Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

A Deep Generative Model for Code-Switched Text

Bidisha Samanta^{*1} , **Sharmila Reddy**^{*1} , **Hussain Jagirdar**^{1} , **Niloy Ganguly**^{1} and
Soumen Chakrabarti^{2}
^{1}Indian Institute of Technology, Kharagpur
^{2}Indian Institute of Technology, Bombay
{bidisha, sharmilanangi, hussainjagirdar.hj}@iitkgp.ac.in, niloy@cse.iitkgp.ac.in, soumen@cse.iitb.ac.in

Abstract

Code-switching, the interleaving of two or more languages within a sentence or discourse is pervasive in multilingual societies. Accurate language models for code-switched text are critical for NLP tasks. State-of-the-art data-intensive neural language models are difficult to train well from scarce language-labeled code-switched text. A potential solution is to use deep generative models to synthesize large volumes of realistic code-switched text. Although generative adversarial networks and variational autoencoders can synthesize plausible monolingual text from continuous latent space, they cannot adequately address code-switched text, owing to their informal style and complex interplay between the constituent languages. We introduce VACS, a novel variational autoencoder architecture specifically tailored to code-switching phenomena. VACS encodes to and decodes from a two-level hierarchical representation, which models syntactic contextual signals in the lower level, and language switching signals in the upper layer. Decoding representations sampled from prior produced well-formed, diverse code-switched sentences. Extensive experiments show that using synthetic code-switched text with natural monolingual data results in significant (33.06%) drop in perplexity.

1 Introduction

Multilingual text is very common on social media platforms like Twitter and Facebook. A prominent expression of multilingualism in informal text and speech is *code-switching*: alternating between two languages, often with one rendered in the other's character set. Many NLP tasks benefit from accurate statistical language models. Therefore, extending monolingual language models to code-switched text is important.

Many state-of-the-art monolingual models are based on recurrent neural networks (RNNs) [Chandu *et al.*, 2018; Winata *et al.*, 2018]. We call them RNN language models or RNNlms. RNNlm decoders, conditioned on task-specific features, are heavily used in machine translation [Sutskever *et al.*, 2014; Bahdanau *et al.*, 2014], image captioning [Vinyals *et al.*, 2015] textual entailment [Bowman *et al.*, 2015a] and speech recognition [Chorowski *et al.*, 2015].

Training RNNlms is data-intensive. The paucity of language-tagged code-switched text has been a major impediment to training RNNlms well. This strongly motivates the automatic generation of plausible synthetic code-switched text to train state-of-the-art neural language models.

Synthetic but realistic monolingual text generation is itself a challenging problem, on which recent deep generative techniques have made considerable progress. Two generative architectures are predominantly used: (a) Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014] and (b) Variational AutoEncoders (VAE) [Kingma and Welling, 2013]. Several recent works have successfully extended GANs [Zhang *et al.*, 2017; Kannan and Vinyals, 2017] and VAEs [Bowman *et al.*, 2015b] to generate diverse and plausible synthetic monolingual texts.

Generating plausible code-switched text is an even more delicate task than generating monolingual text. Linguistic studies show that bilingual speakers switch languages by following various complex constraints [Myers-Scotton, 1997; Muysken *et al.*, 2000] which may even include the intensity of sentiment expressed in various segments of text [Rudra *et al.*, 2016]. [Pratapa *et al.*, 2018] synthesized code-mixed sentences by leveraging linguistic constraints arising from Equivalence Constraint Theory. While this works well for language pairs with good structural correspondence (like English-Spanish), we observe performance degrades with weaker correspondence (like English-Hindi). [Bidisha *et al.*, 2019] proposes a method to generate code-switched text given a source and target sentence pair, with a restrictive set of switching patterns. Therefore, a code-switched text synthesizer needs to learn overall syntax distributions of code-switched sentences, as well as model complex switching patterns conditioned on it.

Owing to its great syntactic and switching diversity, large volumes of language-labeled code-switched text is needed to train monolingual deep generative models, which are not available. The only alternative is to train monolingual models with parallel corpora of the two constituent languages which may be relatively easily obtainable. However, training a GAN with aligned parallel corpora may not help, because it is designed to generate a sentence from a noise dis-

---

[*] Both the authors contributed equally

tribution instead of any learned latent embedding space. Using a VAE RNNlm [Bowman *et al.*, 2015b] is more promising. Aligned parallel corpora are expected to yield similar representations for a source-target sentence pair. Therefore, a VAE decoder conditioned on this embedding may generate some code-switch text without applying explicit external force. However, it is unlikely to learn subtle connections between context and switching decisions as well as a customized VAE solution, which is our goal.

Here we present VACS, a new deep generator of code-switched text, based on a hierarchical VAE augmented with language- and syntax-informed switching components.

- Observed language-labeled code-switched text is encoded to a two-layer compressed representation. The lower layer encodes sequential word context. Conditioned on this lower layer, the encoder models the switching behavior in the higher layer. This contrasts with existing dual-RNNlm architectures [Garg *et al.*, 2018] that do not have any explicit gadget to model the switching behavior. Our encoder learns the two-layer representation via variational inference so that the resulting encoded representations enable our decoder to readily generate new code-mixed sentences.
- Our decoder is designed to sample a context sequence, given a switching pattern. Unlike previous RNNlms [Chandu *et al.*, 2018] which consider context and tag generation as independent processes, the decoder of VACS first decodes a switching pattern from the switching embedding and then uses this switching pattern memory as well as the lower-layer compressed encoding, to generate a sequence of words in a restrictive fashion. context sequence. The
- During the decoding process, VACS (trivially) generates the language labels for each word in the sentence. Thus, VACS lets us synthesize unlimited amounts of labeled code-switched text, starting with modest-sized samples.

Owing to the asymmetry between word and label sequences, our encoder and decoder layers show some asymmetries tailored to code-switching, which distinguishes VACS from a regular RNN-based VAE.

Through extensive experiments reported here, we establish that augmenting scarce natural labeled code-switched text with plentiful synthetic code-switched text generated by VACS significantly improves the perplexity of state-of-the-art language models. The perplexity of the models on held-out natural Hindi-English text drops by 33.32% compared to using only natural training data. Manual inspection also shows that VACS generates sentences with diverse mixing patterns.[1]

2 Background on VAEs

VAEs [Kingma and Welling, 2013] are among the most popular deep generative models. They define a **decoding** probability distribution $p_\theta(x|z)$ to generate observation x, given latent variables z, which are sampled from a simple **prior** distribution $p_\pi(z)$. The objective of the VAE is to learn an approximate probabilistic inference model $q_\phi(z|x)$ that **encodes** latent factors or features z of the variation in the observed data x.

Distributions p and q are often parameterized using deep neural networks. We use the maximum likelihood principle to train a VAE, i.e., maximize the expected lower bound of the likelihood on observed data $x \sim D$:

$$\max_\phi \max_\theta \mathbb{E}_D \Big[\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \mathrm{KL}\big(q_\phi(z|x) || p_\pi(z)\big) \Big]$$

To represent more complex features in the latent space, multiple VAEs are stacked hierarchically [Rezende *et al.*, 2014; Sønderby *et al.*, 2016]. The stack of latent variables Z are designed to learn a "feature hierarchy". For a hierarchical VAE with Λ layers, the prior, encoding and decoding probability distributions are modeled as below:

$$\begin{aligned} \text{Encoder:} \quad & q_\phi(Z|x) = q_\phi(z_1|x) \textstyle\prod_{\lambda=2}^{\Lambda} q_\phi(z_\lambda|z_{\lambda-1}) \\ \text{Prior:} \quad & p_\pi(Z) = p(z_\Lambda) \textstyle\prod_{\lambda=1}^{\Lambda-1} p_\theta(z_\lambda|z_{\lambda+1}) \qquad (1) \\ \text{Decoder:} \quad & p_\theta(x|Z) = p_\theta(x|z_1) \end{aligned}$$

The performance of the above scheme is sensitive to the design of the layers. Layers $\lambda \gg 1$ may fail to capture extra information. Excessively deep hierarchies with large Λ may lead to training difficulties [Sønderby *et al.*, 2016].

3 VACS: A VAE for Code-switched Text

This section gives a high-level overview of VACS, followed by details of the building blocks, highlighting key advances beyond prior art. Our focus will be on components that implement a context-based switching distribution. Later, we describe the training process and other implementation details.

3.1 Overview

We aim to design a VAE for code-switched text, which, once trained on a collection of tagged code-mixed text should be able to generate new code-mixed text from the same vocabulary. We represent a code-switched sentence S as $\{(w_i, y_i) : i = 1, \dots, |S|\}$, where (w_i, y_i) is a pair comprised of a word w_i at position i and the corresponding language label y_i to which it belongs. Here we consider the simple case of switching between two languages, a source language s and a target language t. Let `SOS`, `EOS` denote start and end of sentence markers. Generation of output stops when label `EOS` is generated. We let y_i take values from $\{s, t, \texttt{EOS}\}$. When discrete values like w_i, y_i are input to networks, they are one-hot encoded. VACS is characterized by these components:

Prior: $p_\pi(Z)$

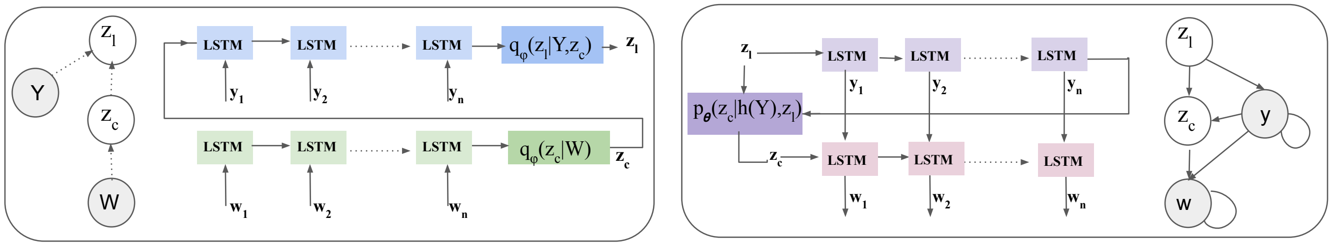
Inference model (encoder): $q_\phi(Z|\boldsymbol{W}, \boldsymbol{Y})$

Generative model (decoder): $p_\theta(\boldsymbol{W}, \boldsymbol{Y}|Z)$

In our formulation, Z will consist of two latent encoded representations z_l and z_c. Here z_l is the representation of language-switching behavior, which is generated conditioned on the context representation z_c, which captures syntactic and structural properties of a sentence. $\boldsymbol{W}$ is the observed sequence of words and $\boldsymbol{Y}$ is the corresponding label sequence. Given our objective, a hierarchical VAE architecture is adapted for the basic formulation with suitable departures

---

[1] https://github.com/bidishasamantakgp/VACS



(a) Encoder (inference)

(b) Decoder (generation)

Figure 1: The encoder and decoder in VACS. (a) Graphical model and the recurrent architecture of the encoder. (b) Graphical model and recurrent architecture of the decoder.

whenever required. The next subsections will cover in details the inference model, generative model and prior.

3.2 Encoder

Given observed labeled sentence $(\boldsymbol{W}, \boldsymbol{Y})$, our inference model q_ϕ defines a hierarchical probabilistic encoding $Z = (z_c, z_l)$ by first learning the content, structural embedding z_c of the entire sentence. Using this embedding z_c along with sequentially learned language label information the inference model q_ϕ encodes the latent switching pattern embedding z_l. Figure 1 (a) illustrates the encoder. We use two distinct RNN (LSTM) cells in the encoder, $r_{q,c}$ and $r_{q,l}$. Their corresponding recurrent states are denoted $h^{(q,c)}$ and $h^{(q,l)}$. Input token positions are indexed as $i = 0, 1, \ldots, I$. The recurrence to estimate z_c goes like this.

$$\text{We initialize} \quad h_0^{(q,c)} = \vec{0} \tag{2}$$

$$\text{For } i = 1, \ldots, I\text{:} \quad h_i^{(q,c)} = r_{q,c}(w_i, h_{i-1}^{(q,c)}) \tag{3}$$

$$\text{Finally,} \quad [\mu_{q,c}, \sigma_{q,c}] = f_{q,c}(h_I^{(q,c)}) \tag{4}$$

$$\text{and then} \quad z_c \sim q_\phi(z_c|\boldsymbol{W}) = \mathcal{N}(\mu_{q,c}, \text{diag}(\sigma_{q,c}^2)) \tag{5}$$

Next we estimate the encoding z_l.

$$\text{We initialize} \quad h_0^{(q,l)} = z_c \tag{6}$$

$$\text{For } i = 1, \ldots, I\text{:} \quad h_i^{(q,l)} = r_{q,l}(y_i, h_{i-1}^{(q,l)}) \tag{7}$$

$$\text{Finally,} \quad [\mu_{q,l}, \sigma_{q,l}] = f_{q,l}(h_I^{(q,l)}) \tag{8}$$

$$\text{and then} \quad z_l \sim q_\phi(z_l|z_c, \boldsymbol{Y}) = \mathcal{N}(\mu_{q,l}, \text{diag}(\sigma_{q,l}^2)). \tag{9}$$

$$\text{Overall,} \quad q_\phi(Z|\boldsymbol{W}, \boldsymbol{Y}) = q_\phi(z_c|\boldsymbol{W}) q_\phi(z_l|z_c, \boldsymbol{Y})$$

Here, $\mu_{q,c}, \sigma_{q,c}$ are the mean and standard deviation for the context encoding and $\mu_{q,l}, \sigma_{q,l}$ are the mean and standard deviation for the switching behavior encoding distribution. $\mathcal{N}$ denotes normal distribution. diag$(\cdot)$ represents a diagonal covariance matrix. $f_{q,c}, f_{q,l}$ are modeled as feed forward stages, $r_{q,c}, r_{q,l}$ are designed as recurrent units. We use the subscript q (or p) to highlight if it belongs to encoder (or decoder).

Summarizing the distinction from traditional hierarchical VAE, VACS's inference module accepts inputs in both encoding layers: word sequence at the ground layer and language label sequence at the upper layer. Learning a sequence model over language labels becomes difficult (even with hierarchical encoding) if we provide both the inputs only in the lowest level, possibly by concatenating suitable embeddings [Winata *et al.*, 2018].

3.3 Decoder

Starting from $Z = (z_l, z_c)$, our probabilistic decoder generates synthetic code-switched text with per-token language ID labels, using a two-level hierarchy of latent encoding. However, unlike the conventional hierarchical variational decoder, VACS decodes a switching pattern given z_l at the upper level, then conditioned on z_l and the decoded tag history it generates a content distribution z_c. Here we need to design a specific decoupling mechanism of z_c from z_l, which is not just the reverse of encoding technique. As z_l has the switching information as well as the context information, we use both z_l and $h(y)$ which is the history of label decoding to decode the distribution of z_c. We design the loss in such a way that tries to minimize the difference between encoding and decoding distribution of z_c.

We use two distinct RNN (LSTM) cells in the decoder, $r_{p,l}$ and $r_{p,c}$. Their corresponding recurrent states are denoted $h^{(p,l)}$ and $h^{(p,c)}$. Output token positions are indexed $o = 1, \ldots, O$. The feedforward network to convert $h_o^{(p,l)}$ to a multinomial distribution over y_o is called $f_{p,l}$.

$$\text{We initialize} \quad h_0^{(p,l)} = z_l \text{ and } y_0 = \texttt{SOS} \tag{10}$$

$$\text{For } o = 1, \ldots, O\text{:} \quad h_o^{(p,l)} = r_{p,l}(y_{o-1}, h_{o-1}^{(p,l)}) \tag{11}$$

$$y_o \sim \text{Multi}(f_{p,l}(h_o^{(p,l)})) \tag{12}$$

Decoding continues until some $y_O = \texttt{EOS}$ is sampled at some O, and then stops. Effectively this amounts to sampling from $p_\theta(\boldsymbol{Y}|z_l)$. Once all labels $\boldsymbol{Y} = y_1, \ldots, y_O$ are generated, we decode z_c and start generating words $w_1, \ldots, w_O$. $f_{p,c}$ denotes the feedforward network to decode z_c as follows.

$$[\mu_{p,c}, \sigma_{p,c}] = f_{p,c}(h_O^{p,l}, z_l) \tag{13}$$

$$z_c \sim p_\theta(z_c|z_l, h_O^{(p,l)}) = \mathcal{N}(\mu_{p,c}, \text{diag}(\sigma_{p,c}^2)) \tag{14}$$

The feedforward network $f_{p,w}$ converts $h_o^{(p,c)}$ to a multinomial distribution over words from the languages indicated by

$y_1, \ldots, y_O$.

$$\text{We initialize} \quad h_0^{(p,c)} = z_c \quad \text{and} \quad w_0 = \texttt{SOS} \tag{15}$$

$$\text{For } o = 1, \ldots, O\text{:} \quad h_o^{(p,c)} = r_{p,c}(w_{o-1}, h_{o-1}^{(p,c)}) \tag{16}$$

$$w_o \sim \text{Multi}(f_{p,w}(h_o^{(p,c)}, y_o)) \tag{17}$$

If $y_o = s$, $f_{p,c}$ returns a multinomial distribution over the source vocabulary, and if $y_o = t$, $f_{p,c}$ returns a multinomial distribution over the target vocabulary. Effectively, we have sampled $\boldsymbol{W}$ from the distribution $p_\theta(\boldsymbol{W}|\boldsymbol{Y}, z_c)$. Overall, decoding amounts to sampling from $p_\theta(\boldsymbol{Y}, \boldsymbol{W}|Z) = p_\theta(\boldsymbol{Y}|z_l)\, p_\theta(\boldsymbol{W}|\boldsymbol{Y}, z_c)$.

Figure 1 (b) illustrates the decoder architecture. VACS departs from existing dual RNN architectures [Garg *et al.*, 2018] (two RNNs dedicated to s and t) in the following two ways:

- Instead of using a softmax output of two decoding RNNs corresponding to two language generators, VACS learns to decode language labels explicitly from a latent space.
- By using tightly-coupled decoding RNNs, parameter learning in VACS becomes more effective.

This way the decoder can generate word sequence in a more controlled fashion. The recursive word decoding unit generates a word given the predicted label from the *language ID* decoding layer.

3.4 Prior

The latent variable z_l can be sampled from the standard normal distribution:

$$p_\pi(z_l) \sim \mathcal{N}(\mathbf{0}, \mathbb{I}) \tag{18}$$

and then reuse $p_\theta(z_c|z_l)$ to define

$$p_\pi(Z) = p_\pi(z_l)\, p_\theta(z_c|z_l) \tag{19}$$

3.5 Training

Given a collection of M code-switched text $S^{(m)} = (\boldsymbol{W}^{(m)}, \boldsymbol{Y}^{(m)}) : m = 1, \ldots, M$, we train our model by maximizing the evidence lower bound (ELBO), as described in Section 2. In our case, after taking into consideration the dependence between z_c and z_l, the ELBO can be simplified to:

$$\begin{aligned}\max_{\phi,\theta} \sum_{m\in[M]} \Big[& \mathbb{E}_{q_\phi(Z^{(m)}|\boldsymbol{W}^{(m)},\boldsymbol{Y}^{(m)})} \log p_\theta(\boldsymbol{W}^{(m)}, \boldsymbol{Y}^{(m)}|Z^{(m)}) \\ & - \mathbb{E}_{q_\phi(z_c|\boldsymbol{W}^{(m)})} \text{KL}(q_\phi(z_l|z_c, \boldsymbol{Y}^{(m)})||p_\pi(z_l)) \\ & - \mathbb{E}_{q_\phi(z_c|\boldsymbol{W}^{(m)}), p_\theta(z_c|z_l)} \text{KL}(q_\phi(z_c)||p_\theta(z_c|z_l)) \Big]\end{aligned} \tag{20}$$

Because human-labeled code-mixed text is scarce, we first train VACS with the parallel corpora specified in Section 4.1, with aligned word embeddings. Then we further tune model parameters using real code-switched data, also specified in Section 4.1. We used Adam optimiser and KL cost annealing technique [Bowman *et al.*, 2015b] to train VACS.

4 Experimental Setup

Here, we describe the labeled data sets, baseline paradigms, and evaluation criteria, followed by the description of language models used to evaluate the utility of the synthesized

4.1 Datasets to Train Generative Models

To train the generative models, we use a subset of the (real) Hindi-English tweets collected by [Patro *et al.*, 2017] and automatically language-tagged by [Rijhwani *et al.*, 2017] with reasonable accuracy. From this set we sample 6K tweets where code-switching is present, which we collect into folds **rCS-train**, **rCS-valid**. 5K tweets are found labeled with only one language. These monolingual instances are converted into parallel corpora by translating Hindi sentences to English and vise versa using Google Translation API[2], generating 10K instances. The word embeddings of the two languages are aligned. We used different charset for Hindi (Devnagari) and English (Roman) .

4.2 Baseline Generative Models

Deep generative models. To understand the difficulties of extending existing monolingual text generators to code-switched text, we design four baselines from two state-of-the-art generative models. [Bowman *et al.*, 2015b] showed impressive results at generating monolingual sentences from a continuous latent space. They extended RNNlms with a variational inference mechanism. However, their model does not allow inclusion of hand crafted features like language ID, POS tag etc. Meanwhile, [Zhang *et al.*, 2017] proposed a GAN model to generate a diverse set of sentences. Based on these, our baseline approaches are:

pVAE: We train the network of [Bowman *et al.*, 2015b] with the parallel corpora. The probability of generating a word is designed as a softmax over the union vocabulary. As both of the corpora are mapped to the same latent space due to aligned embeddings, we expect the model to switch language whenever it finds a word from the other language more probable than a word in the same language as the current word.

rVAE: To further make the model learn specific switching behaviors we train the model with the real code mixed text along with the parallel data.

pGAN: Similar to pVAE, we train the network proposed by [Zhang *et al.*, 2017] with the parallel text corpora.

rGAN: GAN trained with the real code-switched data along with the parallel corpora.

RNNlm based generative models. Though language models are built primarily to estimate the likelihood of a given sentence, they can also be used as a generative tool. Recently, RNN based language models have been used to generate code-switched text as well, giving significant perplexity reduction compared to generic language models that do not consider features specific to code-switching. We compare VACS against the following code-switched LMs:

aLM: We train the system proposed by [Winata *et al.*, 2018] using the real code-switched text and then use a word decoder and language decoder to generate synthetic texts.

---

[2] https://translation.googleapis.com

bLM: After training the system proposed by [Chandu *et al.*, 2018] with the real code-mixed text, we use their LSTM decoder to generate synthetic text.

4.3 Direct/Intrinsic Evaluation

Here we analyse the features like length distribution and diversity of code-switching of generated synthetic texts. We also measured one sentence level metric **Code-Mixing Index (CMI)** coined by [Gambäck and Das, 2016], and three corpus level metrics **Multilingual index (M-Index)**, **Burstiness** and **Span Entropy** that were introduced in [Guzmán *et al.*, 2017] to demonstrate how different the generated texts are from the training corpus in terms of switching.

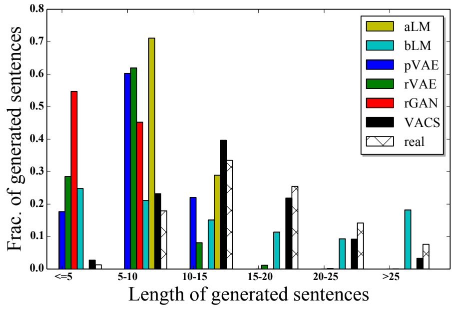


Figure 2: Length distribution of the generated sentences from different methods. VACS generates closest length distribution.

4.4 Indirect/Extrinsic Evaluation

We will use prior methods and VACS to generate large volumes of code-switched text. These will be used to train a *payload* language model (as distinct from the *generative* model of VACS and baselines) — specifically, the character-level LSTM proposed by [Kim *et al.*, 2016]. Each training corpus will result in a trained payload model. The various payload models will then be used to calculate perplexity [Brown *et al.*, 1992] scores on a held-out natural code-switched corpus. The assumption is that the payload model with the smallest perplexity was trained by the 'best' synthetic text.

Training curricula. [Baheti *et al.*, 2017] show that language models perform better when trained with an interleaved *curriculum* of monolingual text from both the participating languages, then ending with code-switched (CS) text, rather than randomly mixing them. We build the curriculum from the following corpora:

Mono: 2K monolingual Hindi and 2K monolingual English tweets were sampled from the dataset. We translated Hindi to English and vice versa and make a set of 8K tweets.

X-gCS: This is the generated synthetic data. We sampled 5K generated synthetic code-switched text from various generative models. Here **X** denotes the generative method, which is one of {**pVAE, rVAE, pGAN, rGAN, aLM, bLM, VACS**}.

The specific curricula we use are:

Dataset	Avg CMI	M-index	Burstiness	Span Entropy
rCS-train	0.56	0.778	0.232	1.498
pVAE-gCS	-0.20	+0.108	+0.081	-0.527
rVAE-gCS	+0.01	+0.216	+0.058	-0.375
VACS-gCS	+0.08	+0.078	+0.065	-0.192
aLM-gCS	+0.12	+0.201	+0.036	-0.299
bLM-gCS	+0.14	+0.155	+0.023	-0.287
rGAN-gCS	+0.17	+0.219	+0.081	-0.622

Table 1: Intrinsic evaluation of real and generated CS.

Mono, which uses no synthetic data.

gCS | Mono, first synthetic then parallel monolingual.

Mono | gCS, first parallel monolingual, then synthetic.

Here $C_1|C_2$ denotes the sequence of corpora used to train the language model. Designing multi-task losses to guard against catastrophic forgetting is left for future work.

Validation and Testing. We sample 7K instances from the original real code-switched pool for validation and 7K for testing. These are considered as scarce evaluation resources and not used in payload training.

5 Results and Analysis

In Section 5.1 we compare intrinsic properties of synthetic texts generated from various models. In Section 5.2 we compare perplexities of payload language models prepared from text synthesized by various generators. Finally, in Section 5.3, we present anecdotes about generated text and its quality. pGAN fails to generate any appreciable rate of code-switching. Therefore, we refrain from considering it further.

5.1 Intrinsic Properties of Synthesized Text

Based on 5000 synthetic sentences sampled from different generative methods, we report the following properties.

Length. We investigate the quality of generation methods in terms of variation in length and diversity. Other than bLM, all baselines tend to produce sentences shorter than 15 words. But Figure 2 shows that real sentences have average length $\sim$16 and may be as long as 25 words. rGAN generates very short sentences, at most 5 words long, and pVAE and rVAE generate sentences with an average length of $\sim$10. For aLM and bLM average lengths are $\sim$10 and $\sim$12 respectively. VACS has a mean of $\sim$17 and follows the distribution of real code-switched data most closely.

CMI, M-index, Burstiness, Span-Entropy. We report the metric values of the generated corpus and the real corpus in Table 1. VACS is closest to the real corpus in terms of M-index and Span entropy, which indicates the ratio different language tokens in the generated sentences and language span distribution is closer to the real data. Though VACS produces a larger CMI and burstiness as it can produce sentences of different lengths and various switching patterns; its CMI is still smaller than GAN, aLM, and bLM and burstiness smaller than GAN and pVAE. GAN generates the highest CMI and burstiness indicating haphazard switching patterns. On the other hand, pVAE produces lowest CMI and

	Training Curricula	Valid PPL	Test PPL
1	Mono	3034.251	3123.827
2a	aLM-gCS | Mono	3094.998	3179.039
	bLM-gCS | Mono	3051.042	3123.510
	rGAN-gCS | Mono	3206.175	3298.085
	pVAE-gCS | Mono	2839.840	2899.397
	rVAE-gCS | Mono	2383.426	2337.617
	VACS-gCS | Mono	2243.578	2296.533
2b	Mono | aLM-gCS	3083.314	3189.905
	Mono | bLM-gCS	2829.149	2896.337
	Mono | rGAN-gCS	3015.820	3069.263
	Mono | pVAE-gCS	2807.296	2869.633
	Mono | rVAE-gCS	2418.342	2493.023
	Mono | VACS-gCS	2081.774	2090.781

Table 2: Perplexity of payload language model using different training curricula. VACS achieves the lowest perplexity. Green: lower perplexity than Mono baseline; yellow and red: larger perplexity than Mono baseline (gray).

span entropy indicating that the generated sentences are "almost monolingual" or the language spans are equal in length. rVAE produces CMI very close to real and less diverse in terms of both switching and length distribution. Along with generating Burstiness with various switching patterns GAN also has highest burstiness

5.2 Extrinsic Perplexity

Table 2 provides a comparative study on the perplexity achieved on real validation and test CS text, after training a payload language model with different curricula spanning parallel monolingual (Mono) and generated CS (gCS) text.

Obviously, a payload language model that has seen only monolingual text when training will have large perplexity on held-out real CS text, which shows a diversity of switching behavior, in terms of both syntax structure near switches and the distribution of words used in switched segments. We expect that, in the absence of real CS text adequate to train the payload model, large volumes of synthetic text will help.

Surprisingly, this does not happen for aLM-gCS, bLM-gCS, and rGAN-gCS. Adding these texts to the monolingual baseline makes payload perplexity generally *worse* and much worse in some cases, in particular, rGAN-gCS. VAE has better success. For the gCS|Mono curriculum, pVAE-gCS improves upon the baseline, but rVAE-gCS does worse. On further investigation we found that, pVAE generates $\sim$80% monolingual data, this contributes to the monolingual corpus which makes the training more coherent than mixing with code-switched data with low quality. For the Mono|gCS curriculum, both pVAE and rVAE perform worse than Mono.

In sharp contrast, VACS-gCS achieves the best (smallest) perplexity in both curricula, and much smaller than the Mono baseline. This shows that synthetic text from VACS can be used effectively to supplement parallel monolingual corpora. pVAE is the second best choice.

GAN-based synthetic text performs poorly. pGAN fails to generate any plausible code-switched text as it does not get any real code-switch samples from the parallel corpora. rGAN performance is also worse than other generative models. Though [Zhang *et al.*, 2017] avoided mode collapsing problems common in GANs, we observed that the problem prevailed for longer sentences (>10 words) when trying to train with small amounts of code-switched text. The problem persists because CS text is much more syntactically diverse than monolingual corpus, so training a GAN with a small number of real samples produces sub-optimal results.

	Sentences
a	hara gaya pakistan hamen logon ke tweet rato karane (Pakistan defeats us to stop people from tweeting) apane logon ko batting upalabdh series ke (Batting series available to our people) cricket run banaake kiya SA ke haar (Defeated SA by making runs in cricket)
b	ladakiyon 20 assembly pratishat se (Girls from 20 assembly percent) vichaar bhee bikree that is 25 guna assembly ka (Justice is also sold, that is 25 times assembly) assembly against asia pradesh irfaan teesree har breaking (Assembly against asia irfaan's third deafeat was breaking)

Table 3: Sentences synthesized by VACS. Each row corresponds to sentences sampled from a fixed context representation. The Blue segments are in Hindi. Green: English translation of CS text.

The performance of aLM and bLM, while better than GANs, is far from VACS. These LMs are designed explicitly for code-switched languages and require language-tagged data. Hence the generative power of such models strongly depends on the size and quality of tagged training data available.

5.3 Sample Synthetic Sentences

Table 3 shows sentences generated by VACS. All sentences in a row block are sampled from the same context embedding z_c, and each row corresponds to a different z_l. Note that the generated sentences seem to be able to produce a similar context. Like row (a) corresponds to cricket and (b) to assembly. It learns to produce meaningful phrases most of the cases which seem reasonable syntactically; however, semantics and pragmatics are not realistic, like in monolingual synthesis.

6 Conclusion

We proposed VACS, a novel variational autoencoder, to synthesize unlimited volumes of language-tagged code-switched text starting with modest real code-switched and abundant monolingual text. We showed that VACS generates text of various lengths and switching pattern. We also showed that synthetic code-switched text produced by VACS can help train a language model which then has low perplexity on real code-switched text. We further demonstrated that we can generate reasonable syntactically valid sentences with language-tags which can be used for various downstream applications like language labeling, POS tagging, NER etc.

Acknowledgments

B. Samanta is funded by Google India Ph.D. fellowship and the "Learning Representations from Network Data" project sponsored by Intel. We would like to thank Dr. Dipanjan Das and Dr. Dan Garrette for their valuable inputs.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Baheti *et al.*, 2017] Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. *Proceedings of ICON*, 2017.

[Bidisha *et al.*, 2019] Bidisha, Niloy Samanta, Soumen Ganguly, and Chakrabarti. Improved sentiment detection via label transfer from monolingual to synthetic code-switched text. 2019.

[Bowman *et al.*, 2015a] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[Bowman *et al.*, 2015b] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, and Dai. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[Brown *et al.*, 1992] Peter F Brown, Vincent J Della Pietra, Robert L Mercer, Stephen A Della Pietra, and Jennifer C Lai. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1), 1992.

[Chandu *et al.*, 2018] Khyathi Chandu, Thomas Manzini, Sumeet Singh, and Alan W Black. Language informed modeling of code-switched text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, 2018.

[Chorowski *et al.*, 2015] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *NIPS*, 2015.

[Gambäck and Das, 2016] Björn Gambäck and Amitava Das. Comparing the level of code-switching in corpora. In *LREC*, 2016.

[Garg *et al.*, 2018] Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. Code-switched language models using dual rnns and same-source pretraining. *arXiv preprint arXiv:1809.01962*, 2018.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[Guzmán *et al.*, 2017] Gualberto A Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. Metrics for modeling code-switching across corpora. In *INTERSPEECH*, 2017.

[Kannan and Vinyals, 2017] Anjuli Kannan and Oriol Vinyals. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*, 2017.

[Kim *et al.*, 2016] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, 2016.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[Muysken *et al.*, 2000] Pieter Muysken, Carmen Pena Díaz, Pieter Cornelis Muysken, et al. *Bilingual speech: A typology of code-mixing*, volume 11. Cambridge University Press, 2000.

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

[Patro *et al.*, 2017] Jasabanta Patro, Bidisha Samanta, Saurabh Singh, Abhipsa Basu, Prithwish Mukherjee, Monojit Choudhury, and Animesh Mukherjee. All that is English may be Hindi: Enhancing language identification through automatic ranking of the likeliness of word borrowing in social media. In *EMNLP Conference*, 2017.

[Pratapa *et al.*, 2018] Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *ACL Conference*, 2018.

[Rezende *et al.*, 2014] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[Rijhwani *et al.*, 2017] Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *ACL Conference*, volume 1, 2017.

[Rudra *et al.*, 2016] Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *EMNLP Conference*, 2016.

[Sønderby *et al.*, 2016] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *NIPS*, 2016.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[Vinyals *et al.*, 2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE CVPR*, 2015.

[Winata *et al.*, 2018] Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. Code-switching language modeling using syntax-aware multi-task learning. *arXiv preprint arXiv:1805.12070*, 2018.

[Zhang *et al.*, 2017] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.