Training Data Augmentation for Code-Mixed Translation

Abhirut Gupta
Google Research
abhirut@google.com

Aditya Vavre and **Sunita Sarawagi**
IIT Bombay
{adityavavre,sunita}@cse.iitb.ac.in

Abstract

Machine translation of user-generated code-mixed inputs to English is of crucial importance in applications like web search and targeted advertising. We address the scarcity of parallel training data for training such models by designing a strategy of converting existing non-code-mixed parallel data sources to code-mixed parallel data. We present an mBERT based procedure whose core learnable component is a ternary sequence labeling model, that can be trained with a limited code-mixed corpus alone. We show a 5.8 point increase in BLEU on heavily code-mixed sentences by training a translation model using our data augmentation strategy on an Hindi-English code-mixed translation task.

1 Introduction

Code-mixing (CM), the phenomenon of mixing words from two languages in a sentence, is getting increasingly commonplace in several bilingual communities[1]. Recently, much research has focused on training language models over code-switched data for tasks like automatic speech recognition (ASR) (Winata et al., 2019; Gonen and Goldberg, 2019). In this paper we focus on the less explored problem of translating code-switched inputs to a high-resource language like English. This task is compelling in applications like Web search, targeted advertising, and recommendations, which require matching user-generated code-mixed queries to rich English content.

A major challenge in such applications is the lack of parallel data from code-mixed input to English. While years of effort have made available rich parallel datasets for translation, these are mostly over formal sources like news, which tend to be less code-mixed. In this paper we show how to create high-quality parallel data for training a code-mixed translation model by exploiting three types of resources: 1) Parallel data from non-code-mixed sentences to English, 2) Code-mixed sentences, and 3) Monolingual sentences in English.

Contributions: (1) We present an mBERT (Devlin et al., 2019) based procedure for converting non-CM parallel data to CM parallel data. The core learnable component of our procedure requires fine-tuning mBERT for a three-way sequence labeling task, and can be easily trained using the limited code-switched sentences alone. We apply this model to convert source sentences of the parallel data to code-mixed sentences, while keeping the target English sentences in-tact. We also extend the existing back-translation method of using monolingual target data, with our code-switched augmentation. (2) We experiment on a rich public code-mixed dataset obtained from a literacy promotion project. We show that with our data augmentation strategy the translation BLEU improves from 43.9 to 46.4 overall. On sentences that are more heavily code-mixed our accuracy increases by 5.8 BLEU points, and on an adversarial test set where the baseline provides poor accuracy we show a 5.4 point BLEU increase. (3) We show that our data augmentation strategy improves performance for code-switched test sets while maintaining state of the art performance on non-code-switched inputs.

2 Related Work

Most prior work on CM has focused on training a language model (LM) in the context of automatic speech recognition. The main challenge addressed in these works is the limited availability of code-mixed sentences. Gonen and Goldberg (2019) and Lee and Li (2020) propose different methods of training LMs for CM sentences without explicitly creating synthetic CM data, but another popular strategy is to first create synthetic CM data and train the LM with such synthetic data. We next summarize existing approaches to generate syn-

[1]https://github.com/gentaiscool/code-switching-papers

thetic CM data:

Chang et al. (2019) propose to learn switching patterns from code-mixed data using a GAN-based adversarial training. Gao et al. (2019) use BERT as the generator which is fine-tuned by masking the English words in a CM corpus using GAN-based adversarial training. In contrast, ours is a much simpler sequence labeling formulation. Taneja et al. (2019) proposes to splice fragments chosen from monolingual corpus of two languages based on statistics of length distribution and phone transition. Pratapa et al. (2018) use Equivalence Constraint Theory to define rules on top of parse trees of two sentences to create grammatically valid artificial CM data. Samanta et al. (2019) use a variational autoencoder to generate synthetic CS data.

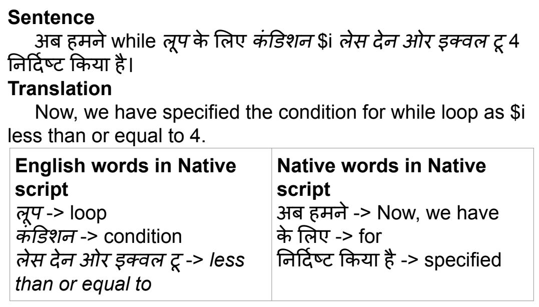
Winata et al. (2019) propose a sequence-to-sequence model using a copy mechanism that learns when to switch from one language to another. To train their model they depended on high-resource commercial translation models for translating the code-mixed input to monolingual sentences in both English and native language. Since our goal is to train such a translation model, we did not want to depend on such resources.

One key difference is that our goal is translation from code-mixed to English, and not designing a representative LM for code-mixed data. Since our final target is English, when designing our data augmentation strategy we give higher priority to preserving the distribution of the target-side (English) than the CM input.

3 Our Approach

Our strategy is to augment the training data by converting existing non-code-mixed parallel corpus into a parallel corpus with code-mixed source. Let $\mathcal{L}, \mathcal{M}, \mathcal{E}$ denote the space of non-code-mixed sentences, code-mixed sentences, and English sentences respectively. We have available a parallel corpus of non-code-mixed and English pairs $(L, E) \subset (\mathcal{L}, \mathcal{E})$, a code-mixed corpus $M \subset \mathcal{M}$, and a monolingual corpus $E_M \subset \mathcal{E}$. Our goal is to train a translation model from code-mixed input to English $T : \mathcal{M} \mapsto \mathcal{E}$.

Our focus is Indic languages such as Hindi that are often code-mixed with English. The code-mixed corpus in our case contains three kinds of tokens: native tokens in native script e.g. Devanagari, English tokens in Latin script, and English tokens transliterated to native script. Figure 1 shows an example sentence pair from our code-mixed test corpus with these different types of tokens. In Table 1 we present statistics of tokens of the three types in a parallel Hindi-English corpus and our test code-mixed corpus. Given the huge gap in the fraction of En tokens in the original parallel data, we propose methods for synthetic data creation that perturb non-code-mixed source sentences in the parallel data to a code-mixed sentence. For this we train a model $F : (\mathcal{L}, \mathcal{E}) \mapsto (\mathcal{M}, \mathcal{E})$ for converting sentences in the native language to their code mixed forms using the parallel English sentence.



Sentence

अब हमने while *लूप* के लिए *कंडिशन* $i *लेस देन ओर इक्वल टू* 4 निर्दिष्ट किया है।

Translation

Now, we have specified the condition for while loop as $i less than or equal to 4.

English words in Native script	**Native words in Native script**
लूप -> loop *कंडिशन* -> condition *लेस देन ओर इक्वल टू* -> *less than or equal to*	अब हमने -> Now, we have के लिए -> for निर्दिष्ट किया है -> specified

Figure 1: An example sentence pair from Spoken Tutorial which illustrates the different token types in code-mixed source sentences. In addition to Hindi tokens written in Devanagari script, sentences can contain English words in Latin script (like *while*) and English words written in Devanagari script (like *loop* and *condition*).

Dataset	En	En-Trans
IITB Parallel Train	0.021	0.132
Code-Mixed Test (Hi)	0.121	0.121

Table 1: Fraction of English (En) and English transliterated to Devanagari (En-Trans) tokens in Hi-En parallel dataset and our code-mixed Hindi test set.

Our model is based on mBERT, and consists of two phases: the first phase predicts words to switch in a monolingual sentence, and the second phase generates the switched words by harnessing parallel data. We describe these phases next:

3.1 Predict Code-Mixed Patterns in Monolingual Sentences

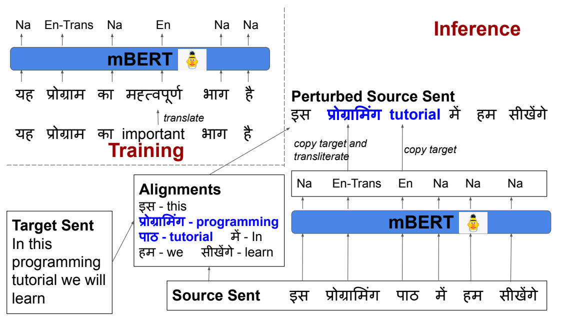
We train an m-BERT based sequence-labeling task that takes as input a monolingual sentence and predicts tokens that should be translated to the other language to produce a natural sounding code-mixed sentence. For training data, we use the small amount of code-mixed data $M \in \mathcal{M}$. These sentences are first labeled with word level Language

IDs, using Zhang et al. (2018). The langId tool assigns three types of labels:{En, En-Trans, Na} where En-Trans refers to English words written in native script, and Na refers to native words in native script. We then generate a synthetic monolingual sentence from each code mixed sentence $\mathbf{z} \in M$ by replacing all words in $\mathbf{z}$ with the label 'En' to their translations in the source language and script. Words that are predicted 'En-Trans' (English words written in Native script) are transliterated[2] to Latin and then translated only a fraction f of the time. Since sentences in $\mathcal{L}$ comprising the parallel data also contain transliterated English words in our corpus, we choose f so as to account for the difference in transliterated English words between native and code-mixed. The resulting sentence $\mathbf{z}'$ is treated as being from $\mathcal{L}$, and thus compatible with the monolingual sentences in L that we wish to code-mix. In Figure 2 we show an example of this transformation in the 'Training' box. Finally, we fine-tune m-BERT for a sequence-labeling task of predicting the language ID tags on $\mathbf{z}'$. Note, if we reapply the langID tool on $\mathbf{z}'$, the replaced tokens will not be predicted as English. In contrast, m-BERT can learn the code-mixing patterns so that it can predict which tokens in a monolingual sentence are most natural candidates for expressing in English.

3.2 Generate Switched Words

In the second phase, we use existing alignment libraries such as SimAlign (Jalili Sabet et al., 2020) to align source and target words between sentence pairs $(\mathbf{x}, \mathbf{y})$ in the parallel data (L, E). Let $p_1, \dots, p_n$ denote the predicted switches on an input non-code-mixed sentence $\mathbf{x} : x_1, \dots, x_n$ by the m-BERT model above. Then for each token $x_i \in \mathbf{x}$ that is predicted to switch to English i.e., $p_i \in \{\text{En, En-Trans}\}$ we replace the word with its aligned word(s) in $\mathbf{y}$ if they exist. Additionally, if p_i is En-Trans we transliterate the aligned English word to the native script. The resulting code-mixed sentence $\mathbf{x}'$ and $\mathbf{y}$ form a parallel pair for training the translation model. In Figure 2 we show an example in the 'Inference' box.

An advantage of this method of augmenting the training data is that the target sentence $\mathbf{y}$ is not synthetically generated, and thus helps to preserve the language model of the target sentences. We apply the above transformation on the given parallel corpus (L, E). Also, for the monolingual English sentence E_M we use a back-translation model to translate E_M to sentences L_M in the native language $\mathcal{L}$. This gives us pseudo parallel data (L_M, E_M). We transform this corpus also to code-mixed parallel data using the above process.



Na
En-Trans
Na
En
Na
Na
mBERT
यह प्रोग्राम का महत्वपूर्ण भाग है
translate
यह प्रोग्राम का important भाग है
Training
Inference
Perturbed Source Sent
इस प्रोग्रामिंग tutorial में हम सीखेंगे
copy target and transliterate
copy target
Na En-Trans En Na Na Na
mBERT
Alignments
इस - this
प्रोग्रामिंग - programming
पाठ - tutorial
में - In
हम - we
सीखेंगे - learn
Target Sent
In this programming tutorial we will learn
Source Sent
इस प्रोग्रामिंग पाठ में हम सीखेंगे

Figure 2: An illustration of our mBERT based perturbation method through an example. The top left box shows our method for training the mBERT model with limited in-domain code-mixed data. The rest of the image shows the inference procedure for creating a code-mixed to pure English sentence pair starting from a pure Native to pure English pair.

4 Experiments

Parallel Corpus For Hi→En experiments, we use the IIT Bombay English-Hindi Parallel Corpus (Kunchukuttan et al., 2018) as the base parallel training data for our models. The corpus contains parallel data from a number of diverse sources and domains. Test and dev splits are from the WMT 2014 English-Hindi shared task (Bojar et al., 2014). The training set has about 1.6M sentence pairs, the dev set has 500 sentences and the test set has 2507 sentences. We also move about 2,000 randomly selected sentences from the training set to the dev set. For Bn→En, we use 1M parallel sentences from Opus (Tiedemann, 2012) for training and 2000 randomly selected pairs each for validation and testing.

Code-Mixed Parallel Test Dataset While code-mixing is most common in social media and web search, it is difficult to get parallel data from these applications. One rare find was a video lectures website called the Spoken Tutorial Project [3]. The project comprises of transcripts of video lectures spanning technologies like operating systems, programming languages, and popular software in heavily code-mixed Hindi and Bangla (among other

[2]We use the IndicTrans (Bhat et al., 2015) library for transliterating target words.

[3]https://spoken-tutorial.org/

Indian languages), and also in English. After aligning the timestamps and some cleaning we collected 30.6K parallel sentences for code-mixed Hindi, and 28.6K sentences for code-mixed Bengali. [4] Code-mixing statistics on this dataset is shown in Table 1.

Non-Parallel Code-Mixed We also collect all source sentences that could not be aligned and are therefore not a part of the parallel test data. This dataset of 26.4K sentences for code-mixed Hindi and 17.4K sentences for code-mixed Bangla, serves as our limited code-switched corpus M during training.

Mononlingual English All English sentences from the Spoken Tutorial dataset for which there are no parallel code-mixed sentence in Hindi or Bangla comprise the monolingual English corpus. We found around 54K such sentences. This dataset is used to create back-translated data, and serves to domain adapt the translation models to the target distribution. For En$\rightarrow$Hi, use the Helsinki-NLP model [5] from Huggingface for back-translation. For En$\rightarrow$Bn back-translations, we train a model with the same parallel data from Opus that we use for forward models.

PHINC Dataset We also evaluate the efficacy of our data augmentation methods on the recently released PHINC dataset (Srivastava and Singh, 2020). The dataset contains roughly 13.5K translated sentence pairs from Twitter. The source texts are almost exclusively written in Latin and contain a mixture of Hindi and English words. Since no train-test splits are provided by the authors, we randomly split the dataset into 5000 test sentence pairs and use 500 sentence pairs for validation. We separate the remaining 8000 sentence pairs into a code-mixed corpus and a monolingual English corpus, to match the setup for our other experiments. All models that we train for this dataset involve a preliminary step of transliterating the Devanagari source (in IITB parallel data, and back-translations) to Latin.

Model and Experiment Setup All models are trained with the fairseq toolkit (Ott et al., 2019). For data preparation, we first run tokenization with IndicNLP (Kunchukuttan, 2020) for source sentence and Moses tokenizer [6] for target sentences. For models trained for PHINC data, Devanagari source is transliterated to Latin using IndicTrans (Bhat et al., 2015). Next, we apply BPE with code learnt on training set for source and target jointly, for 20,000 operations. We train with the transformer architecture with shared source and target embeddings. We use Adam optimizer with lr = 5e-4 and 4000 warmup steps, train upto 100 epochs and select the best checkpoint based on loss on the validation split. Results on Hi$\rightarrow$En Spoken Tutorial dataset are reported by training 3 models with different seeds and averaging BLEU scores from the best checkpoint for each model. For other datasets we only train a single model for each method.

Baselines To evaluate the importance of conditioning on the monolingual sentence, we design simpler variants that switch tokens based on content independent code-mixing statistics from the limited code-mixed data M. These two methods serve as competitive baselines for our model: **Unigram Random** that switches tokens to En or En-Trans based on their unigram statistics in M, and **Bigram Random** that switches based on bigram statistics in the LangId of adjacent tokens. We also compare against Samanta et al. (2019), by training their model on our limited code switched data, and then sampling switching patterns to perturb data similar to the Bigram Random method.

Finally, to tease apart the effect our perturbations from domain adaptation we also compare against the **As Is** baseline where we train models with parallel and back-translated in-domain monolingual English data.

Overall Results In Table 2 we present BLEU for our code-mixed Hindi translation model on four test sets: the code-mixed test set (ST-Test), the non-code-mixed test-set (NewsTest), and two adversarial subsets of ST-Test that we create as follows. The first, *ST-OOV*, comprises of sentence pairs where across source and target, at least two words were not found in the training data. This check is performed before sub-word tokenization. The second, *ST-Hard*, comprising of the 2,000 sentence pairs on which the sentence-level BLEU from the base model was the lowest. For code-mixed Bangla, we have equivalent test sets except the NewsTest. Table 3 presents our results for code-mixed Bangla. In Table 4, we present results on models trained for the PHINC dataset on the code-mixed test set only.

[4]Our aligned data is available at https://github.com/shruikan20/Spoken-Tutorial-Dataset

[5]https://huggingface.co/Helsinki-NLP/opus-mt-en-hi

[6]https://github.com/moses-smt/mosesdecoder

Method	ST-Test	ST-OOV	ST-Hard	NewsTest
As Is	43.93 (±0.44)	41.37 (±0.46)	18.63 (±0.61)	21.66 (±0.27)
Samanta19	45.42 (±0.37)	43.33 (±0.43)	21.97 (±0.78)	21.86 (±0.11)
Unigram	45.15 (±0.18)	43.08 (±0.31)	21.92 (±0.33)	21.59 (±0.28)
Bigram	45.63 (±0.1)	43.22 (±0.03)	22.27 (±0.2)	21.60 (±0.26)
mBERT	**46.40** (±0.38)	**44.55** (±0.25)	**23.41** (±0.25)	21.67 (±0.19)

Table 2: Average BLEU scores comparing models trained with different perturbation methods for code-mixed Hindi to English translation. Standard deviation is reported in brackets.

Method	ST-Test	ST-OOV	ST-Hard
As Is	36.4	36.02	16.36
Unigram	36.77	36.61	17.58
Bigram	37.45	37.34	18.39
mBERT	37.43	37.28	17.82

Table 3: BLEU scores on code-mixed Bangla to English Spoken Tutorial test set.

Method	PHINC Test
As Is	25.28
Unigram	29.33
Bigram	29.14
mBERT	29.3

Table 4: BLEU scores on the PHINC test set.

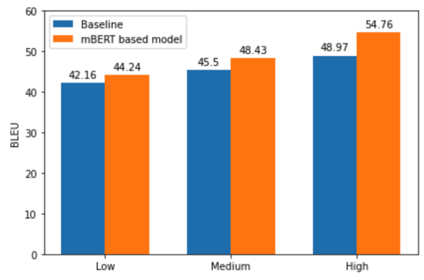


Figure 3: Improvements in BLEU with mBERT based model versus baseline across three splits of the test set.

We observe that our mBERT-based method substantially beats the AsIs method across all test sets for code-mixed Hindi and Bangla Spoken Tutorial data and PHINC data. The mBERT method also provides higher gains than the baselines on all three code-mixed test sets for Hi→En, while not reducing the accuracy on the original NewsTest. For Bn→En and PHINC, we observe that the Unigram and Bigram methods perform similar to the mBERT method showing that these are competitive methods in themselves. Overall, the effectiveness of perturbing parallel data is shown clearly in these experiments.

An interesting observation from Table 2 is that although our gain was about 2.5 BLEU points on ST-Test, on the adversarial sets we observed much higher gains — 3.2 for ST-OOV and 4.8 for ST-Hard. Our model also outperforms Samanta et al. (2019) on all code-mixed test sets while maintaining similar performance on NewsTest.

Sensitivity to amount of Code-mixing We investigate the gains in BLEU achieved by our method on sentences with varying levels of code mixing measured as the fraction of En and En-Trans words in source sentences in the code-mixed Hindi ST-Test set. We split the test set into three parts — Low (below 0.25), Medium (below 0.5), and High. Figure 3 shows the BLEU achieved by our method and the baseline. The biggest gains of about 5.8 BLEU can be seen in the test sentences with high levels of code-mixing. This shows that our data augmentation strategy does have the desired effect of better handling of heavily code-mixed inputs.

5 Conclusion

Machine translation of code-mixed inputs to English is an important task for which parallel training data is scarce. We presented a simple mBERT-based method of converting existing parallel data into code-mixed parallel data. Augmenting existing training data with this synthetic parallel data leads to substantial gains in BLEU on heavily code-mixed inputs without worsening accuracy on non-code-mixed inputs. However, gains are larger for some language pairs than others. Furthermore, code-mixed data from informal sources like Twitter presents additional challenges like noisy inputs stemming from non-canonical transliterations, informal language use, and misspellings. Our ongoing and future work includes evaluating the model on more languages and handling noisy inputs.

Acknowledgements The experiments reported in the paper were made possible by a Google Cloud Platform grant and a Tensor Flow Research Cloud free TPU access. We would like to thank Prof. Kannan M. Moudgalya from IIT Bombay, who leads the Spoken Tutorial initiative and provided us access to transcripts from the website enabling these experiments. Finally, we would also like to thank Aravindan Raghuveer from Google Research for discussions which helped shape the direction for this work.

References

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

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

Ching-Ting Chang, Shun-Po Chuang, and Hung-yi Lee. 2019. Code-switching sentence generation by generative adversarial networks and its application to data augmentation. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 554–558. ISCA.

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

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.

Hila Gonen and Yoav Goldberg. 2019. Language modeling for code-switching: Evaluation, integration of monolingual data, and discriminative training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

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

Xinhui Hu, Qi Zhang, Lei Yang, Binbin Gu, and Xinkang Xu. 2020. Data Augmentation for Code-Switch Language Modeling by Fusing Multiple Text Generation Methods. In *Proc. Interspeech 2020*, pages 1062–1066.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1627–1643, Online. Association for Computational Linguistics.

Anoop Kunchukuttan. 2020. The Indic-NLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The iit bombay english-hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Grandee Lee and Haizhou Li. 2020. Modeling code-switch languages using bilingual parallel corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5175–5181. AAAI Press.

Vivek Srivastava and Mayank Singh. 2020. PHINC: A parallel Hinglish social media code-mixed corpus for machine translation. In *Proceedings of the*

Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pages 41–49, Online. Association for Computational Linguistics.

Karan Taneja, Satarupa Guha, Preethi Jyothi, and Basil Abraham. 2019. Exploiting monolingual speech corpora for code-mixed speech recognition. In *INTERSPEECH*, pages 2150–2154.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Yuan Zhang, Jason Riesa, Dan Gillick, Anton Bakalov, Jason Baldridge, and David Weiss. 2018. A fast, compact, accurate model for language identification of codemixed text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 328–337.