CoCoa: An Encoder-Decoder Model for Controllable Code-switched Generation

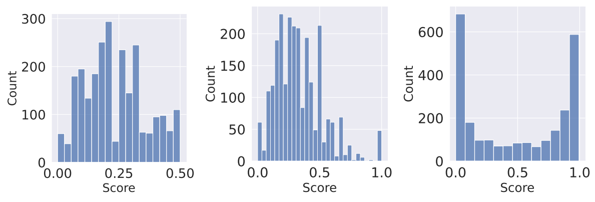
Sneha Mondal[1], **Ritika.**[2*], **Shreya Pathak**[2*], **Preethi Jyothi**[2], **Aravindan Raghuveer**[1]
[1]Google Research [2]IIT Bombay
{snehamondal, araghuveer}@google.com,
{ritikagoyal, shreyapathak, pjyothi}@cse.iitb.ac.in

Abstract

Code-switching has seen growing interest in recent years as an important multilingual NLP phenomenon. Generating code-switched text for data augmentation has been sufficiently well-explored. However, there is no prior work on generating code-switched text with fine-grained control on the degree of code-switching and the lexical choices used to convey formality. We present CoCoa, an encoder-decoder translation model that converts monolingual Hindi text to Hindi-English code-switched text with both encoder-side and decoder-side interventions to achieve fine-grained controllable generation. CoCoa can be invoked at test-time to synthesize code-switched text that is simultaneously faithful to syntactic and lexical attributes relevant to code-switching. CoCoa outputs were subjected to rigorous subjective and objective evaluations. Human evaluations establish that our outputs are of superior quality while being faithful to desired attributes. We show significantly improved BLEU scores when compared with human-generated code-switched references. Compared to competitive baselines, we show 10% reduction in perplexity on a language modeling task and also demonstrate clear improvements on a downstream code-switched sentiment analysis task.

1 Introduction

Bilingual speakers form a significant portion (current estimates of 43%[1]) of the world's population. To cater to the human-computer interaction needs of this user segment, Natural Language Generation (NLG) and Natural Language Understanding (NLU) tasks for code-switching (CS) are receiving increasing amount of attention from the research community (Zhang et al., 2021). Code-switching in



(a) CMI (b) SPI (c) Formality

Figure 1: Distribution of CMI, SPI and formality scores over CS samples in the AllCS test set.

a sentence typically involves switching between a matrix language L1 and an embedded language L2. A key challenge in CS that has not been previously addressed by generation models is to explicitly control for syntactic and lexical diversity (Doğruöz et al., 2021). Such controllable NLG models would help build robust computational models for CS text. We draw attention to three specific dimensions of diversity that are commonly observed in CS text. In this work, we aim to generate text that specifically spans these three CS dimensions.

1. Language Mix Ratio: The first *syntactic* dimension of diversity refers to the varying number of L1 and L2 words in a CS sentence that depends on a number of factors like language pair, socio-economic context, etc. For instance, Al-Azami (2006) find that among immigrant Bengalis in the UK, first-generation immigrants tend to use Bengali and English as L1 and L2 respectively, while the order is reversed for younger generation immigrants. *Code-mixing index* (CMI) (Gambäck and Das, 2014) is commonly used to quantify the ratio of L1 vs. L2.

2. Language Burstiness: The second *syntactic* dimension of diversity captures burstiness, i.e. the length of homogenous spans of L1 and L2, in CS text. This is also a function of the languages involved and social contexts. For instance, Czech-English speakers switch to English for high-information content words in prominent prosodic positions while speaking Czech (Myslín and Levy,

* Equal contribution
[1]https://ilanguages.org/bilingual.php

2015). *I-index* (Guzmán et al., 2017), which we also refer to as *switch-point index* (SPI) is popularly used to quantify the extent of burstiness.

3. Formality: This *lexical* dimension of diversity refers to the choice of words used in a CS sentence. A CS sentence can be constructed using either formal or informal words depending on whether it is used in a news broadcast or a social media post, respectively. Many studies suggest that CS patterns and word choices are speaker-dependent (Vu et al., 2013) and sometimes even gender-dependent (Finnis, 2014).

We further substantiate the practical relevance of the above three dimensions of diversity by analyzing the test set of the ALLCS benchmark (Tarunesh et al., 2021a) which consists of 2.5K human-generated Hindi-English sentences. We compute CMI/SPI using counts of L1 and L2 words/spans in CS text and formality is determined using a pretrained classifier.[2] Figure 1 shows significant variation of CS text across all three dimensions, thus offering evidence of diversity in real CS text. Figure 4 in Appendix A contains ALLCS examples that span the three dimensions of interest.

In this paper, we tackle the problem of generating CS text from monolingual text while providing inference-time levers to control for CMI, SPI and formality of the generated CS text. To the best of our knowledge, there is no prior work that can generate CS text while controlling for these attributes. For NLU models, data augmentation (Chang et al., 2018; Volpi et al., 2018; Shen et al., 2020) is a common and effective way to make them robust to a wide variety of inputs. Therefore, it follows that building controllable code-switched generation models would be very useful in making NLU (via data augmentation) and NLG models robust to lexical and syntactic diversity.

We propose COCOA for **co**ntrollable **co**de-switched gener**a**tion using a machine translation-based model where the source and target languages are monolingual Hindi text and Hindi-English CS text, respectively. In practical scenarios, parallel data with diverse vocabulary in conjunction with diverse attribute (CMI, SPI, formality) values for CS text is not easily available. We tackle this challenge through task-specific multi-task training objectives. We employ encoder-side training to control for attributes when parallel training data is available, and decoder-side control via beam re-weighting when parallel data is not available. In summary, we make the following four key contributions:

- We introduce the problem of controllable code-switching to enable generation of CS text with a desired set of attributes. (Sections 3.1)
- We propose a modeling methodology for controllable code-switched generation by using both encoder and decoder level controls (Section 3.2).
- We demonstrate the efficacy of the proposed approach with detailed ablations and comparison to state-of-the-art benchmarks. (Section 4, 5, 6).
- We release a new Hindi-English CS dataset, DIVERSE-ALLCS which is an extension to the ALLCS dataset with additional diversity of CMI and SPI (Section 7).

2 Related Work

2.1 Diversity in Code Switching: Linguistic Perspective

Different from conventional languages, code-switched languages are complex, personalized (Bawa et al., 2020), evolving languages (Pratapa and Choudhury, 2017) that depend on linguistic (Al-Azami, 2006), social (Gardner-Chloros and Edwards, 2004) and economic contexts (Mohanty, 2006). Below we discuss a few reasons that cause CS diversity and refer the reader to recent surveys (Sitaram et al., 2019; Doğruöz et al., 2021) for a more detailed treatment of this topic.

In a linguist's perspective of code-switching, Doğruöz et al. (2021) makes a strong argument that computational linguistics (primarily via large language models) are constrained by the lack of available of diverse CS training data, evaluation benchmarks and absence of user-facing applications. Systematically producing diverse CS text for robustifying NLU and NLG tasks is stil an open problem (Doğruöz et al., 2021). Though computational models of CS exist, they fall short in terms of coverage of the natural utterances that a bilingual speaker would produce (Pratapa et al., 2018).

Despite significant advances in NLP powered by large pretrained language models, generating and understanding CS text with the diversity and richness required by applications is harder compared to tasks involving monolingual data (Winata et al., 2021). Doğruöz et al. (2021) observe that while benchmarks like LINCE (Aguilar et al., 2020) and GLUECoS (Khanuja et al., 2020a) serve as critical resources for the CS research community, they do

[2]More details about these attributes are in Section 3.1.

not yet represent the entire spectrum of CS. Specifically, many tasks in the above benchmarks consist of annotated tweets which only represent a certain type of CS. In this paper, we address this gap by generating diverse CS text with dimensions of diversity inspired from linguistic usage patterns of CS.

2.2 Computational models for Code-switched Generation

Prior work on generating synthetic code-switched text has explored the use of purely generative models (Garg et al., 2018; Samanta et al., 2019; Gao et al., 2019; Chang et al., 2019), sequence-to-sequence models that leverage parallel monolingual text (Winata et al., 2019, 2018) and transduction models that translate from the matrix or embedded language to code-switched text (Gautam et al., 2021; Gupta et al., 2021; Tarunesh et al., 2021b). Ours is the first work to tackle the problem of controllable code-switched generation.

2.3 Controlled Text Generation

Controlled text generation (Hu et al., 2017) using large pretrained LMs is a popular subarea of NLG, where control is typically exercised by finetuning a pretrained LM for an attribute(e.g., (Keskar et al., 2019)) or combining the pretrained LM with attribute classifiers or other experts (Dathathri et al., 2019; Yang and Klein, 2021; Liu et al., 2021). A popular technique for controlled translation is to introduce a discrete tag at the source and/or the target (Sennrich et al., 2016). Recent work by Schioppa et al. (2021) explored the use of additive control vectors instead of discrete tags for more fine-grained control. There is no prior work that explores controllable generation for code-switching and this paper aims to address that gap.

3 Methodology

3.1 Problem Description

Our goal is to build a model CoCoa that transforms a monolingual sentence into a semantically equivalent CS sentence, while simultaneously satisfying multiple CS attributes specified by the user at test time. The three main architectural choices in CoCoa are:

• **Model architecture:** Transformer-based encoder-decoder models are currently the ubiquitous choice for neural translation (Vaswani et al., 2017). We use a pretrained text-to-text multilingual model mT5 (Xue et al., 2021), as our base model.

• **Enabling controlled code-switched generation:** Attribute-aware generation can be achieved via both encoder and decoder-side controls. Encoder-side controls assume the availability of (at least a small amount of) parallel monolingual to CS text with attribute annotations that are further used to train the model. In contrast, decoder-side controls are applied only during the decoding step and hence do not require any training-time intervention. Either approach can be adopted depending on the type of annotated data that is available.

• **Multi-task training:** We introduce a multi-task training objective that involves jointly training two auxiliary translation tasks from English to Hindi, and Hindi to English, along with the primary translation task of Hindi to Hindi-English. For the primary task, 20% of the source Hindi tokens are masked. This masking step dramatically reduces the model's propensity to copy source tokens, and was found to be critical for performance especially when using decoder-side controls.

In this work, we aim at controlling three CS attributes: CMI, SPI and formality. CMI and SPI values can be deterministically computed for a CS sentence. CMI for a CS sentence of length n with η_1, η_2 tokens in L1 and L2, respectively (i.e. $n = \eta_1 + \eta_2$), is written as $1 - \frac{\max(\eta_1,\eta_2)}{n}$. Low CMI values indicate monolingualism and the highest CMI value of 0.5 indicates an equal number of L1 and L2 tokens in the sentence. SPI is computed as $\frac{\sum_{i=1}^{n-1} S(i,i+1)}{n-1}$, where $S(i, i+1)$ is 1 if tokens i and $i+1$ belong to different languages, and is 0 otherwise. SPI varies between 0 (monolingual utterance) and 1 (consecutive tokens coming from L1 and L2). CMI and SPI are dependent attributes. A low CMI score typically results in a low SPI score, since low code-switching implies fewer switches between L1 and L2. In contrast, a high CMI score can be attained with either a low SPI score (longer language spans, less interleaving) or a high SPI score (short language spans, frequent interleaving). Unlike CMI and SPI that are deterministic functions of the CS text, formality values are obtained using a binary classifier trained to detect formality in English text. More details about finetuning the formality classifier to make it more amenable to CS are in Appendix B.2.

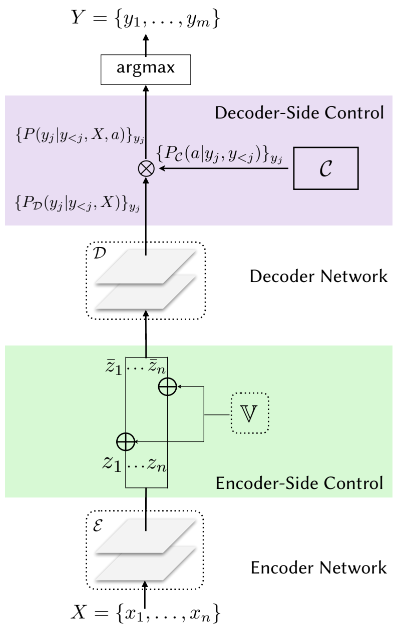

$Y = \{y_1, \ldots, y_m\}$
argmax
Decoder-Side Control
$\{P(y_j|y_{<j}, X, a)\}_{y_j}$
$\{P_{\mathcal{C}}(a|y_j, y_{<j})\}_{y_j}$
$\mathcal{C}$
$\{P_{\mathcal{D}}(y_j|y_{<j}, X)\}_{y_j}$
$\mathcal{D}$
Decoder Network
$\bar{z}_1 \ldots \bar{z}_n$
$\mathbb{V}$
$z_1 \ldots z_n$
Encoder-Side Control
$\mathcal{E}$
Encoder Network
$X = \{x_1, \ldots, x_n\}$

Figure 2: Schematic diagram illustrating the main components of CoCoa that transforms a monolingual sentence X to a controlled CS sentence Y. $\bigoplus$ and $\bigotimes$ denote an element-wise addition and multiplication between vectors.

3.2 Our Approach

CoCoa scaffolds on a Transformer-based translation model consisting of multi-layered encoder and decoder networks. The encoder $\mathcal{E}$ converts a monolingual sentence $X = \{x_i\}_{i=1}^n$ into a sequence of higher-level representations denoted by $Z = \{z_i\}_{i=1}^n, z_i \in \mathbb{R}^d$ that is further passed to the decoder $\mathcal{D}$. $\mathcal{D}$ autoregressively converts the encoded representations Z into the target CS sequence $Y = \{y_j\}_{j=1}^m$ one token at a time. CoCoa additionally provides encoder-side and decoder-side controls to the base Transformer model to support fine-grained CS generation. Figure 2 illustrates the main components of CoCoa.

Encoder-side Control: A commonly adopted technique for encoder-side control is to introduce an explicit attribute tag at the start of the source sentence as an input to the model (Kobus et al., 2016; Sennrich et al., 2016). The main limitation of tag-based control is in dealing with continuous-valued attributes that have to be discretized into bins. Keeping the number of bins too small might result in coarse characterizations of the attribute, while increasing the number of bins could lead to limited data being available for each bin. A more natural handling of continuous-valued attributes is to directly learn a vector embedding for each attribute that can be added to the encoder representations.

Motivated by Schioppa et al. (2021), we define an attribute vector $\mathbb{V}_a \in \mathbb{R}^d$ for each CS attribute a. Each $\mathbb{V}_a$ is scaled with a weight w_a that is equal to the attribute value. Multiple attributes are handled using a linear combination of the attribute vectors, $\mathbb{V} = \sum_a w_a \mathbb{V}_a$. $\mathbb{V}$ is added to each encoder representation z_i from $\mathcal{E}$ before being passed as input to $\mathcal{D}$. Thus, z_i now becomes $z_i + \mathbb{V}$. The attribute vectors are learned during training using the end-to-end translation objective. Parallel text without attribute annotations can still be used during training by setting $\mathbb{V}$ to $\mathbf{0}$.

Decoder-side Control: Decoder-side control is desirable when we want to control for attributes like formality for which we do not have parallel text. Towards this, we borrow the idea of conditioning an existing autoregressive model on a desired attribute from the FUDGE framework (Yang and Klein, 2021).

FUDGE aims to alter the output probabilities from a trained decoder with the help of an attribute classifier. The Transformer-based decoder autoregressively models the conditional probability distribution of a CS token at time-step j given the entire monolingual sequence, i.e., $P(y_j|y_{<j}, X)$. If we want to additionally condition the decoder on an attribute a, the required probability distribution would become:

$$\begin{aligned} P(y_j|y_{<j}, X, a) &\propto P(a|y_{1:j}, X) P(y_j|y_{<j}, X) \\ &\approx P(a|y_{1:j}) P(y_j|y_{<j}, X) \quad (1) \end{aligned}$$

We assume conditional independence between the attribute and the monolingual input sentence, given the prefix of the output sentence. $P(y_j|y_{<j}, X)$ from Equation 1 is already modeled by the Transformer-based decoder. $P(a|y_{1:j})$ can be estimated using a binary classifier for a given the prefix $y_{1:j}$. The binary classifier $\mathcal{C}$ predicts whether the prefix $y_{1:j}$, when expanded to completion, will satisfy attribute a or not. During decoding, new probabilities for each output word y_j are obtained by multiplying and re-normalizing the probability distributions shown in Equation 1.

4 Experiments and Results

4.1 Evaluation Metrics

Semantic consistency with the source monolingual sentence is computed using BLEU scores (Papineni et al., 2002) between a reference CS sentence and the generated sentence. Attribute faithfulness for CMI is measured using two metrics: CMI_{acc} and CMI_{corr}. CMI_{acc} measures the percentage of test instances where the binned CMI score of the generated CS sentence exactly matches the binned CMI score of the reference CS sentence. CMI_{corr} measures Pearson correlation between real-valued CMI scores of generated CS sentences and reference CS sentences. Analogously, we can define SPI_{acc} and SPI_{corr}. Unless specified otherwise, we use 3 bins to compute CMI_{acc} and 2 bins to compute SPI_{acc}.

4.2 Model Architecture and Implementation Details

For all our experiments, we start with the publicly available `mt5-small` checkpoint[3], which is further finetuned on task-specific training data. We use the AdamW optimizer with a constant learning rate of 5e-4. All our models are trained on a single NVIDIA A100 GPU. We use the harmonic mean of BLEU and attribute correlation (CMI_{corr} and/or SPI_{corr}) as the checkpoint selection criterion to ensure that the generated outputs are both attribute-preserving and semantically meaningful. We also implement the multi-task training objective of monolingual and masked CS translation that was described in Section 3.1. More implementation details appear in Appendix B.1.

For decoder-side control, we use the formality adapter[4] by Krishna et al. (2020), stacked on an `XLM-R` Hindi language adapter[5] by Pfeiffer et al. (2020). Recall that our approach requires the classifier to act on sentence *prefixes* at every decoding step, instead of complete sentences. To this end, we finetune the formality classifier on prefixes of formal and informal Hindi sentences. More details of the finetuning dataset are in Appendix B.2.

4.3 Datasets

For parallel Hindi to Hindi-English text, we use the ALLCS corpus (Tarunesh et al., 2021b) that consists of around 21K, 1K, and 2.5K samples in training, dev, and test splits respectively. The domain is largely conversational as many of the CS sentences are extracted from movie scripts. We use the IIT Bombay Hindi-English parallel corpus (Kunchukuttan et al., 2017) for multi-task training.

[3]https://huggingface.co/google/mt5-small

[4]https://adapterhub.ml/adapters/martiansideofthemoon/xlm-roberta-base_formality_classify_gyafc_pfeiffer/

[5]https://adapterhub.ml/adapters/ukp/xlm-roberta-base-hi-wiki_pfeiffer/

Model	BLEU	CMI_{acc}	CMI_{corr}	SPI_{acc}	SPI_{corr}
CoCoA $_{\text{zc}}$	61.06	0.50	0.41	0.66	0.55
CoCoA $_{\text{cmi}}$	61.79	**0.88**	**0.92**	0.84	0.80
CoCoA $_{\text{spi}}$	61.92	0.65	0.61	0.88	0.89
CoCoA $_{\text{m,zc}}$	60.74	0.53	0.41	0.67	0.53
CoCoA $_{\text{m,cmi}}$	62.72	**0.88**	0.91	0.82	0.78
CoCoA $_{\text{m,spi}}$	61.32	0.68	0.63	0.89	0.87
CoCoA $_{\text{cmi,spi}}$	62.70	0.82	0.87	**0.91**	**0.92**
CoCoA $_{\text{m,cmi,spi}}$	**64.37**	0.85	0.90	0.90	0.91

Table 1: Evaluating CoCoA using single and multiple attribute control of CMI and SPI. Subscripts *zc* and *m* refer to zero control and use of multi-task training, respectively; others are self-explanatory.

4.4 Encoder-side Control

4.4.1 Single and Multi-Attribute Control

Table 1 details the performance of CoCoA in various control settings. Going from CoCoA_{zc} to $\text{CoCoA}_{\text{cmi}}$, we see substantial gains in CMI_{acc} and CMI_{corr}. A similar gain in SPI_{acc} and SPI_{corr} is seen when we compare CoCoA_{zc} with $\text{CoCoA}_{\text{spi}}$. Controlling for both CMI and SPI simultaneously (c.f., last two rows in Table 1) leads to a significant boost in BLEU scores along with maintaining high CMI/SPI correlations. Multi-task training leads to a clear improvement in BLEU scores, but does not significantly vary performance on the two attributes.

To understand how our model performs with using smaller amounts of CS training data, we train CoCoA on a random half of ALLCS. Even with using only 50% of ALLCS during training, we achieve close to 90% on all evaluation metrics when compared to using ALLCS in full (BLEU of 60.11; CMI_{acc}, CMI_{corr}, SPI_{acc} and SPI_{corr} values of 0.74, 0.82, 0.79 and 0.84, respectively). This shows that CoCoA is able to generate high-quality and diverse CS sentences even when trained on fairly limited amounts of CS data.

4.4.2 Human Evaluation

We sampled 500 Hindi sentences from ALLCS test and compared CS outputs produced by different CoCoA variants, a competitive baseline (called TCS) by Tarunesh et al. (2021b), and human references. Three human raters with native fluency in Hindi and English assessed the model outputs along two dimensions: 1) *naturalness* indicating

Method	Automated metrics				Naturalness			Semantic Consistency		
	verbatim copy	source BLEU	CMI_{acc}	SPI_{acc}	very natural	somewhat natural	not at all natural	very well	somewhat well	not at all
Human references	1.6	50.60	100	100	94.7	4.4	0.9	94.8	3.8	1.4
TCS	3.6	39.07	48.2	69.2	71.9	15.5	12.6	66.6	19.3	14.1
CoCoA $_{\text{zc}}$	11.4	64.08	47.2	69.6	89.6	9.7	0.7	86.8	12.4	0.8
CoCoA $_{\text{cmi}}$	2.7	53.13	86.0	87.0	80.9	16.4	2.7	78.0	19.2	2.8
CoCoA $_{\text{spi}}$	2.6	53.43	82.4	92.4	79.3	18.0	2.7	76.2	20.8	3.0
CoCoA $_{\text{m,cmi}}$	2.4	51.82	87.7	86.8	88.4	9.9	1.7	86.8	11.4	1.8

Table 2: Human evaluations for encoder-side control of CMI or SPI, expressed as a %age.

Model	Formality			Naturalness			Semantic Consistency			CMI_{corr}
	more	equal	less	very natural	somewhat natural	not at all natural	very well	somewhat well	not at all	
CoCoA $_{\text{zc}}$	54.4	23.6	22.0	67.6	22.4	10.0	59.0	30.8	10.2	0.42
CoCoA $_{\text{m,zc}}$	56.6	23.0	20.4	73.8	23.8	2.4	67.6	29.4	3.0	0.44
CoCoA $_{\text{m,cmi}}$	45.0	43.0	12.0	85.4	12.4	2.2	79.8	18.0	2.2	0.85

Table 3: Human evaluations for decoder-side control of formality, expressed as a %age.

whether the code-switching is natural, and 2) *semantic consistency*, measuring how well the model output preserves the meaning of the monolingual Hindi sentence. Each dimension was rated using one of three quality scores shown in Table 2. Table 2 also lists a few automated metrics, including “verbatim copy" that refers to the percentage of model outputs that are an exact copy of the Hindi source and “source BLEU" which is the BLEU score between the generated output and the Hindi source.

All CoCoA variants perform significantly better than TCS on both naturalness and semantic consistency scores. CMI_{acc} and SPI_{acc} increase substantially when using controlled CoCoA variants compared to CoCoA_{zc}. Among all the models, $\text{CoCoA}_{\text{m,cmi}}$ has the least propensity to copy, as evidenced by the lowest scores of verbatim copying and source-BLEU. Moreover, $\text{CoCoA}_{\text{m,cmi}}$ attains naturalness and semantic consistency scores very close to CoCoA_{zc}, despite far less copying, which is truly indicative of superior code-switching.

4.5 Decoder-side Control

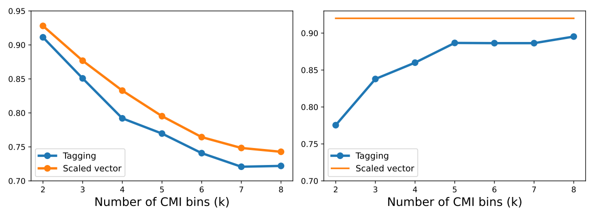
We sampled 500 monolingual sentences from the ALLCS test split and generated two model outputs with and without decoder-side control for formality.[6] Raters were shown both outputs, and asked to rate if one was more formal, less formal, or equally formal compared to the other. The formality-controlled outputs were also evaluated for naturalness and semantic consistency, as in Section 4.4.2. From Table 3, we see that for all models a significant number of formality-controlled outputs are marked as more formal than their no-control counterparts. Comparing CoCoA_{zc} and $\text{CoCoA}_{\text{m,zc}}$, we find that multi-task training dramatically improves naturalness and semantic consistency of generated outputs while simultaneously making them more formal. The ability of the model to generate more formal outputs drops a little when also controlling for CMI. However, interestingly, the encoder-side control for CMI makes the model more robust to decoder-side interventions, as evidenced by the higher naturalness and semantic consistency scores in the last row compared to the previous two rows.

5 Ablations and Model Variants

Comparison with Tagging. Figure 3b shows how tagging compares to the scaled vector approach for encoder-side control. On increasing the number of bins (k) from 3 to 8 for tagging, CMI_{corr} increases from 0.77 to 0.89, making it more comparable to the CMI_{corr} of 0.92 using the scaled approach. As we increase the number of bins, k-class CMI_{acc} drops for both the tagging and scaled vector approaches, with the latter showing a small but consistent improvement across bins.

Interpolation to Unseen Attributes. The tagging technique will not be able to generalize to CMI/SPI bins that were never seen during training. In con-

[6]Formality is measured only using human evaluations. We do not have ground-truth formal CS reference sentences in order to compute metrics like BLEU.



(a) CMI bin accuracy (b) CMI correlation

Figure 3: Comparing tagging with the scaled vector approach for encoder-side control.

trast, the scaled vector technique could potentially interpolate to unseen CMI/SPI values. We test this out by holding out specific CMI bins and evaluating performance on all three bins in Table 4. As expected, the performance on the held-out bin drops when considering each bin, but only by fairly small margins. This validates our claim that the scaled technique is able to successfully interpolate to unseen attribute values. A similar analysis is done for SPI in Table 10 in Appendix C.

Held out bin	Bin 1 performance			Bin 2 performance			Bin 3 performance		
	BLEU	CMI_{acc}	CMI_{corr}	BLEU	CMI_{acc}	CMI_{corr}	BLEU	CMI_{acc}	CMI_{corr}
None	69.8	0.93	0.71	55.5	0.75	0.49	40.9	0.75	0.41
Bin 1	**67.2**	**0.83**	**0.62**	56.2	0.73	0.41	42.8	0.80	0.43
Bin 2	69.2	0.91	0.63	**56.0**	**0.65**	**0.46**	40.9	0.71	0.46
Bin 3	69.5	0.94	0.70	55.9	0.75	0.42	**35.8**	**0.68**	**0.36**

Table 4: Held-out performance of $\text{CoCoA}_{\text{cmi}}$ over CMI bins. Held-out bin is highlighted in bold.

Freezing Style Vector. Is it important to learn the attribute vector or will performance be as good with an attribute vector whose weights are frozen? Table 5 shows that while there is a small gain in performance with using a trained attribute vector, most of the model's performance can be attributed to the decoder effectively learning how to scale the attribute vector regardless of its actual values.

6 Downstream Tasks

6.1 Language Model Perplexity

We train language models (LMs) on text from CoCoA and other baseline approaches and evaluate perplexity on two different test sets containing real code-switched data. We start with the XLM-RoBERTa checkpoint[7] and finetune it on different synthetic datasets with a causal LM loss, using the script provided by Huggingface.[8]

$\text{CoCoA}_{\text{m,cmi,spi}}$ is compared against two prior techniques - TCS (Tarunesh et al., 2021b) and GCM (Rizvi et al., 2021). 100K real monolingual sentences are sampled from the IIT-B Hindi corpus and used to generate 200K synthetic CS sentences from each technique. Note that $\text{CoCoA}_{\text{m,cmi,spi}}$ requires both CMI and SPI values to be provided during inference. We experiment with two sampling schemes for assigning CMI and SPI scores to monolingual sentences:

1. **Random:** Each sentence is assigned a CMI score uniformly sampled from $(0, 0.5]$ and an SPI score uniformly sampled from $(0, 1]$. This scheme does not account for the relationship between CMI and SPI.

2. **Discretized:** Each sentence is first assigned a CMI score uniformly sampled from $\{1/n, 2/n, \ldots, \lceil n/2 \rceil/n\}$, where n is the number of tokens in the monolingual sentence. If the sampled CMI score is less than or equal to 0.33, the SPI is uniformly sampled between $(0, 0.6]$; otherwise it is uniformly sampled between $(0, 1]$. This scheme ensures that the assigned CMI score is meaningful for a sentence of length n. It also accounts for the relationship between CMI and SPI by enforcing that low CMI scores co-occur only with low SPI scores, while high CMI scores can co-occur with either low or high SPI scores.

Table 6 compares perplexities on the test split of AllCS. CoCoA models yield the lowest perplexity among all synthetic data generation methods, and the discretized scheme attains a lower score compared to the random scheme. The synthetic CS data is derived starting from the IIT-B Hindi corpus. LMs trained on this synthetic data are evaluated on an out-of-domain test set, and hence do not outperform the LM trained on AllCS-train (that is distributionally similar to AllCS-test). Table 7 shows how the trend reverses when evaluating perplexities on a new test set from HinGE (Srivastava and Singh, 2021). The LM trained on synthetic

Method	BLEU	CMI_{acc}	CMI_{corr}
CoCoA $_{\text{cmi}}$ Frozen	60.96	0.84	0.83
CoCoA $_{\text{cmi}}$ Trained	61.79	0.88	0.92

Table 5: BLEU, CMI_{acc}, CMI_{corr} scores after freezing or training the CMI attribute vector.

[7]https://huggingface.co/xlm-roberta-base

[8]https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_clm.py

Train set	Test perplexity ↓
GCM	253.26
TCS	214.04
CoCoa $_{zc}$	212.61
CoCoa $_{m,cmi,spi}$Random	194.47 ± 0.38
CoCoa $_{m,cmi,spi}$Discretized	192.55 ± 0.41
AllCS Train (21K samples)	110.42

Table 6: LM perplexities on AllCS test set.

Train set	Test perplexity ↓
CoCoa $_{m,cmi,spi,50pc}$	647.65 ± 0.29
CoCoa $_{m,cmi,spi}$	613.78 ± 0.53
AllCS Train-50pc	794.38
AllCS Train	637.41

Table 7: LM perplexities on HinGE. Synthetic CS data is generated with the discretized sampling scheme.

Model	Max F1 ↑	Mean F1 ↑
TCS	58.30	57.31
GCM	59.72	58.67
CoCoa $_{m,cmi,spi}$ Random	59.90	59.12
CoCoa $_{m,cmi,spi}$ Discretized	61.33	59.43
AllCS Train (21K samples)	61.28	59.56

Table 8: Max and mean F1 scores on GLUECoS sentiment analysis task.

data from CoCoa$_{m,cmi,spi}$ yields a 3.5% improvement over the LM trained on AllCS. We also train an LM on 50% of AllCS and another on synthetic sentences from CoCoa$_{m,cmi,spi,50pc}$ trained on half of AllCS. We observe only a small degradation in test perplexity when CS training data is halved for CoCoa (c.f., first two rows in Table 7). The LM trained on 50% of AllCS sees a large drop in performance (794.38), while CoCoa trained on this subset is still able to generate diverse samples with little loss in quality (647.65).

6.2 Code-switched Sentiment Analysis

We show the effect of pretraining with synthetic text from CoCoa on a downstream task with CS inputs. We select sentiment analysis from the GLUECoS benchmark (Khanuja et al., 2020b) consisting of roughly 13K, 3K and 1K training/dev/test instances. We use the same sets of 200K synthetic samples described earlier for TCS, GCM and CoCoa$_{m,cmi,spi}$. We pretrain an mBERT model (Pires et al., 2019) on this synthetic text followed by finetuning on the sentiment analysis data. We identify the best model via the dev set and report max/mean F1 scores on the test set computed over five random seeds. Table 8 shows that CoCoa performs better than both TCS and GCM, and the discretized sampling improves slightly over the random sampling.

Naturalness			Semantic Consistency		
very natural	somewhat natural	not at all natural	very well	somewhat well	not at all
83.87	12.21	3.91	83.85	12.78	3.35

Table 9: Distribution of human evaluation scores on Diverse-AllCS, expressed as a %age.

7 The Diverse All-CS Dataset

We release a new dataset Diverse-AllCS which is an extension of AllCS with more natural diversity in CMI and SPI.[9] We select 1928 sentences from AllCS-test with a sentence length between 5 and 15 tokens. To mimic the natural distribution of CMI/SPI scores, we refer to AllCS-train and divide CMI/SPI values into two bins, resulting in four CMI, SPI combinations. We use CoCoa$_{m,cmi,spi}$ to generate model outputs and sample proportional to the number of AllCS sentences assigned to each CMI/SPI combination. After removing duplicates, we have 5380 unique CS sentences that were subjected to human evaluations. Table 9 shows the distribution of scores for naturalness and semantic consistency. We release the subset of data that is at least “Somewhat Natural" and scored “Somewhat well" or higher on semantic consistency.

8 Discussion and Conclusion

We list a few observations from our experiments.
1. We find that decoder-side control is more brittle and requires interventions like multi-task training to improve on naturalness and consistency of the generated outputs. Encoder-side control, in comparison, is more robust. The trade-off is that the latter requires parallel data that is more tedious to acquire, while decoder-side control does not.
2. The tagged approach nears the scaled vector approach in performance on increasing the number of bins to 8. However, the scaled vector has the advantage of interpolating to nearby CMI/SPI values without having seen them explicitly during

[9]Diverse-AllCS is available at https://www.cse.iitb.ac.in/~pjyothi/CoCoa.

training. 3. CMI/SPI could also be controlled on the decoder side. However, our current method of classifier-based reweighting would not work since a classifier will not be able to meaningfully predict CMI/SPI, which are length-dependent values, based on only a sentence prefix.

In conclusion, we establish CoCoa as a state-of-the-art controllable code-switched generation tool with test-time controls that outperforms existing unconstrained baselines.

9 Limitations

Our method is currently evaluated on a single language pair. The main bottleneck that prevented us from evaluating on a second language pair was the lack of a high-quality dataset like AllCS. We believe our model will be able to generalize to a different language pair given similar amounts of resources of comparable quality. We fully intend to test this out as future work after creating such resources for other languages and publicly releasing them to the community for further use. It would also be interesting to investigate how to work with larger but noisier code-switched text mined from social media text and whether such resources could be used to supplement a resource like AllCS of much smaller size.

Another limitation relates to the model itself. CoCoa is an encoder-decoder model and inherits the limitations associated with such models, the main one being that the model has difficulty scaling to long sentences.

10 Ethical Considerations

As with all language generation models, the generated outputs are a function of the data that was used to train the model. One should be cognizant of this when deploying such a model in applications that could benefit from controllable code-switching, e.g., chatbots.

11 Acknowledgements

The fourth author would like to thank Google Research India for the compute support and the grant towards research on models for code-switching. We are very grateful to the Task Mate team (especially Auric Bonifacio Quintana) for their inputs on framing annotation guidelines and helping us scale human evals on their platform. We would also like to thank Abhirut Gupta from Google Research for useful discussions on training multilingual text-to-text models.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. Lince: A centralized benchmark for linguistic code-switching evaluation. *arXiv preprint arXiv:2005.04322*.

Salman Al-Azami. 2006. Linguistic manipulations in the bengali language by the bangladeshis in manchester. *South Asian Cultural Studies*, 1(1):53–59.

Anshul Bawa, Pranav Khadpe, Pratik Joshi, Kalika Bali, and Monojit Choudhury. 2020. Do multilingual users prefer chat-bots that code-mix? let's nudge and find out! *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–23.

Ching-Ting Chang, Shun-Po Chuang, and Hung-Yi Lee. 2018. Code-switching sentence generation by generative adversarial networks and its application to data augmentation. *arXiv preprint arXiv:1811.02356*.

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 *Proc. Interspeech 2019*, pages 554–558.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

A Seza Doğruöz, Sunayana Sitaram, Barbara E Bullock, and Almeida Jacqueline Toribio. 2021. A survey of code-switching: Linguistic and social perspectives for language technologies. In *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*. Association for Computational Linguistics.

Katerina A Finnis. 2014. Variation within a greek-cypriot community of practice in london: Code-switching, gender, and identity. *Language in society*, 43(3):287–310.

Björn Gambäck and Amitava Das. 2014. On measuring the complexity of code-mixing. In *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*, pages 1–7.

Yingying Gao, Junlan Feng, Ying Liu, Leijing Hou, Xin Pan, and Yong Ma. 2019. Code-switching sentence generation by bert and generative adversarial networks. In *INTERSPEECH*, pages 3525–3529.

Penelope Gardner-Chloros and Malcolm Edwards. 2004. Assumptions behind grammatical approaches to code-switching: when the blueprint is a red herring. *Transactions of the Philological Society*, 102(1):103–129.

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*, pages 3078–3083, Brussels, Belgium. Association for Computational Linguistics.

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.

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.

Gualberto A Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. In *INTERSPEECH*, pages 67–71.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020a. Gluecos: An evaluation benchmark for code-switched nlp. *arXiv preprint arXiv:2004.12376*.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020b. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Catherine Kobus, Josep Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *arXiv preprint arXiv:1612.06140*.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The IIT bombay english-hindi parallel corpus. *CoRR*, abs/1710.02855.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Ajit K Mohanty. 2006. Multilingualism of the unequals and predicaments of education in india: Mother tongue or other tongue. *Imagining multilingual schools*, pages 262–283.

Mark Myslín and Roger Levy. 2015. Code-switching and predictability of meaning in discourse. *Language*, pages 871–905.

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*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. *arXiv preprint*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

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.

Adithya Pratapa and Monojit Choudhury. 2017. Quantitative characterization of code switching patterns in complex multi-party conversations: A case study on hindi movie scripts. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 75–84.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. *arXiv preprint arXiv:1803.06535*.

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.

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-19*, pages 5175–5181. International Joint Conferences on Artificial Intelligence Organization.

Andrea Schioppa, David Vilar, Artem Sokolov, and Katja Filippova. 2021. Controlling machine translation for multiple attributes with additive interventions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6676–6696, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*.

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*.

Vivek Srivastava and Mayank Singh. 2021. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *arXiv preprint arXiv:2107.03760*.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021a. From machine translation to code-switching: Generating high-quality code-switched text. *arXiv preprint arXiv:2107.06483*.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021b. 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 (Volume 1: Long Papers)*, pages 3154–3169, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. 2018. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31.

Ngoc Thang Vu, Heike Adel, and Tanja Schultz. 2013. An investigation of code-switching attitude dependent language modeling. In *International Conference on Statistical Language and Speech Processing*, pages 297–308. Springer.

Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 142–153, Online. Association for Computational Linguistics.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Learn to code-switch: Data augmentation using copy mechanism on language modeling. *CoRR*, abs/1810.10254.

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 *CoNLL*.

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.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Daniel Yue Zhang, Jonathan Hueser, Yao Li, and Sarah Campbell. 2021. Language-agnostic and language-aware multilingual natural language understanding for large-scale intelligent voice assistant application. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1523–1532. IEEE.

A Diverse Examples From AllCS

Figure 4 contains real CS utterances from the AllCS test set, with varying scores across the three diversity dimensions of interest. First two rows in this table illustrate that a low CMI score co-occurs typically with low SPI scores. In contrast, a high CMI score can co-occur with either a low or a high SPI score, as seen in the last two rows of the table.

B Implementation Details

B.1 Multi-Task Objective

To implement the multi-task objective, we proceed as follows. First, we create a monolingual translation corpus that consists of 800K parallel samples randomly selected from the IIT Bombay Hindi-English corpus[10]. Next, we create a masked CS translation corpus from the AllCS train split by masking each source Hindi token independently with a probability of 0.2. We then make 10 copies of this masked CS translation corpus, and merge it with the monolingual translation corpus. Upsampling CS training data in this manner reduces the skew between the size of monolingual translation (800K) and CS translation corpus (≈21K). We train for 3 epochs on this merged dataset, with a task-specific prefix indicating whether the model should perform monolingual or CS translation for the current instance. Metrics are evaluated on the AllCS dev split every 500 steps. Checkpoint selection criterion is the harmonic mean of BLEU and attribute correlation of the attribute(s) of interest. Metrics from the best checkpoint are reported on the AllCS test split.

B.2 Formality Classifier

For decoder-side control, we use the formality task adapter open-sourced by Krishna et al. (2020) stacked on a Hindi language adapter by Pfeiffer et al. (2020). This formality classifier is trained on English sentences from the GYAFC formality classification dataset released by Rao and Tetreault (2018). To finetune it on Hindi prefixes, we require a similar Hindi formality classification dataset. To the best of our knowledge, no such dataset is publicly available, therefore we attempt to build a synthetic dataset for our purpose.

We sample 500 sentences from the BBC Hindi News dataset[11] and mark them “formal”. This dataset contains news documents from 14 unique categories, including Science, International and Business. Similarly we sample 500 monolingual Hindi sentences from the Movie-CS split of AllCS train, and mark them “informal”. The Movie-CS split contains subtitles from 30 contemporary Bollywood movies, and is therefore assumed informal. Our formality classifier is finetuned on all prefixes of length greater than 3 tokens, derived from this dataset.

We finetune only the task adapter, keeping the language adapter and base model weights frozen. We used the official training script provided by AdapterHub[12]. Training was done for 2 epochs, with a constant learning rate of 5e-6.

C Interpolation to Unseen SPI

Held out	Bin 1 performance			Bin 2 performance		
bin	BLEU	SPI acc	SPI corr	BLEU	SPI acc	SPI corr
None	64.48	0.87	0.63	52.65	0.85	0.71
Bin 1	**62.42**	**0.71**	**0.57**	52.45	0.83	0.70
Bin 2	64.61	0.87	0.65	**50.45**	**0.75**	**0.61**

Table 10: Held out performance over SPI bins

Similar to the set up in Section 5, we hold out specific SPI bins during training, and evaluate if the scaled vector approach can successfully interpolate to the unseen bin. Results from this analysis are reported in Table 10. While the performance on the held-out bin does drop, it is by fairly small margins, thereby establishing the generalizability of the scaled vector approach.

D Human Evaluations

To evaluate CoCoa model outputs, we created a template that asks linguistic experts to assess model outputs on naturalness, semantic consistency and formality.

Every CS sentence was evaluated along each of these dimensions by 3 human raters. Annotations for a single sentence were aggregated according to a majority vote among raters. There were no instances where a majority vote could not be computed. Numbers reported in Tables 2 and 3 are aggregates reported over a set of 500 instances,

[10]From https://www.cfilt.iitb.ac.in/iitb_parallel/

[11]https://github.com/NirantK/hindi2vec/releases/tag/bbc-hindi-v0.1

[12]https://docs.adapterhub.ml/training.html#train-a-task-adapter

Monolingual Hindi	Real Code-Switched	CMI	SPI	Formality
माफ़ करना आपके घर में ऐसे ही घुस आये हम लोग (Maaf karna aapke ghar mein aise hi ghus aaye hum log)	**Sorry** आपके घर में ऐसे ही घुस आये हम लोग (Sorry aapke ghar mein aise hi ghus aaye hum log)	0.1	0.111	0.066
दोनों भाइयों के हाथ पांव रस्सियों से बांध रखे थे (Dono bhaiyo ke haath paanv rassiyo se bandh rakhe the)	दोनों भाइयों के हाथ पांव **rope** से बांध रखे थे (Dono bhaiyo ke haath paanv rope se bandh rakhe the)	0.1	0.222	0.911
लेकिन उन्होंने कहा मेरी शक्ल हुबहु किसी से मिलती है (Lekin unhone kaha meri shakal hubahu kisi se milti hai)	**But he said** मेरी शक्ल हुबहु किसीसे से मिलती है (But he said meri shakal hubahu kisi se milti hai)	0.3	0.11	0.032
जबकि भारत में यह रकम ५० फीसदी हो जाती है (Jabki Bharat mein ye rakam 40 fisadi ho jati hai)	**Whereas** भारत में यह **amount** ५० **percent** हो जाती है (Whereas Bharat mein ye amount 40 percent ho jati hai)	0.3	0.56	0.952
क्या बात है तुमने आखरी बार कब पार्टी की थी (Kya baat hai tumne aakhri baar kab party ki thi)	**wow** तुमने **last time** कब **party** की थी (Wow tumne last time kab party ki thi)	0.5	0.714	0.025
होटलों की दृष्टि से यह अमेरिका का दूसरा बड़ा शहर है (Hotelo ki drishti se ye America ka doosra bada sheher hai)	यह अमेरिका का दूसरा बड़ा शहर है **from the point of view of hotels** (Ye America ka doosra bada sheher hai from the point of view of hotels)	0.5	0.08	0.931

Figure 4: Code-switched examples from ALLCS dataset with varying scores for CMI, SPI, and formality. Romanized sentences are in parenthesis.

sampled randomly from the ALLCS test set. In the following sections, we provide exact annotation instructions, as well as relevant details about the rater pool.

D.1 Naturalness

Raters were shown a monolingual Hindi sentence, its machine-generated code-switched counterpart, and asked *How natural does the mixed language sentence sound?* They could select one of three possible options -

Very natural - The mixed language sentence sounds natural. This is how a bilingual speaker might speak. The flow between English and Hindi is spontaneous. The sentence sounds grammatically correct.

Somewhat natural - There may be some awkward construction or minor spelling/grammar errors. Someone who is just learning to speak Hindi, or a non-native speaker may communicate this way.

Not at all natural - The mixed language sentence sounds awkward and artificial. It mixes English and Hindi words in a way that is not actually spoken or written. As a result, the grammar of the mixed-language sentence is incorrect.

D.2 Semantic Consistency

Raters were shown a monolingual Hindi sentence, its machine-generated code-switched counterpart, and asked *How well does the mixed-language sentence capture the meaning of the Hindi sentence?* They could select one of three possible options -

Very well - Both sentences are exactly identical in meaning.

Somewhat well - There is some meaning overlap between the Hindi and mixed-language sentence, but they are not identical. The mixed sentence adds or drops some critical component of meaning from the Hindi sentence.

Not at all - The mixed sentence has no overlap with the Hindi sentence; the two are entirely different.

D.3 Formality

Raters were shown a monolingual Hindi sentence, two machine generated code-switched outputs, and asked *Which among the code-switched senteces is more formal?* Annotation instructions specified that formal language was more likely to be seen while reporting, talking to seniors or colleagues, and in polite conversation.

D.4 Rater Pool

Evaluations were performed by bilingual raters from India, who are fluent in Hindi and English. Every rater either held or was working towards a Diploma. Raters were paid USD 0.10 for a completed task, where a task involves evaluating one

CS text along all three dimensions.