contains as much as 30000 higher number of En(src) tokens than CTRANS as the alignment based substitution method replaces different words based on the target sentence alignment. Further, the CM sentences in the test set have longer average sentence length than the train set (34.5%↑ character-level and 34.3%↑ word-level), demonstrating the difficulty of code-mixed machine translation at test-time.

We also evaluate the complexity of datasets using codemix-specific metrics such as Code-Mixing Index (CMI) and Switch Point Fraction (SPF). CMI measures the percentage of code-mixing in a sentence, whereas SPF calculates the complexity of code-mixing in a sentence. On average, the CALIGN dataset is 7.1% less complex and has a 21.6% lower presence of code-mixed words than CTRANS making it relatively easier to translate.

Training details: We use a standard seq2seq Transformer model (Vaswani et al., 2017) in all our experiments to ensure the same number of parameters. Both encoder and decoder consist of a stack of 6 identical layers. Each layer comprises a Multi-Head Attention layer with 4 attention heads and a Feed-forward layer with an inner dimension of 1024. The shared input and output embedding dimensions are set to 512. We use a dropout rate of 0.1, a learning rate of 5×10^{-4} and an Adam optimizer with warmup steps of 4000. A unigram model with character coverage 1.0 is trained on all languages to obtain a common vocabulary of size 32000. To implement our model, the fairseq (Ott et al., 2019) toolkit is employed. We compute **SacreBLEU** (Ott et al., 2019), and **METEOR** (Banerjee and Lavie, 2005) to evaluate the quality of the translation.

Tokenization: We apply a language-free SentencePiece[5] tokenizer with a unigram subword model (Kudo, 2018) to generate a vocabulary directly from the raw text. As the unigram model calculates subwords according to the occurrence probabilities, directly applying the tokenization to the corpora would result in the underrepresentation of low-resource languages. Therefore, we undersample the high-resource language by randomly choosing a fixed set of sentences from the corpora to obtain the shared dictionary.

Qualitative Analysis of CTRANS and CALIGN We determine the quality of the synthetic code-mixed sentences in CTRANS and CALIGN as well the generated translations using JAMT. In Table 6, samples from both datasets highlight the distinction between our two CM generation approaches. In the translation approach, the word "*prerana*" is replaced by "*inspiration*" due to its frequent usage in the corpus as well as the real world. But due to the existence of a relatively uncommon word "*persuasion*" in its target pair, the CALIGN dataset chooses "*persuasion*" for substitution. Similarly, "*sankshipt*" is replaced by "*brief*" in CTRANS and by a rare word "*abridged*" in CALIGN. This makes our CTRANS code-mixed vocabulary consistent throughout every occurrence of a source word, whereas CALIGN benefits from the rich lexicons in generated CM sentences.

[5] https://github.com/google/sentencepiece

Error Analysis: We end with the analysis of some common errors when translating CM text to English.

- **Alignment Errors**: Despite the context-dependent word substitution in CALIGN, this approach is susceptible to all the alignment errors. Incorrect word mapping between the source-target could completely alter its CM meaning. Also, we substitute words with an only one-to-one correspondence between the source and target, thereby abandoning all words with multiple alignment mapping.
- **Translation Errors**: The benefit of imitating real-world code-mixed usage by substitution with prevalent words (learned from translation model) leads to incorrect handling of Homonyms (Anekarthi Shabd). An individual word, when passed through a translation model, gives a single translation independent of context. This leads to incorrect translation in scenarios when the same word represents a different meaning. For instance, in Table 6, the word "*aam*" in Hi incorrectly translates to "*common*" where the correct translation would be "*mango*" according to the context.
- **POS Tagging Errors**: A good POS tagger forms the basis of our code-mixed creation process. In cases when a word in the source sentence is incorrectly tagged to a tag in POS inclusion list I, it will be replaced by both substitution approaches. For example in Table 6, the verb "*khate*" gets mistagged to a noun, thereby being replaced by its translation "*account*" in CTRANS and "*ate*" in CALIGN. Note that the word "*khate*" is a homonym, thereby producing both translation and POS-tagging error in a single word.