Code-Switched Text Synthesis in Unseen Language Pairs

I-Hung Hsu[1]* **Avik Ray**[2] **Shubham Grag**[2] **Nanyun Peng**[2,3] **Jing Huang**[2]
[1] Information Science Institute, University of Southern California
[2] Amazon Alexa AI [3] University of California, Los Angeles
ihunghsu@usc.edu {avikray, gargshu}@amazon.com
violetpeng@cs.ucla.edu jinghuang.zhu@gmail.com

Abstract

Existing efforts on text synthesis for code-switching mostly require training on code-switched texts in the target language pairs, limiting the deployment of the models to cases lacking code-switched data. In this work, we study the problem of synthesizing code-switched texts for language pairs absent from the training data. We introduce GLOSS, a model built on top of a pre-trained multilingual machine translation model (PMMTM) with an additional code-switching module. This module, either an adapter or extra prefixes, learns code-switching patterns from code-switched data during training, while the primary component of GLOSS, i.e., the PMMTM, is frozen. The design of only adjusting the code-switching module prevents our model from overfitting to the constrained training data for code-switching. Hence, GLOSS exhibits the ability to generalize and synthesize code-switched texts across a broader spectrum of language pairs. Additionally, we develop a self-training algorithm on target language pairs further to enhance the reliability of GLOSS. Automatic evaluations on four language pairs show that GLOSS achieves at least 55% relative BLEU and METEOR scores improvements compared to strong baselines. Human evaluations on two language pairs further validate the success of GLOSS.

1 Introduction

Code-switching, the linguistic phenomenon of using more than one language within a single utterance or conversation,[1] is a common expression of multilingualism in informal text and speech (Auer and Wei, 2008; Gumperz, 1982). To accommodate the needs of multicultural and multilingual societies and individuals, there is a growing interest in investigating models dedicated to code-switching within the realm of conversational AI (FitzGerald et al., 2022; Khanuja et al., 2020; Winata et al., 2022; Sitaram et al., 2019). However, a notable obstacle in code-switching modeling is the scarcity of large-scale code-switched text datasets for different applications in diverse language pairs (Gupta et al., 2020; Tarunesh et al., 2021). This necessitates generative models capable of synthesizing code-switched texts, facilitating subsequent studies for code-switching.

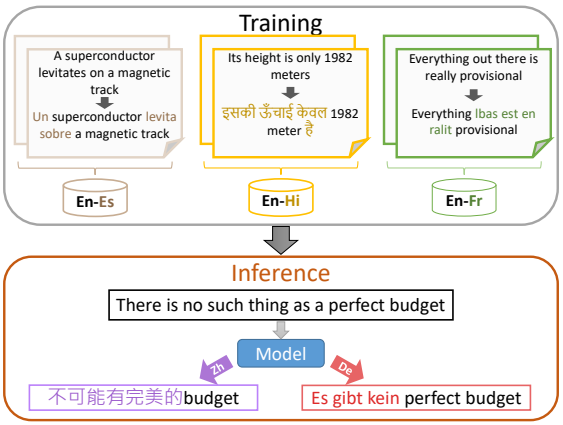

Figure 1: An illustration of our problem setting. Given a sentence in arbitrary languages (English *(En)* in this figure) and a designated language (Chinese *(Zh)* and German *(De)*, in the figure), the model needs to synthesize a corresponding code-switched sentence that mixes *the original language* and *the designated language*. Additionally, we allow the designated language selections to differ from examples seen during training.

Most prior work on text synthesis for code-switching assumes the availability of training data for all language pairs being tested. Early trials concentrate on individual language pair (Samanta et al., 2019; Chang et al., 2019; Tarunesh et al., 2021). For example, Bhat et al. (2016) develop a code-switched text synthesizer for Hindi-English based on linguistic rules (Poplack, 1980; Belazi et al., 1994; Myers-Scotton, 1997), while Lee et al.

*Work was done when the author interned at Amazon.
[1]In this paper, we mainly focus on the sentence-level code-switching involving only two languages.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5137–5151
July 9-14, 2023

(2019); Winata et al. (2019); Garg et al. (2018) explore neural generative models for Chinese-English code-switched text synthesis. More recently, Gupta et al. (2020) presents pioneering efforts in developing a generic method for producing high-quality and fluent code-switched sentences across diverse language pairs. This is achieved through the collection of code-switched texts in multiple languages.

However, the requirement of training on code-switched texts for target language pairs hinders the scalability of existing models to cover a broader range of language pairs. Many real-world code-switching scenarios, such as Swahili-English in Tanzania (Kanijo, 2018), Shona-English in Zimbabwe (Mashiri, 2002) suffer from limited or non-existent curated datasets. Recognizing this resource limitation, in this work, our study focuses on synthesizing code-switched text in multiple language pairs, including those language pairs that are *unseen* during training (*zero-shot transfer* setting (Huang et al., 2021, 2022)). In this setting, models must learn code-switched patterns from limited code-switched training data in some language pairs and generalize to other language pairs, as shown in Fig. 1. The setting enables a more flexible process of code-switched text synthesis by using existing resources to assist resource-limited language pairs. Yet, it also introduces new challenges: (1) models must possess the ability to generate tokens in multiple languages; (2) models need to acquire a transferable ability for code-switching such that they can generate code-switched text in unseen language pairs.

To overcome the challenges, we propose GLOSS, a **G**enera**L**ized c**O**de-**S**witched text **S**ynthesizer that introduces an additional code-switching module to a pre-trained multilingual machine translation model (PMMTM). The code-switching module, implemented either through an adapter (Houlsby et al., 2019) or extra prefixes (Li and Liang, 2021), offers a parameter-efficient approach to transfer learning from machine translation to code-switched text synthesis. Inheriting the ability of PMMTM, GLOSS can generate text across multiple languages. The incorporation of an additional code-switching module, instead of directly fine-tuning the PMMTM, serves as an effective method to prevent models from overfitting to the specific training code-switched language pairs.

Furthermore, we develop a self-training algorithm on the target language pairs to improve GLOSS further. Specifically, our preliminary study shows that although GLOSS can successfully generate reasonable code-switched sentences, when performing zero-shot transfer to unseen language pairs, it may still generate non-code-switched sentences (around 11% to 13% of cases). The proposed self-training framework aims to introduce weakly-supervised signals to help GLOSS more stably generate target domain cases when the target language pair is known.[2] To achieve this, we iteratively fine-tune GLOSS on a *filtered* dataset that is generated by GLOSS itself in the target domain case. The filter incorporates a language identification model to remove low-quality instances.[3] Being fine-tuned on filtered data, GLOSS learns to generate texts that satisfy the filtering rules and become more stable.

Our contribution is three-fold. First, we present GLOSS, a code-switched text synthesizer that can generate code-switched sentences across multiple language pairs, even those not in the training data. To the best of our knowledge, we are the first to study this setting. Second, we introduce a self-training framework to further improve GLOSS under the setting where the target language pair is known. Third, extensive experiments, including automatic evaluations on four languages and human evaluations on two languages, showcase GLOSS's strong performance. GLOSS achieves at least 55% relative BLEU and METEOR score improvements compared to strong baselines.

2 Problem Formulation

Our goal is to synthesize code-switched (CS) texts for language pairs where their CS examples are *never* provided during training.

Given a monolingual input sentence $\mathbf{x}^e$ in language l^e and an assigned language l^m ($l^m \neq l^e$), we aim to generate a sentence $\mathbf{x}^{m,e}$ that mixes l^m and l^e, while preserving the semantic meaning of $\mathbf{x}^e$.[4] We consider the setting where the assigned language l^m in the testing time is different from those in the training time. More formally, as illustrated in Figure 1, the training set consists of N language pairs $(l^{e_n}, l^{m_n})(n \in \{1, 2, ..., N\})$, while

[2] For example, we know the target scenario is to synthesize Bengali-English code-switched text, despite no Bengali-English code-switched training data being available.

[3] The language identification model is trained without using any code-switched data. More details are given in §3.3.

[4] Following the matrix language frame theory (Myers-Scotton, 1997; Joshi, 1982), l^m is called the matrix language and l^e is the embedded language.

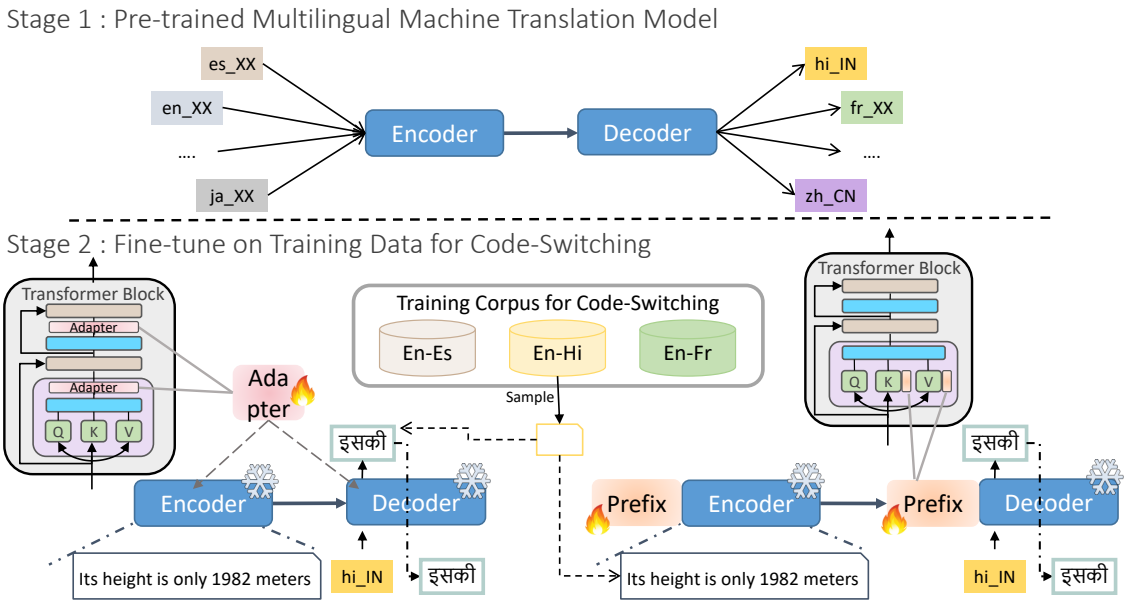

Figure 2: An overview of our GLOSS model. GLOSS is built on top of a pre-trained multilingual machine translation model, which is trained using machine translation data in many different language pairs. After the pre-trained multilingual machine translation model is prepared, we augment an adapter or extra prefixes to the model. The adapter or prefixes are trained using code-switched data while the pre-trained multilingual machine translation model's parameter will be frozen during the fine-tuning.

the testing set includes target language pairs where $l^{m_t} \notin \{l^{m_1}, ..., l^{m_N}\}, \forall t$. This scenario reflects real-world situations where code-switched data is more readily available for certain language pairs, such as Spanish-English and Hindi-English, while it is less accessible for others, such as Bengali-English and Swahili-English.

3 Method

We introduce GLOSS, a **G**ener**a**L**i**zed c**O**de-**S**witched text **S**ynthesizer that tackles the two specific challenges raised by our problem setting: (1) the model needs to generate texts across many languages, some are not even in the CS training data; (2) the model needs to learn transferable CS ability such that they generate reasonable CS sentences in unseen language pairs. Fig. 2 provides an overview.

To address the first challenge, we begin by obtaining a Pre-trained Multilingual Machine Translation Model (PMMTM) using multilingual machine translation data, which covers all languages that would be used for final CS text synthesis (§3.1).[5] The remaining challenge is how to make PMMTM a code-switched text synthesizer with only limited language coverage of training data.

[5]Here, we assume that machine translation data is more available, which is often the case in practice.

We propose to augment an additional code-switching module onto PMMTM, thereby creating GLOSS (§3.2). This additional code-switching module is trained on our limited CS data while keeping PMMTM parameters fixed. Instead of fine-tuning the entire PMMTM, this modularized design improves systematic generalization (Bahdanau et al., 2019; Ruis and Lake, 2022), where PMMTM focuses on generating translated sentences and the code-switching module concentrates on *"mixing"* languages. This approach allows GLOSS to be more adaptable and less prone to overfitting during the fine-tuning process on CS data.

Finally, we present a self-training framework that enables GLOSS to more stably generate CS texts in target language pairs (§3.3).

3.1 PMMTM

Multilingual machine translation models (Ha et al., 2016; Johnson et al., 2017; Baziotis et al., 2022; Tang et al., 2020) enable simple deployment and parameter-efficient support of machine translation for a large number of language pairs by using a shared representation space. To train a PMMTM, we follow the strategy of mBART-50 (Tang et al., 2020) to notify the model of the source language and the target language to be translated into. Specifically, a language-specific special token is

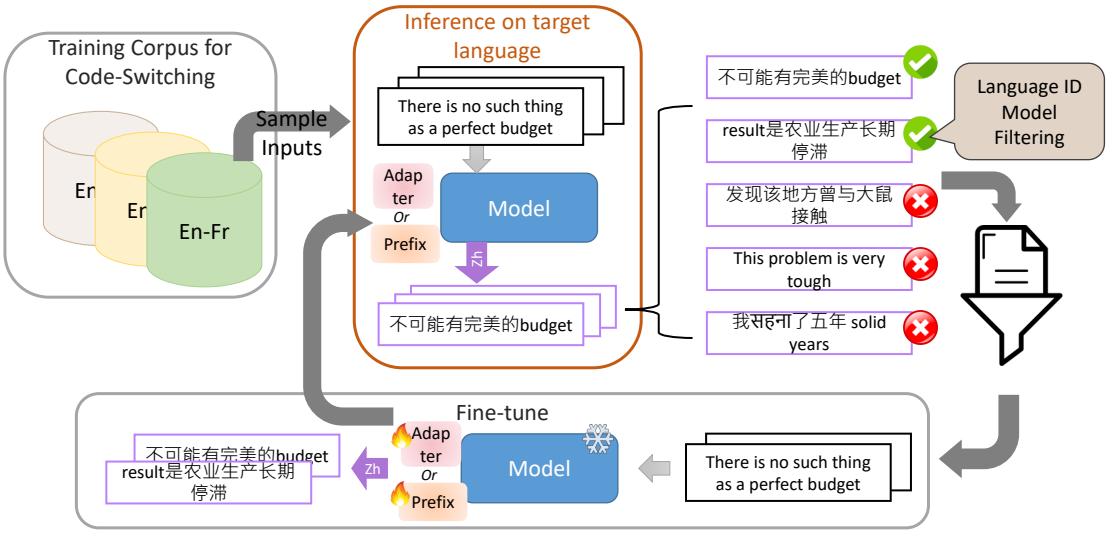

Figure 3: An illustration of the self-training procedure we designed for GLOSS. We use Chinese-English to be the target language pair as an example in this figure.

prepended to both the source and target sentences. Hence, during decoding, the first token fed to the decoder is the target language's special token that guides the translation. This is illustrated in Fig. 2.

3.2 The GLOSS Model

After obtaining a PMMTM, which can comprehend and generate phrases across multiple languages, our next step is to transform a PMMTM into a CS text synthesizer. A commonly used way is to directly fine-tune the PMMTM on CS training data (Tarunesh et al., 2021; Gupta et al., 2020). However, models directly fine-tuned on new data could easily overfit to the fine-tuning scenario. Thus it is hard to adapt the ability to perform code-switching to unseen language pairs. Therefore, instead of directly fine-tuning the whole PMMTM, we propose to use an *additional* code-switching module paired with the PMMTM. The module is specifically learned to *mix* languages for a given translation pair generated by PMMTM.

To implement the design and enable end-to-end training, we employ either an *adapter* (Houlsby et al., 2019) or extra *prefixes* (Li and Liang, 2021) as the code-switching module. These approaches are parameter-efficient methods to introduce control into pre-trained models and guide the final generation (He et al., 2022):

Adapter is an additional layer (and parameters) that is introduced inside each Transformer block (Vaswani et al., 2017), and it was shown to be an effective way to conduct transfer learning for NLP tasks (Houlsby et al., 2019). This layer is appended after each feed-forward layer (in a Transformer block). It projects the original feature size to a smaller dimension and then projects them back to the original size, ensuring that the number of parameters stays substantially small.

Prefix is another parameter-efficient way to conduct transfer learning for NLP tasks (Li and Liang, 2021). *"Prefix"* are the new key and value matrices used when calculating attention in Transformer. More specifically, trainable prefixes are a set of vectors that will be concatenated with the original key and value matrices when calculating dot-product attention. Hence, in each layer, inputs will be influenced by these additional keys and values after attention is applied.

During fine-tuning using CS training data, we keep the parameters of PMMTM frozen and solely train the adapter or prefixes. This allows the code-switching module to learn how to blend a *translated* distribution with the input sentence. When GLOSS is tested and tasked with generating a code-switched sentence in an unseen target language pair, the frozen PMMTM, having been trained to produce translations for this specific pair, can still generate reliable translations. With reliable translations, our code-switching module continues to perform a similar function during training by blending languages. As a result, GLOSS exhibits improved generalization capabilities.

3.3 GLOSS with Self-Training

Although GLOSS has the ability to generalize to synthesize CS text to languages that the PMMTM

supports, the generation could still be unstable. As we will show in §5, GLOSS still has around 11% to 13% of cases that will generate non-CS sentences when performing zero-shot transfer to unseen language pairs. Hence, we aim to improve this stability issue if more information about the test case is provided. We assume a common scenario in real practice — the target language pair l^m and l^e is known, and we can update GLOSS for fitting this specific target language pair.

We design a self-training procedure to incorporate off-the-shelf language identification models to help GLOSS synthesize target CS sentences more stably. The procedure is illustrated in Fig. 3. To be more specific, we first use the input sentence written in l^e in the CS training data as the input query and ask GLOSS to make a prediction on the target language l^m, forming potential CS sentences $\mathbf{x}^{m,e}$. Then, we use language identification models to perform sentence filtering based on the following constraints:

- The synthesized sentence should at least cover one token from l^m.
- The synthesized sentence should at least cover tokens from l^e.
- The synthesized sentence cannot cover tokens from other languages except l^m and l^e.

We use CLD3 [6] as the language identification model, which extracts character n-grams from the input text and computes an embedding based on the fraction of times each n-gram character appears. Notably, CLD3's training does not rely on code-switched text. We leverage CLD3's predicted language distribution for each token to determine if each generated sentence meets the aforementioned constraints. We filter out low-quality instances and collect the remaining sentences as a synthetic code-switching corpus specific to the target domain. This corpus is subsequently used for further fine-tuning of GLOSS. The procedure can be executed repeatedly in R rounds, where R is a hyper-parameter. Notice that other advanced filtering can be easily included in our proposed procedure and we leave the exploration as a future work.

Different from the classic self-training algorithm in semi-supervised research (Fei et al., 2023), in our procedure, the initial model is a zero-shot transfer model. Additionally, we apply a filtering process to further improve the quality of the synthetic code-switching corpus.

[6] www.github.com/bsolomon1124/pycld3

3.4 Discussion

Utilizing pre-trained models that are initially trained on machine translation data as a foundation for constructing code-switched (CS) text synthesizers has gained significant attention recently due to the resemblance between machine translation and CS text synthesis tasks (Tarunesh et al., 2021; Gupta et al., 2020). However, our work differs from theirs in that we train a *single* model capable of consuming all the machine translation data, thereby supporting translation across multiple language pairs. In contrast, prior works rely on selecting data based on the target language pair (l^m and l^e) as a priori.

Our approach enables a unified model that possesses the ability to generate phrases in multiple languages, thereby facilitating CS text synthesis across various language pairs. Conversely, constraining the training of the PMMTM to a limited number of languages, such as a few specific pairs, would result in GLOSS losing its ability to generalize to a broader range of CS language pairs.

4 Automatic Evaluation

4.1 Experimental Settings

Dataset and Evaluation Metrics. We use the data provided by Gupta et al. (2020), which covers eight language pairs, including Bengali-English (Bn-En), German-English (De-En), Spanish (Es-En), French-English (Fr-En), Hindi-English (Hi-En), Malayalam-English (Ml-En), Tamil-English (Ta-En), and Telugu-English (Te-En). Note that in this dataset, the input language sentence is always English. Hence, the target code-switched (CS) language pair is X-English, where X is the different languages that the dataset covers. In the original paper, they used English-X to call the language pair in their dataset, but we changed the naming to present the dominant language first. The dataset statistics are listed in Appendix §A.

In our setting, we conduct leave-one-out experiments, i.e., seven CS language pairs are selected as the CS training data, and the remaining is the test language pair. We select Bn-En, De-En, Es-En, and Hi-En as the four test scenarios based on the language resource levels defined in Tang et al. (2020), such that our selection covers high-resource (German, Spanish), medium-resource (Hindi), and low-resource (Bengali) languages. We evaluate the synthesized text using BLEU (Papineni et al., 2002)

	Model	Type	Bn-En		De-En		Es-En		Hi-En	
			B	M	B	M	B	M	B	M
UB.	Gupta et al. (2020)*	Sup.	21.49	27.32	24.15	30.47	22.47	29.45	21.55	28.37
	Fine-tuned PMMTM on all language pairs (mBART50-MMT)	Sup.	12.49	38.67	32.24	59.75	37.82	62.54	27.93	54.81
	Fine-tuned PMMTM on all language pairs (augment-MMT)	Sup.	13.08	38.69	32.65	59.96	38.59	63.36	28.88	55.10
	Copy Input	Unsup.	2.66	19.28	3.29	22.76	3.28	22.31	5.22	24.20
	Machine Translation	Unsup.	4.78	16.82	6.30	30.28	9.63	32.97	9.87	24.26
	Translate, Align, then Swap	Unsup.	1.91	16.06	5.53	27.30	7.80	30.11	6.61	24.90
	Fine-tuned PMMTM on available language pairs	Zst.	3.05	18.57	9.09	32.34	8.77	30.41	3.93	22.22
Proposed.	GLOSS (mBART50-MMT + adapter)	Zst.	2.31	22.07	18.63	48.28	23.04	49.75	4.09	22.02
	GLOSS (mBART50-MMT + prefix)	Zst.	5.21	26.83	20.49	48.49	23.47	50.52	7.51	29.82
	GLOSS (augment-MMT + adapter)	Zst.	2.16	18.60	14.58	40.75	16.62	42.31	8.61	30.39
	GLOSS (augment-MMT + prefix)	Zst.	**9.65**	**32.63**	**21.88**	**50.33**	**24.85**	**51.88**	**12.16**	**36.94**

Table 1: Automatic evaluation results for GLOSS. We evaluate the result in BLEU (B) and METEOR (M). We classify the models into three types — unsupervised baselines (Unsup.), supervised baselines (Sup.), and zero-shot transfer baselines (Zst.). The training of supervised baselines contains CS data in target language pairs and hence it can be viewed as an upper bound (UB.) for GLOSS. Numbers in bold are the best performance among all zero-shot transfer models and unsupervised models. *We report the numbers from the original paper.

and METEOR (Banerjee and Lavie, 2005) scores following Gupta et al. (2020).

Implementation Details. We use two different PMMTM for GLOSS. The first one directly adapts the pre-trained mBART50-many-to-many-MMT model (**mBART50-MMT**) from (Tang et al., 2020), which is a machine translation model trained on 50 language pairs using the ML50 benchmark. The other one is to further fine-tune mBART50-MMT on the machine translation data collected by Gupta et al. (2020) to make an "augmented mBART50-MMT" (**augment-MMT**). The second setting is considered since machine translation data in the ML50 benchmark are limited for Indic languages. Hence, we further fine-tune mBART50-MMT on the machine translation data provided in (Gupta et al., 2020) for three epochs. Notice that the machine translation data in (Gupta et al., 2020) only covers eight language pairs, making augment-MMT a more restricted machine translation model in terms of supported languages.

All GLOSS (mBART50-MMT/augment-MMT paired with adapter/prefix) are implemented using the Huggingface package (Wolf et al., 2020) as the backbone. To implement the adapter and prefix, we leverage AdatperHub (Pfeiffer et al., 2020). We use their default setting to set prefix length as 30 and use all prefixes in the self-attention block in the Transformer encoder, and cross-attention block as well as the self-attention block in the Transformer decoder. We train GLOSS with a machine equipped with 4 NVIDIA Tesla V100 GPUs. We train GLOSS using 1 GPU at a time with around 30 hrs of training.

We consider AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate set to 10^{-5} and the weight decay set to 10^{-5}. We set the batch size to 12 and the number of training epochs to 15. For GLOSS with self-training, we experiment with $R \in \{1, 2, 5\}$ rounds with heuristics. Hyper-parameter determination, except for R, is based on the available CS data in the development set without considering the leave-out language pair. Due to the computational resource restriction, our experiment results from a single seed. We note the gradual performance improvement as R increased in §4.3. However, determining the optimal stopping point for R presented a challenge since no development data exist under the zero-shot scenario. As a result, we decide not to increase R further in our experiments.

Compared baselines. Three types of baselines are considered:

- Unsupervised baselines — (1) **Copy Input**: directly copy the input sentence as the prediction, (2) **Machine Translation**: augment-MMT's machine translation results, (3) **Translate, Align, then Swap**: we use advanced unsupervised word-alignment tool (Dou and Neubig, 2021) to extract potential word alignment between the input sentence and the Machine Translation's prediction. Then, we generate the final output by having a probability p to swap words in Machine Translation's prediction with the aligned input word, where $p = 0.35$ is based on the statistics from the training data.

Model	Bn-En		De-En		Es-En		Hi-En	
	B	M	B	M	B	M	B	M
GLOSS (mBART50-MMT + prefix)	5.21	26.83	20.49	48.49	23.47	50.52	7.51	29.82
GLOSS (mBART50-MMT + prefix) + self-training($R = 1$)	5.84	28.31	20.55	48.83	24.00	51.12	8.22	31.55
GLOSS (mBART50-MMT + prefix) + self-training($R = 2$)	**6.66**	29.00	20.97	49.12	24.12	51.47	9.27	33.16
GLOSS (mBART50-MMT + prefix) + self-training($R = 5$)	6.26	**29.75**	**21.49**	**49.71**	**24.58**	**51.53**	**10.31**	**35.84**
GLOSS (augment-MMT + prefix)	9.65	32.63	21.88	50.33	24.85	51.88	12.16	36.94
GLOSS (augment-MMT + prefix) + self-training($R = 1$)	9.80	33.63	21.78	49.73	25.96	52.68	12.99	38.59
GLOSS (augment-MMT + prefix) + self-training($R = 2$)	10.19	34.70	22.36	50.59	26.22	52.88	**13.70**	**40.09**
GLOSS (augment-MMT + prefix) + self-training($R = 5$)	**10.32**	**35.46**	**22.45**	**50.63**	**26.31**	**53.13**	13.63	40.05

Table 2: Automatic evaluation results for GLOSS paired with our self-training procedure. We evaluate the result in BLEU (B) and METEOR (M). Numbers in bold are the best performance among models using the same architecture. We can observe the gradual improvement when more rounds of self-training are applied to GLOSS.

- Supervised baselines — (1) **Gupta et al. (2020)**: a sequence-to-sequence model that leverages XLM (Conneau and Lample, 2019) features and utilizes the transfer learning signal from machine translation to warm-up the model, (2) **Fine-tuned PMMTM on all language pairs**: we fine-tune mBART50-MMT on CS data in *all* eight language pairs.
- Zero-shot transfer baselines — (1) **Fine-tuned PMMTM on available language pairs**: fine-tune whole mBART50-MMT on *available* CS training data only (excluding test language pair).

Note that the training of supervised baselines contains CS data in target language pairs; hence, it can be viewed as an upper bound for GLOSS. Zero-shot transfer baselines are trained only using CS data from other language pairs but not the target language pair. Unsupervised baseline training does not use any CS training data.

4.2 Main Results

Tab. 1 shows the results. From the table, we can observe that the unsupervised baselines generate very unreliable CS sentences in general. Additionally, naively fine-tuning the whole PMMTM could perform even worse than the unsupervised methods. GLOSS improves unsupervised baselines and zero-shot transfer baselines by at least 55% relative scores across the board, and every variation of GLOSS could outperform these baselines. By comparing different variations of GLOSS, we can observe that GLOSS with prefixes is more robust than using an adapter, especially in the cases where the PMMTM model has worse performance (Bengali & Hindi due to limited training machine translation data used in mBART50-MMT). Furthermore, by comparing GLOSS equipped with augment-MMT and GLOSS equipped with mBART50-MMT, we highlight the PMMTM's impact on our model.

4.3 Results Given Known Target Language

When the target language pair is known, we can then apply our self-training procedure to GLOSS. We experiment on GLOSS using prefixes and present results in Tab. 2. From the table, we can observe the consistent improvement when adopting self-training to GLOSS, and the improvement is especially significant for Hindi-English. Additionally, by conducting self-training with more rounds, we can observe the gradual improvements in both of the cases for GLOSS with mBART50-MMT and augment-MMT.

5 Human Evaluation

To further verify the quality of our method, we conduct the human evaluation for Hindi-English and Chinese-English code-switched (CS) text using sentences in English as the source language.

5.1 Evaluator Selection

Considering the expertise of the annotation task requires people familiar with both English and Chinese (or English and Hindi), we have a high-standard selection process to recruit 3 professionals for the human evaluation. For Hindi-English annotation, We engaged the services of a team of expert professionals who were contracted to provide labels for various Hindi and English-related tasks. They're all native Hindi speakers and highly skilled in speaking Hindi-English code-switching. Conversely, our Chinese-English annotators are native Chinese NLP researchers with over three

	Model	Type	Hi-En				Zh-En			
			CS Rate.	F	S	Geo. Mean	CS Rate.	F	S	Geo. Mean
	Translate, Align, then Swap	Unsup.	98.6%	2.69	2.83	2.75	91.3%	3.19	3.62	3.33
	Fine-tuned PMMTM on available language pairs	Zst.	4.0%	1.00	1.00	1.00	2.0%	1.0	1.0	1.0
Ours.	GLOSS (prefix)	Zst.	87.3%	3.06	3.10	3.08	89.3%	3.67	3.84	3.73
Ours.	GLOSS (prefix + self-training)	Zst.	93.3%	3.84	3.96	3.90	99.3%	3.73	4.01	3.85
UB.	Fine-tuned PMMTM on all language pairs	Sup.	98.0%	4.09	4.21	4.15	−−	−−	−−	−−
UB.	Ground truth	−−	96.0%	4.40	4.39	4.84	94.0%	4.18	4.42	4.28

Table 3: Human evaluation results for GLOSS in Hindi-English (Hi-En) and Chinese-English (Zh-En). Code-switching correctness rate (*CS Rate.*) measures the percentage of the prediction is a correct CS. *F* is the abbreviation of the Fluency score. *S* is the abbreviation of the Semantic Correctness score. Geometric Mean (*Geo. Mean*) is the average of each sample's geometric mean between its code-switching correctness score, fluency and semantic scores. Results for the supervised baseline and the ground truth are presented as an upper bound (UB.) for reference.

years of experience, residing in the US for at least four years, and proficient in Chinese, English, and Chinese-English code-switching. We offer a competitive hourly payment that meets regional legal standards, though it's difficult to determine the average payment for them on this single task.

5.2 Experimental Settings

Dataset. To avoid the evaluation being biased in the domain we trained on, we collect testing English instances by selecting sentences from the following CS dataset. We sample 50 sentences for each language pair.

- Hindi-English: We use the data released from Tarunesh et al. (2021), who collected the dataset via crowd-sourcing in India. Every data point in this dataset is a pair of an English sentence and its corresponding Hindi-English CS sentence.
- Chinese-English: We use the transcript data of the SEAME dataset (Lyu et al., 2010), which is a Chinese-English CS speech recognition dataset. To get the English counterpart of the Chinese-English sentences, we ask experts to translate the CS sentence back to their English version.

Compared models. We compare six methods: (1) **Translate, Align, then Swap**, which serves as a representative of unsupervised methods, (2) **Fine-tuned PMMTM on available language pairs**, which serves as a baseline for zero-shot transfer, (3) **GLOSS + prefix**, we use augment-MMT as the backbone for Hindi-English, while using mBART50-MMT as the base model for Chinese-English, (4) **GLOSS + prefix + self-training**, we apply self-training ($R = 5$) to GLOSS + prefix, (5) **Fine-tuned PMMTM on all language pairs**, which serves a strong supervised baseline. Notice that since the training dataset in Gupta et al. (2020) does not contain the Chinese-English pair. Hence, when evaluating on Chinese-English, this baseline is not applicable, (6) **Ground truth**, the original CS sentences we sampled from the dataset.

Evaluation Procedure We ask each expert annotator to evaluate all the output of 50 testing instances from all models (i.e., 300 sentences for Hindi-English and 250 for Chinese-English). Our questionnaire covers the following three questions when using Hindi-English as an example.

- Code-switching Correctness: We measure whether the present sentence is correct CS (binary score). Specifically, we define a sentence as a correct CS sentence if it satisfies the constraints: (a) It's not fully Hindi or English, (b) It should be mainly in Hindi, and (c) There's no other language except English and Hindi.
- Fluency: Measuring the fluency of the prediction presented to humans with scores from 1 to 5, with 5 as the best.
- Semantic Correctness: Measuring whether the predicted sentence correctly reveals the meaning of the corresponding input sentence with scores from 1 to 5, with 5 as a fully correct translation.

5.3 Results

Tab. 3 presents the results. First, we can observe that the code-switching correctness rate is extremely low for the zero-shot baseline — Fine-tuned PMMTM on available language pairs. Second, although the unsupervised baseline – Translate, Align, then Swap gets a high code-switching success rate, the low fluency reveals that deciding a suitable position to switch languages is a task beyond random. Third, we can observe that self-training can successfully improve the code-switching quality across all metrics in both lan-

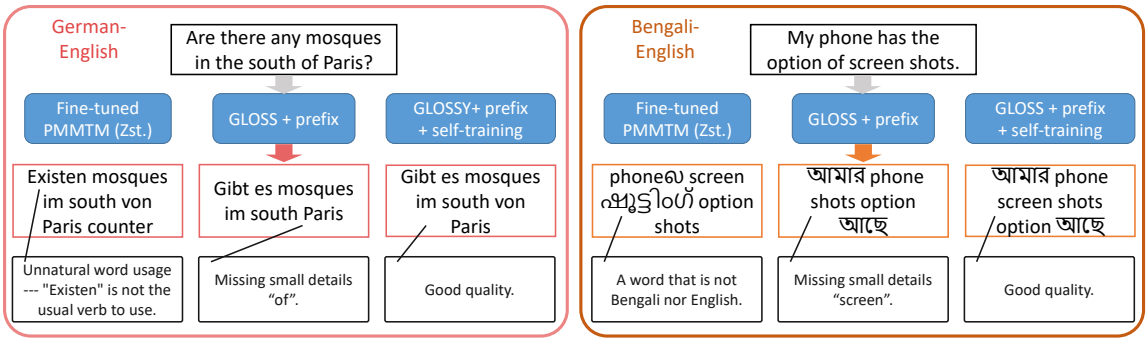

Figure 4: Real examples generated by the models for German-English and Bengali-English cases. Fine-tuned PMMTM (Zst.) refers to the *Fine-tuned PMMTM on available language pairs* method. The explanation for each prediction is presented in the bottom boxes in the Figure.

guages, indicating the method's effectiveness.

5.4 Output Examples

Lastly, we present real examples generated by our models in Fig. 4. For these examples, we can see that directly fine-tuning the whole PMMTM on CS training data will generate unnatural or even predictions containing tokens in other languages. In contrast, GLOSS can generate more stable results, and our self-training algorithm can even help GLOSS to generate high-quality CS sentences.

6 Related Work

Early approaches (Pratapa et al., 2018; Bhat et al., 2016; Pratapa and Choudhury, 2021; Li and Fung, 2014) on code-switched (CS) text synthesis were built based on various linguistic theories, such functional head constraints (Belazi et al., 1994), Matrix-Language theory (Myers-Scotton, 1997; Joshi, 1982), and Equivalence-Constraint theory (Poplack, 1980; Sankoff, 1998). To turn linguistic theories into computational models, Bhat et al. (2016); Pratapa and Choudhury (2021) leverage trained constituency parser to extract parses of translation pairs and create CS sentences by mixing translation pairs following the syntactic constraints derived from the theories. However, constraints cannot be postulated as a universal rule for all CS scenarios, especially for languages that are syntactically divergent (Berk-Seligson, 1986), such as English and Chinese, since they have word alignments with an inverted order (Winata et al., 2019). Owing to the limitation, more and more recent works start to build CS text synthesizers in a data-driven way. Garg et al. (2018) train a sequence generative adversarial model on real CS text to generate Chinese-English CS sentences. Chang et al. (2019) build a CS text synthesizer using the generative adversarial network, while several follow-up works (Samanta et al., 2019; Winata et al., 2019; Gonen and Goldberg, 2019) using different generative model techniques are also presented. More studies have been introduced to improve the synthesis quality such that we cannot exhaust them in this short summary. We refer readers to the recent survey (Winata et al., 2022; Sitaram et al., 2019).

Although many of these efforts had some success, the above-mentioned methods can only generate CS text in the same language pair sets used in training. Given the difficulties of acquiring CS data, this requirement hinders the scalability of these models to support more language pairs. Hence, in this paper, we take a step forward to explore the possibility of zero-shot transfer generalization in CS text synthesis and present GLOSS that can generate reasonable outputs.

7 Conclusion

In this paper, we develop a novel generalized code-switched text synthesizer, which can even generate code-switched sentences where the corresponding code-switched training data is unavailable. We introduce GLOSS that is built on top of a pre-trained multilingual machine translation model and augmented with an adapter or prefixes. The modularized design of learning specific parameters for mixing languages from a translated distribution helps the overall system generalization, hence, fulfilling our goal. Extensive experiments verify our methods' effectiveness qualitatively and quantitatively. In the future, we plan to investigate how our synthesizer performs on downstream tasks such as conversational understanding under a code-switched scenario.

Limitation

Our paper presents a pilot exploration of investigating a new setting in code-switched text synthesis — we allow the target language pair selection not limited to those for which we already have training data. Although we have shown the strength of GLOSS qualitatively and quantitatively, our experimental setting is still confined due to the dataset restriction — all the input text is in English. It would be an even harder challenge if the source languages are more diverse and we leave such exploration for future work.

Additionally, due to the computational restriction, in GLOSS, we only explore mBART50-MMT and an augment-MMT as our PMMTM. From the experimental results, we do observe the benefit of having a more stable PMMTM in GLOSS. We anticipate the models' performance can be further improved by leveraging more stronger PMMTM, and the exploration is left for the future.

Broader Impacts

Our proposed models are based on a model that is pre-trained on a large scale of multilingual machine translation data. It is known that the machine translation model could capture the bias reflecting the training data (Wang et al., 2022). Therefore, our models can potentially generate code-switched text containing offensive or biased content. We suggest that for deploying our model in any real-world applications, careful examination of the potential bias is an essential step.

Acknowledgements

The authors would like to thank Chris Hench, Chenyang Tao, Mingyu Derek Ma, Che-Ping Tsai, and Tanmay Parekh for their feedback and help regarding human evaluation. We also thank anonymous reviewers for their helpful feedback on the paper.

References

Peter Auer and Li Wei. 2008. *Handbook of multilingualism and multilingual communication*. Walter de Gruyter.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. 2019. Systematic generalization: What is required and can it be learned? In *7th International Conference on Learning Representations, ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL*.

Christos Baziotis, Mikel Artetxe, James Cross, and Shruti Bhosale. 2022. Multilingual machine translation with hyper-adapters. *arXiv preprint arXiv:2205.10835*.

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*.

Susan Berk-Seligson. 1986. Linguistic constraints on intrasentential code-switching: A study of spanish/hebrew bilingualism. *Language in society*.

Gayatri Bhat, Monojit Choudhury, and Kalika Bali. 2016. Grammatical constraints on intra-sentential code-switching: From theories to working models. *arXiv preprint arXiv:1612.04538*.

Ching-Ting Chang, Shun-Po Chuang, and Hung-yi Lee. 2019. Code-switching sentence generation by generative adversarial networks and its application to data augmentation. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*.

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, EACL*.

Ben Fei, Weidong Yang, Liwen Liu, Tianyue Luo, Rui Zhang, Yixuan Li, and Ying He. 2023. Self-supervised learning for pre-training 3d point clouds: A survey. *arXiv preprint arXiv:2305.04691*.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gökhan Tür, and Prem Natarajan. 2022. MASSIVE: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018. Code-switched language models using dual rnns and same-source pretraining. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*.

Hila Gonen and Yoav Goldberg. 2019. Language modeling for code-switching: Evaluation, integration of monolingual data, and discriminative training. 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 2019, Hong Kong, China, November 3-7, 2019*.

John J Gumperz. 1982. *Discourse strategies*. Cambridge University Press.

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*.

Thanh-Le Ha, Jan Niehues, and Alex Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation, IWSLT*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations, ICLR*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Kuan-Hao Huang, Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. Improving zero-shot cross-lingual transfer learning via robust training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kuan-Hao Huang, I-Hung Hsu, Premkumar Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Multilingual generative language models for zero-shot cross-lingual event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Trans. Assoc. Comput. Linguistics*.

Aravind K. Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th International Conference on Computational Linguistics, COLING*.

Ponsiano Kanijo. 2018. Code-switching and code-mixing errors among swahili-english bilinguals in tanzania. *Kiswahili*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. Gluecos: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*.

Grandee Lee, Xianghu Yue, and Haizhou Li. 2019. Linguistically motivated parallel data augmentation for code-switch language modeling. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*.

Ying Li and Pascale Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR*.

Dau-Cheng Lyu, Tien-Ping Tan, Eng Siong Chng, and Haizhou Li. 2010. Seame: a mandarin-english code-switching speech corpus in south-east asia. In *Eleventh Annual Conference of the International Speech Communication Association*.

Pedzisai Mashiri. 2002. Shona-english code-mixing in the speech of students at the university of zimbabwe. *Southern African Linguistics and Applied Language Studies*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics ACL*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en espanol: toward a typology of code-switching.

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, ACL*.

Adithya Pratapa and Monojit Choudhury. 2021. Comparing grammatical theories of code-mixing. In *Proceedings of the Seventh Workshop on Noisy User-generated Text, W-NUT*.

Laura Ruis and Brenden M. Lake. 2022. Improving systematic generalization through modularity and augmentation. *arXiv preprint arXiv:2202.10745*.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code switched text. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*.

David Sankoff. 1998. A formal production-based explanation of the facts of code-switching. *Bilingualism: language and cognition*.

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

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021. From machine translation to code-switching: Generating high-quality code-switched text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems NeurIPS*.

Jun Wang, Benjamin I. P. Rubinstein, and Trevor Cohn. 2022. Measuring and mitigating name biases in neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*.

Genta Indra Winata, Alham Fikri Aji, Zheng Xin Yong, and Thamar Solorio. 2022. The decades progress on code-switching research in NLP: A systematic survey on trends and challenges. *arXiv preprint arXiv:2212.09660*.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

A Dataset Details

Tab. 4 presents the dataset statistics for our automatic evaluation. The dataset is created by Gupta et al. (2020) and under a Creative Commons Attribution-NoDerivatives 4.0 International License.[7]

Language Pairs	Train	Dev.	Test
Es-En	196,725	2,000	2,000
De-En	188,131	2,000	2,000
Fr-En	193,922	2,000	2,000
Hi-En	248,330	2,000	2,000
Bn-En	163,893	2,000	2,000
Ml-En	178,453	2,000	2,000
Ta-En	11,380	2,000	2,000
Te-En	9,105	2,000	2,000

Table 4: Dataset statistics of the dataset provided by Gupta et al. (2020).

B Inter Annotator Agreement

We measure the mutual agreement rate among our human annotators by calculating the average absolute differences between the scores they give for the same instance. For example, if the semantic correctness score is given with score $(2, 2, 3)$. Then, the average absolute difference is 0.66. We then take a micro average across all our human-annotated instances. We get a score of 0.50 and 0.52 for the fluency and semantic correctness score for Chinese-English, respectively. As for Hindi-English, we get a score of 0.59 and 0.55 for the fluency and semantic correctness score. This indicates our experts agree with each other with only a little disagreement.

[7] To download the data, please refer to https://docs.google.com/forms/d/e/1FAIpQLSfR8st2eNu6oq5i499bglxrbJ2BYCQKfpHyaIYq6oS-KrDibA/viewform.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation Section

☑ A2. Did you discuss any potential risks of your work?
Broader Impacts Section

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Introduction Section and Abstract Section

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We did simple checking of the potential harmful information in the dataset, but since the date size is large, we couldn't perform manual check on every instance.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix A

C ☑ Did you run computational experiments?

Section 4 & 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.1 & Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Appendix B

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 5

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Appendix C

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Appendix C

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Appendix C

☑ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Broader Impacts Section

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Appendix C