A Semi-supervised Approach to Generate the Code-Mixed Text using Pre-trained Encoder and Transfer Learning

Deepak Gupta, Asif Ekbal, Pushpak Bhattacharyya
Indian Institute of Technology Patna, India
{deepak.pcs16, asif, pb}@iitp.ac.in

Abstract

Code-mixing, the interleaving of two or more languages within a sentence or discourse is ubiquitous in multilingual societies. The lack of code-mixed training data is one of the major concerns for the development of end-to-end neural network-based models to be deployed for a variety of natural language processing (NLP) applications. A potential solution is to either manually create or crowd-source the code-mixed labelled data for the task at hand, but that requires much human efforts and often not feasible because of the language specific diversity in the code-mixed text. To circumvent the data scarcity issue, we propose an effective deep learning approach for automatically generating the code-mixed text from English to multiple languages without any parallel data. In order to train the neural network, we create synthetic code-mixed texts from the available parallel corpus by modelling various linguistic properties of code-mixing. Our code-mixed text generator is built upon the encoder-decoder framework, where the encoder is augmented with the linguistic and task-agnostic features obtained from the transformer based language model. We also transfer the knowledge from a neural machine translation (NMT) to warm-start the training of code-mixed generator. Experimental results and in-depth analysis show the effectiveness of our proposed code-mixed text generation on eight diverse language pairs.

1 Introduction

Multilingual content is very prominent on social media handles, especially in the multilingual communities like the Indian ones. Code-mixing is a common expression of multilingualism in informal text and speech, where there is a switch between the two languages, frequently with one in the character set of the other language. This has been a mean of communication in a multi-cultural and multi-lingual society, and varies according to the culture, beliefs, and moral values of the respective communities.

Linguists have studied the phenomenon of code-mixing, put forward many linguistic hypotheses (Belazi et al., 1994; Pfaff, 1979; Poplack, 1978), and formulated various constraints (Sankoff and Poplack, 1981; Di Sciullo et al., 1986; Joshi, 1982) to define a general rule for code-mixing. However, for all the scenarios of code-mixing, particularly for the syntactically divergent languages (Berk-Seligson, 1986), these limitations cannot be postulated as a universal rule.

In recent times, the pre-trained language model based architectures (Devlin et al., 2019; Radford et al., 2019) have become the state-of the-art models for language understanding and generation. The underlying data to train such models comes from the huge amount of corpus, available in the form of Wikipedia, book corpus etc. Although, these are readily available in various languages, there is a scarcity of such amount of data in code-mixed form which could be used to train the state-of-the-art transformer (Vaswani et al., 2017) based language model, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLM (Lample and Conneau, 2019) etc. The existing benchmark datasets on various NLP tasks can also be transformed to the code-mixed environmental setup, and subsequently be leveraged to assess the model's flexibility under the multilingual framework. Creating large-scale code-mixed datasets for such tasks is expensive and time-consuming as it requires considerable human efforts and language expertise to generate these manually. Therefore, it is necessary to build an automated code-mixed generation system capable of modeling intra-sentential language phenomenon.

In this paper, we formulate the code-mixed phenomenon using the feature-rich and pre-trained language model assisted encoder-decoder paradigm. The feature-rich encoder assists the model to capture the linguistic phenomenon of code-mixing, especially to decide when to switch between the two languages. Similarly, the pre-trained language model provides the task-agnostic feature which helps to encode the generic features. We adopt the gating mechanism to fuse the features of the pre-trained language model and the encoder. Additionally we also perform transfer learning to learn the prior distribution from the pre-trained NMT. The pre-trained NMT weights are used to initialize the code-mixed generation network. Transfer learning guides the code-mixed generator to generate syntactically correct and fluent sentences.

We summarize the contributions of our work below:
(i). We propose a robust and generic method for code-mixed text generation. Our method exploits the capabilities of linguistic feature-rich encoding and pre-trained language model assisted encoder to capture the code-mixed formation across the languages. Our model is further tailored to generate the syntactically correct, adequate and fluent code-mixed sentences using the prior knowledge acquired by the transfer learning approach.
(ii). To warm start the training, we devise a robust and generic technique to automatically create the synthetic code-mixed sentences by modeling the linguistic properties using the parallel corpus. To the best of our knowledge, this is the very first step where we attempt to propose a generic method that produces the correct and fluent code-mixed sentences on multiple language pairs. The generated synthetic dataset will be a useful resource for machine translation and multilingual applications.
(iii). We demonstrate with detailed empirical evaluations the effectiveness of our proposed approach on eight different language pairs, *viz.* English-Hindi (en-hi), English-Bengali (en-bn), English-Malayalam (en-ml), English-Tamil (en-ta), English-Telugu (en-te), English-French (en-fr), English-German (en-de) and English-Spanish (en-es).

2 Related Work

In the literature, there have been efforts for creating code-mixed texts by leveraging the linguistic properties. Pratapa et al. (2018) explored the equivalence constraint theory to construct artificial code-mixed data to reduce the perplexity of the RNN-based language model.

Winata et al. (2018) proposed a multitask learning framework to address the issue of data scarcity in code-mixed setting. Particularly, they leveraged the linguistic information using a shared syntax representation, jointly learned over Part-of-Speech (PoS) and language modeling on code-switched utterances. Garg et al. (2018) exploited SeqGAN in the generation of the synthetic code-mixed language sequences. Most recently, Winata et al. (2019a) utilized the language-agnostic meta-representation method to represent the code-mixed sentences. There are also other studies (Adel et al., 2013a,b, 2015; Choudhury et al., 2017; Winata et al., 2018; Gonen and Goldberg, 2018; Samanta et al., 2019) for code-mixed language modelling.

There are some other NLP areas like parts-of-speech (Solorio and Liu, 2008b; Gupta et al., 2017; Patel et al., 2016), sentiment analysis (Rudra et al., 2016; Gupta et al., 2016a), question answering (Gupta et al., 2018b; Chandu et al., 2017), language identification (Solorio et al., 2014; Gupta et al., 2014; Hidayat, 2012; Solorio and Liu, 2008a), entity extraction (Gupta et al., 2018a; Bhat et al., 2016; Gupta et al., 2016b), etc, where code-mixing phenomena are explored and analyzed.

In contrast to sthese existing works, firstly, we provide a linguistically motivated technique to create the code-mixed datasets from multiple languages with the help of parallel corpus (English to respective language). Thereafter, we utilize this data to develop a neural based model to generate the code-mixed sentences from the English sentence. Our current work has a wider scope as the underlying architecture can be used to harvest the code-mixed data for the various NLP tasks not only limited to the language modelling and speech recognition as it is generally been focused in the literature. In contrast to the previous studies, where only a few of the language pairs were considered for code-mixing, we propose an effective approach which shows its effectiveness in generating code-mixed sentences for eight different language pairs of diverse origins and linguistic properties.

3 Synthetic Code-Mixed Generation

We follow the matrix language frame (MLF) (Myers-Scotton, 1997; Joshi, 1982) theory to generate the code-mixed text. It is less restrictive and can easily be applied on many language pairs. According to MLF, a code-mixed text will have a

Language (L1)		Language (L2)		Code-Mixed (L1-L2)
en	India’s agriculture is their main strength.	hi	भारत की कृषि उनकी मुख्य ताकत है।	India’s कृषि इसकी main strength है।
en	Especially valuable people like Connor Rooney.	bn	বিশেষত কনর রুনির মতো মূল্যবান ব্যক্তি।	বিশেষ Connor Rooney মতো valuable ব্যক্তি।
en	Glasses and cups, whatever they are, can be turned upside down.	ta	கண்ணாடிகள் மற்றும் கோப்பைகள், அவை எதுவாக இருந்தாலும், தலைகீழாக மாற்றலாம் .	Glasses மற்றும் cups அவை எதுவாக இருந்தாலும், தலைகீழாக மாற்றலாம்
en	Democracy and development go hand in hand.	de	Demokratie und Entwicklung gehen Hand in Hand.	Democracy und Development gehen Hand in Hand.
en	We abolish national embassies.	fr	Nous abolissons les ambassades nationales.	Nous abolissons les embassies national.

Table 1: Samples of code-mixed **(L1-L2)** generated sentences from the parallel sentence of the language **L1** and **L2**.

dominant language (matrix language) and inserted language (embedded language). The insertions could be words or larger constituents and they will comply with the grammatical frame of the matrix language. However, random word insertions could lead to the formation of unnatural code-mixed sentences, which are very rare in practice.

Linguistically informed strategy to insert the words or constituents can improve the quality of code-mixed text. It is also shown in the literature (Gupta et al., 2018b) that such strategy benefits the quality of generated code-mixed text. In our work, we utilize the parallel corpora to learn the alignments between English and other languages. Given a pair of parallel sentences, we identify the words from English and substitute their aligned counterparts with the identified English words to synthesize the English embedded code-mixed sentences. The input to our synthetic code-mixed generation algorithm (details are in **Appendix**) is a parallel sentence pair. We use the Indic-nlp-library[1] to tokenize the sentences of the Indic languages. Moses based tokenizer[2] is used to translate the European and English language texts. Thereafter, we learn the alignment matrix, which guides to select the words or phrases to be mixed in the language.

We use the official implementation[3] of the fast-align algorithm (Dyer et al., 2013) to obtain the alignment matrix. The alignment matrix is used to construct the aligned phrases between the parallel sentences. We extract the PoS (mainly adjective), named entity (NE) and noun phrase (NP) from the English sentences, and insert them into the appropriate places of the sentences in the other language (i.e. the target language) counterparts. We use the Stanford library[4] Stanza (Qi et al., 2020) to extract these linguistic features. We can extract multiple aligned phrases from the alignment matrix. However, in our proposed algorithm, we are interested in aligned words/phrases which are the NEs of types ‘Person’, ‘Location’ and ‘Organization’, noun phrases and adjective words. Let us see an example of En-Hi parallel sentence:

- **En:** When was Mahatma Gandhi born?
- **Hi:** महात्मा गांधी का जन्म कब हुआ था?
- **Code-Mixed (En-Hi):** *Mahatma Gandhi* का जन्म कब हुआ था?

The NE *Mahatma Gandhi* of type ‘Person’ is mixed in En-Hi[5] code-mixed sentence.

The need of replacing the aligned noun phrases can be understood with the examples of parallel sentences shown in Fig 1. In the given example, ‘*girl*’ and ‘*red umbrella*’ are the noun phrases[6] in the English sentence. To obtain the corresponding code-mixed sentence, their aligned phrases `लड़की' and `लाल छाता' need to be replaced with English counterparts ‘*girl*’ and ‘*red umbrella*’, respectively. Similarly, we can visualize the requirement of choosing the adjective words to be mixed in the code-mixed sentence by the following example:

- **En:** The situation in Mumbai has not yet

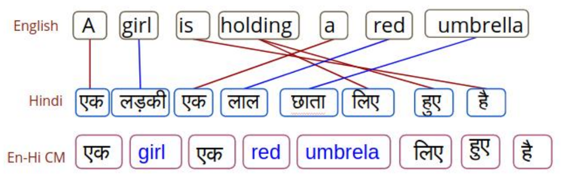


Figure 1: An example of the alignment between a pair of parallel sentences. The aligned words which are mixed in En-Hi code-mixed (CM), are shown in blue.

[1] https://github.com/anoopkunchukuttan/indic_nlp_library

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

[3] https://github.com/clab/fast_align

[4] https://github.com/stanfordnlp/stanza

[5] Please use the following link to transliterate the Indic scripts: http://www.learnsanskrit.org/tools/sanscript

[6] We remove the determiner from the noun phrases as the insertion of determiner in the code-mixed make the sentence unnatural and incorrect.

come to normal.

- **Hi:** मुंबई में स्थिति अभी तक सामान्य नहीं हुई है।
- **Code-Mixed (En-Hi):** Mumbai में situation अभी normal नहीं हुई है ।

In the given example the adjective '*normal*' is present in the English sentence. To make the corresponding code-mixed sentence the adjective word has to be inserted in the code-mixed sentence. In this case corresponding target (i.e. Hindi here) word `सामान्य' need to be replaced with the word '*normal*' in the En-Hi code-mixed sentence. We show some samples in Table 1, and more details in the **Appendix**.

4 Methodology

We depict the architecture of our proposed model in Figure 2.

Problem Statement: Given an English sentence E having m words $e_1, e_2, \dots, e_m$, the task is to generate the code-mixed sentence $\hat{C}$ having a sequence of n words $\hat{C} = \{y_1, y_2, \dots, y_n\}$.

4.1 Sub-word Vocabulary

The task of generation using neural networks requires a fixed-sized vocabulary. To deal with the problem of Out-of-Vocabulary (OOV) words, we use the Byte-pair encoding (BPE) (Sennrich et al., 2016), and segment the words into sub-words. The sub-word based tokenization schemes inspired by BPE have become the norm in most of the advanced models including the very popular family of contextual language models like XLM (Lample and Conneau, 2019), GPT-2 (Radford et al., 2019), etc. In this work, we process the language pairs with the vocabulary created using the BPE.

4.2 Feature-rich and Pre-trained Language Model Assisted Encoder

We introduce a specific encoder which is equipped with linguistic features and pre-trained language model features. Firstly, we discuss the linguistic feature encoding to the standard long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder. Later, we describe the pre-trained language model feature assisted encoder.

In order to encode the input English sentence, we use the two-layered LSTM networks. Firstly, we tokenize the English sentence to the sub-word tokens using BPE. Each sub-word is mapped to a real-valued vector through an embedding layer. In addition, we also incorporate the linguistic features in the form of NE and PoS. The motivation to use these linguistic features comes from the synthetic code-mixed text generation (c.f. section 3) itself, where these features guide the generation process by selecting the words to either replace with their aligned English words or to keep the same word in the code-mixed sentence. In neural based generation, explicit linguistic features help the decoder to decide whether to copy from the English (source) or generate from the vocabulary.

The network takes the concatenation of word embedding u_t, NE encoding n_t and p_t (will be discussed shortly) at each time step t and generate the hidden state as follows:

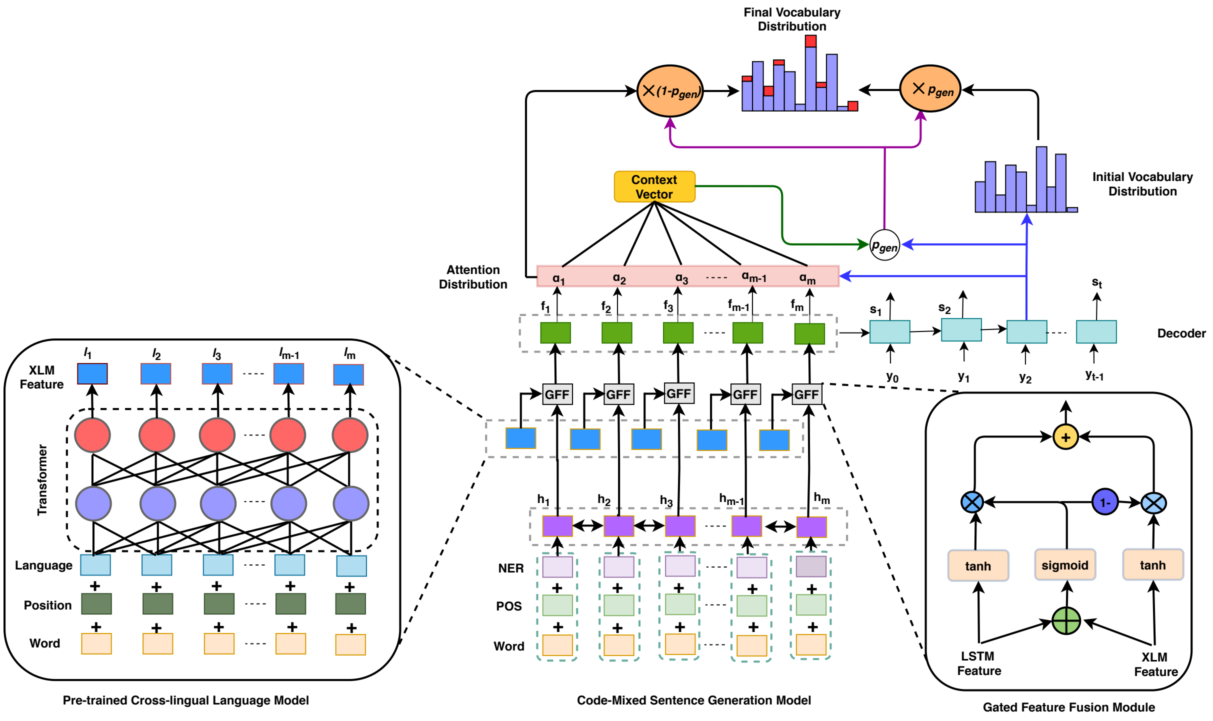
$$h_t = LSTM(h_{t-1}, [u_t, n_t, p_t]) \quad (1)$$

We compute the forward and backward hidden states $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$, and compute the document encoder as the concatenation of the two hidden states, $h_i = [\overrightarrow{h}_i \oplus \overleftarrow{h}_i]$.

Feature Encoding: The NE and PoS features are encoded to the real valued vectors. We initialize the NE and PoS feature representations n_t and p_t at time t using the random vectors of size 20. The NE and PoS features are represented by the $\{n_1, n_2, \dots, n_m\}$ and $\{p_1, p_2, \dots, p_m\}$, respectively.

Pre-trained Language Model Feature: Recent studies have shown the effectiveness of language model pre-training for text generation (Radford et al., 2019; Dong et al., 2019; Song et al., 2019). We utilize the pre-trained feature from the cross-lingual language model (XLM) (Lample and Conneau, 2019). The XLM model is trained with three objective functions: Masked Language Modeling (MLM), Causal Language Modeling (CLM), and Translation Language Modeling (TLM). In the CLM objective the task is to model the probability of a word given the previous words. The MLM objective was introduced in Devlin et al. (2019), where the task is to predict the masked words from the sentence given the remaining words. The TLM objective is an extension of MLM for the parallel sentences. For the TLM objective function the input sentence is the concatenation of the source and target sentence and a random word is masked from the concatenated sentence and rest of the words is used to predict the masked word.

The XLM model trained with multiple objective functions on different languages together has



Final Vocabulary Distribution
Initial Vocabulary Distribution
Context Vector
Attention Distribution
Decoder
XLM Feature
Transformer
Language
Position
Word
NER
POS
GFF
LSTM Feature
tanh
sigmoid
Pre-trained Cross-lingual Language Model
Code-Mixed Sentence Generation Model
Gated Feature Fusion Module

Figure 2: The architecture of the proposed code-mixed sentence generation model. The **left** part of the image shows the architecture of cross-lingual language model (XLM). The XLM feature along with the linguistic feature obtained from the Bi-LSTM encoder is passed to the Gated Feature Fusion (GFF) module. The **right** part of the image demonstrates the working of GFF module. It is to be noted that the transfer learning is enabled by initializing the parameters of the proposed model from the pre-trained neural machine translation model.

shown the effectiveness on cross-lingual classification and machine translation. By virtue of dealing with multiple languages and setting the state-of-the-arts in language generation task, the pre-trained XLM model is adopted to extract the language model features for code-mixed generation as it is reminiscence of the cross-lingual and generation paradigms. For the given input sentence $E : \{e_1, e_2, \ldots, e_m\}$, we extract the language model feature $L : \{l_1, l_2, \ldots, l_m\}$.

The extracted language model features are fused to the linguistic features as follows:

$$\begin{aligned} h_t^* &= tanh(W_h h_t + b_h) \\ l_t^* &= tanh(W_l l_t + b_l) \\ g &= \sigma(W_g.[h_t \oplus l_t]) \\ f_t &= g \odot h_t^* + (1-g) \odot l_t^* \end{aligned} \quad (2)$$

where, $\oplus$ and $\odot$ are the concatenation and element-wise multiplication operator. First, we project both the features h_t and l_t into the same vector space h_t^* and l_t^* via feed-forward network. Thereafter, we learn the gated value g which controls the flow of each feature. The gated value g controls how much of each feature should be the part of the final encoder representation f_t.

4.3 Decoding with Pointer Generator

We use the one-layer LSTM network with the attention mechanism (Bahdanau et al., 2015) to generate the code-mixed sentence $y_1, y_2, \ldots, y_n$ one word at a time. In order to deal with the rare or unknown words, the decoder has the flexibility to copy the words from documents via the pointing mechanism (See et al., 2017; Gulcehre et al., 2016). The LSTM decoder reads the word embedding u_{t-1} and the hidden state s_{t-1} to generate the hidden state s_t at time step t. Concretely,

$$s_t = LSTM(s_{t-1}, u_{t-1}) \quad (3)$$

Similar to (See et al., 2017), we compute the attention distribution α_t and context vector c_t. The generation probability is computed as follows:

$$p_{gen} = \sigma(\mathbf{W_a} c_t + \mathbf{W_b} s_t + \mathbf{W_u} u_t) \quad (4)$$

where $\mathbf{W_a}$, $\mathbf{W_b}$ and $\mathbf{W_s}$ are the weight matrices and σ is the Sigmoid function. We also consider the copying of the word from the English sentence. The probability to copy a word from English sentence at given time t is computed by the following equation:

$$P_{copy}(w) = \sum_{i=1}^{m} \alpha_{t,i} * \mathbf{1}\{w == w_i\} \qquad (5)$$

where $\mathbf{1}\{w == w_i\}$ denotes the vector of length m having the value 1 where $w == w_i$, otherwise 0. The final probability distribution over the dynamic vocabulary (English and code-mixed sentence vocabulary) is calculated by the following:

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen})P_{copy}(w) \qquad (6)$$

4.4 Transfer Learning for Code-mixing

Transfer learning deals with the performance improvement of a task by using the learned knowledge from a near similar task. It has shown promise in solving various problems (Torrey and Shavlik, 2010; Pan and Yang, 2009) by significantly reducing the amount of training instances. In our case, we formulate the problem of code-mixed text generation with respect to the NMT framework. A closer to the code-mixed sentence reveals that the translated target text (XX[7]) and code-mixed (En-XX) shares many words. For example:

- **Source (En):** The situation in Mumbai has not yet come to normal.
- **Target (Hi):** मुंबई में स्थिति अभी तक सामान्य नहीं हुई है ।
- **Code-Mixed (En-Hi):** Mumbai में situation अभी normal नहीं हुई है ।

In the above sentences, Target (Hi) and Code-mixed (En-Hi) share many words (underlined words). Because of this underlying similarity between the machine translation and code-mixed sentence generation, we adapted the transfer learning approach used in machine translation (Zoph et al., 2016; Kocmi and Bojar, 2017) for code-mixed text generation.

We first train an NMT model on a large corpus of parallel sentences as discussed in Section 3. Next, we initialize the code-mixed text generation model with the already-trained NMT model. This is then trained on the synthetic code-mixed dataset. Rather than initializing the code-mixed model from the random parameters, we initialize it with the weights from the NMT model. By doing this, we achieve strong prior distribution from the NMT model to code-mixed text generation. When we train the code-mixed generation model initialized with the weights of the NMT model, it acquires the prior knowledge of translating the English sentences into the target language XX, and then is fine-tuned to adopt to the code-mixed phenomenon.

5 Results and Analysis

We evaluate the performance of our proposed approach on the synthetic code-mixed text from eight different language pairs. The datasets can be found here[8]. We compare the performance of our proposed code-mixed generation model with the **(i) Seq2Seq** (Sutskever et al., 2014), **(ii) Attentive-Seq2Seq** (Bahdanau et al., 2015) and **(iii) Pointer Generator** (See et al., 2017) baselines.

5.1 Experimental Setup

In our experiments, we use the same vocabulary for both the encoder and decoder. For the language pairs: en-hi, en-es, en-de, en-fr, we use the learned BPE codes[9] on 15 languages to segment the sentences into sub-words and use this vocabulary[10] to index the sub-words. For the language pairs: en-bn, en-ml, en-ta, en-te, we use the learned BPE codes[11] on 100 languages from the XLM model to segment the sentences into sub-words and use the correspondent vocabulary to index the sub-words. The same set of vocabulary is used to extract the pre-trained language model feature and the corresponding NMT model for the transfer learning. We use the aligned multilingual word embedding[12] of dimension 300 for the language pairs: en-es, en-de, en-fr, en-hi and en-bn from Bojanowski et al. (2017); Joulin et al. (2018). For the rest of the language pairs, we obtain the monolingual embedding[13] from Bojanowski et al. (2017) and use the MUSE library released by Lample et al. (2018) to align the vector in the same vector space. The embeddings of NE and PoS information are randomly initialized with the dimension of 20.

[7]XX may belong to *'es', 'de', 'fr', 'hi', 'bn', 'ml', 'ta', 'te'*

[8]http://www.iitp.ac.in/~ai-nlp-ml/resources.html

[9]https://dl.fbaipublicfiles.com/XLM/codes_xnli_15

[10]https://dl.fbaipublicfiles.com/XLM/vocab_xnli_15

[11]https://dl.fbaipublicfiles.com/XLM/codes_xnli_100

[12]https://fasttext.cc/docs/en/aligned-vectors.html

[13]https://fasttext.cc/docs/en/pretrained-vectors.html

Model	en-es			en-de			en-fr			en-hi		
	B	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**
Seq2Seq	16.42	36.03	24.23	19.19	36.19	24.87	19.28	38.54	26.41	15.49	35.29	23.72
Attentive-Seq2Seq	17.21	36.83	25.41	20.12	37.14	25.64	20.12	39.30	27.54	16.55	36.25	24.97
Pointer Generator	18.98	37.81	26.13	21.45	38.22	26.14	21.41	40.42	28.76	17.62	37.32	25.61
Proposed Model	**22.47**	**41.24**	**29.45**	**24.15**	**42.76**	**30.47**	**24.89**	**43.54**	**31.26**	**21.55**	**40.21**	**28.37**
(-) BPE	21.72	40.67	28.65	23.31	41.89	29.76	24.27	43.02	30.84	20.89	39.54	27.43
(-) PoS Feature	22.21	40.92	29.12	23.76	42.12	29.88	24.21	42.95	30.86	21.02	39.84	27.91
(-) NE Feature	21.52	40.32	28.41	22.19	41.64	29.39	23.92	42.52	30.37	20.42	39.20	27.46
(-) LM Feature	21.56	40.36	28.42	23.21	41.85	29.56	23.82	42.48	30.29	20.47	39.17	27.24
(-) GFF	21.59	40.28	28.59	23.24	41.75	29.50	23.87	42.58	30.46	20.31	39.24	27.51
(-) Transfer Learning	20.69	39.39	27.53	22.39	40.98	28.87	22.64	41.57	29.34	19.48	38.34	26.41

Table 2: Performance comparison of the proposed model for code-mixed generation with the baseline models. The impact of each component (by removing one at a time) on the performance of the model. Here, **B**: BLEU, **R**: Rouge-L and **M**: METEOR

Model	en-bn			en-ml			en-ta			en-te		
	B	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**	**B**	**R**	**M**
Seq2Seq	16.32	33.02	21.82	15.92	34.97	23.12	11.82	25.14	20.21	10.87	24.92	19.05
Attentive-Seq2Seq	17.29	34.12	23.08	17.21	35.91	23.94	13.09	26.57	21.41	12.14	26.17	20.11
Pointer Generator	18.24	35.86	24.36	18.49	37.16	25.12	14.03	27.84	22.53	13.21	27.37	21.17
Proposed Model	**21.49**	**39.11**	**27.32**	**21.61**	**40.23**	**28.01**	**15.69**	**29.56**	**23.88**	**14.81**	**29.23**	**22.56**
(-) BPE	20.81	38.64	26.65	20.89	39.73	27.49	15.12	28.92	23.19	14.15	28.75	21.82
(-) POS Feature	21.04	38.77	26.94	21.11	39.91	27.55	15.23	28.11	22.34	14.23	28.67	21.86
(-) NER Feature	20.49	38.14	26.33	20.63	39.29	27.11	15.19	29.06	23.48	14.51	28.63	22.26
(-) LM Feature	20.13	37.73	25.95	20.54	38.69	26.44	14.73	28.64	22.89	13.97	28.07	21.79
(-) GFF	20.57	38.11	26.36	20.69	39.18	27.07	15.24	28.84	23.19	14.29	28.67	21.88
(-) Transfer Learning	19.67	37.49	25.87	20.12	38.74	26.54	14.48	28.34	22.72	13.79	28.12	21.53

Table 3: Performance comparison of the proposed model for code-mixed generation with the baseline models

The hidden dimension of all the LSTM cells is set to 512. We use the pre-trained XLM model[14] to extract the language model feature of dimension 1024 for en-hi, en-es, en-de, en-fr language pairs. For the rest of the language pairs, the pre-trained model[15] trained on MLM objective function is used to extract the language model feature. We use beam search of beam size 4 to generate the code-mixed sentence. Adam (Kingma and Ba, 2015) optimizer is used to train the model with (i) $\beta_1 = 0.9$, (ii) $\beta_2 = 0.999$, and (iii) $\epsilon = 10^{-8}$ and initial learning rate of 0.0001. The maximum length of English and code-mixed tokens are set to 60 and 30, respectively. We set 5 as minimum decoding steps in each code-mixed language pair. We use the *en-hi* development dataset to tune the network hyper-parameters. All the model updates use a batch size of 16.

We evaluate the generated text using the metrics, BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005).

[14]`https://dl.fbaipublicfiles.com/XLM/mlm_tlm_xnli15_1024.pth`

[15]`https://dl.fbaipublicfiles.com/XLM/mlm_100_1280.pth`

5.2 Quantitative Analysis

We report the results of our proposed model in Table 2 and Table 3. Performance comparisons to the three baselines are reported in Table 2 and Table 3. The Pointer Generator based baseline is the superior amongst all the baselines and achieve the maximum Bleu score of 21.45 for the *en-de* code-mixed language pair. Our proposed model achieves the maximum Bleu score of 24.89 for the *en-fr* code-mixed language pair. The minimum Bleu score that we achieve is 14.81 for the en-te language pair. We achieve lower Bleu scores for the language pairs, en-ta and en-te compared to the other language pairs. It is because the number of training samples for *en-ta* and *en-te* are very low ($11,380$ and $9,105$) as compared to the other language pairs. Among the European languages, for *en-fr* pair, our model attains the highest performance; while for the Indian languages, our proposed model reports the comparable performance for both *en-hi* and *en-bn* language pairs.

We also perform the ablation study to asses the efficacy of the model's components. We remove each component at a time from the proposed model

en-de	**Input**	The real problem is statesponsored lawlessness.
	Reference	Das real problem ist die vom statesponsored lawlessness.
	PG	Das echtes problem ist die vom statesponsored Gesetz.
	Proposed	Das real problem ist vom statesponsored lawlessness.
	(-) TL	Das problem ist die statesponsored Gesetzlosigkeit.
en-es	**Input**	However we have proposed some minor changes.
	Reference	Con todo hemos propuestos algunas minor changes.
	PG	Sin embargo, hemo propuestos minero changes.
	Proposed	Sin embargo hemos propuestos algunas minor changes.
	(-) TL	Con todo hemos propuestos algunas minor cambios.
en-hi	**Input**	India's agriculture is their main strength.
	Reference	India का agriculture इसकी main strength है।
	PG	India's agriculture इसकी शक्ति है।
	Proposed	India का agriculture इसकी main strength है।
	(-) TL	India कृषि इसका main strength.
en-fr	**Input**	Read the statements by Giscard dEstaing.
	Reference	Lisez les statements de Giscard dEstaing.
	PG	Lisez déclarations de Giscard dEstaing.
	Proposed	Lisez les statement de Giscard dEstaing.
	(-) TL	Lisez de déclarations Giscard dEstaing.

Table 4: Sample code-mixed sentences generated using the pointer generator (**PG**), proposed model, and the variant of the proposed model without transfer learning (**-TL**).

Approach	**Human**	**B**	**R**	**M**
Synthetic	4.19	67.51	73.56	71.21
Pointer Generator	2.34	19.47	39.48	27.39
Proposed Model	3.26	24.65	43.55	29.11

Table 5: Comparison of different code-mixed text generation approaches on human and automatic evaluation metrics.

and report the results for each language pair in Table 2 and Table 3. The removal of BPE brings down the Bleu score from 0.57 (*en-ta*) to 0.84 (*en-de*). The BPE encoding helps the model to mitigate the OOV word issue by providing the sub-word level information. Similarly, the removal of PoS feature reduces the Bleu score by 0.26 (*en-es*) to 0.58 (*en-te*). The NE feature helps most to the *en-bn* code-mixed language pair as we observe the decrease of 1.0 Bleu points while the NE feature is removed. The LM feature is obtained from the pre-trained language model, and it helps the model to obtain the better encoded representation. The ablation study reveals that removal of LM feature decreases the Bleu score by 1.36 points. We observe the near similar impact of LM feature on each language pair. Finally, the transfer learning is also proven to be an integral component of the proposed model as it contributes to the maximum of 2.25 Bleu score for *en-fr* and minimum of 1.02 Bleu score of *en-te* code-mixed language pair. The difference between the maximum and minimum contribution may be attributed to the fact that, we have sufficient parallel corpus (197, 922) to train the *en-fr* NMT model as compared to the *en-te* parallel corpus (10, 105). We follow the bootstrap test (Dror et al., 2018) which confirms that the performance improvement over the baselines are statistically significant as $(p < 0.005)$.

5.3 Qualitative Analysis

We assess the quality of the generated code-mixed text, and show these samples in Table 4. We observe that the code-mixed sentences generated using the PG model are able to copy the entities from the given English sentence, but the generated code-mixed sentences are incomplete and not fluent compared to the reference sentences. For example, in *en-hi* pair the PG based code-mixed sentence missed the '*main*' word and it copies '*India's*' rather than generating '*India का*' which seems more natural and human-like code-mixed sentence.

Our analysis also reveals that quality of the generated code-mixed sentence without transfer learning lacks in fluency. The examples can be seen in the (-) TL generated code-mixed sentence (in Table 4) in the *en-hi* and *en-fr*. In contrast, the generated output using the proposed model takes the benefits of both the pointer generator and transfer learning to generate adequate, fluent and complete human-like code-mixed sentences. We observe that the proposed model learns when to switch between the languages, and when to either copy the entity/phrase from the English sentence or to generate from the vocabulary. The examples can be seen in *en-hi* language pair, where the model copies the word '*main strength*' from the English sentence, and it also switches between the languages at the appropriate time step by generating the correct word from the vocabulary.

We perform human evaluation to judge the quality of the generated code-mixed text. For human evaluation, we randomly sample 100 English sentences from the *en-hi* code-mixed dataset, and ask three English and Hindi speakers to manually formulate the code-mixed sentences. These were then used to evaluate the quality of the generated code-mixed sentences. We ask the speakers to score (from 1 to 5) the machine generated code-mixed sentence with respect to the human generated sentences. The rate will define how natural and human-like the code-mixed sentence sounds as compared to the human one. The scores are associated with the quality of the generated code-mixed sentence, where 1 shows that there is a *strong dis-*

agreement between the machine generated and human formulated code-mixed sentence. Similarly, 2, 3, 4 and 5 are the categorical scores for *Disagreement*, *Not Sure*, *Agreement* and *Strongly Agreement*, respectively.

We also compute the automatic evaluation metrics, BLEU, Rouge-L and Meteor. The comparison between the different approaches on human and automatic evaluation metrics are reported in Table 5. The reported human evaluation score corresponds to the average of all the three human experts. The proposed model achieves the human evaluation (naturalness) score of 3.26 compared to the synthetic generation of 4.19. It is to be noted that the algorithm of synthetic text generation needs parallel corpus. However, our neural generation model does not require any parallel data *except* at the time of warm-start with the synthetic data. The human evaluation achieves better score (3.26) compared to the strongest (2.34) pointer generator based baseline model.

Error Analysis: We closely analyze the outputs of our proposed model to realize the challenges faced. We take up the language pair (*en-hi*), study the errors encountered by the proposed approach. and We categorize them into the following types:

(1). Reference Inaccuracy: The errors encountered during the word alignment phase propagate, and lead to the inaccurate reference code-mixed sentences. Since, we use these sentences to train the generator model, it introduces errors in the generated code-mixed sentences too. This issue could possibly be reduced with an improved alignment algorithm.

(2). Missing/Incorrect Words: This is one of the very common error types, where the model generates the incorrect words/phrases. The missing or incorrect words cause fluency problem in the generated code-mixed sentence. We also observed that the majority of the missing words are *function words*, while incorrectly generated words belong to the *content words* category.

(3). Factual Inaccuracy: Our proposed model sometimes generates the factually incorrect NEs. These types of errors were mainly seen in the longer sentences, where the model was found to be confused to copy/generate the relevant entity in the given context.

(4). Code-Mixed Inaccuracy: We observe the inaccuracy in the generated sentence, where the model sometimes produces the sentence which either violates the code-mixed theory or is unnatural (not human-like).

(5) Rare Language Pairs: We notice that, the system makes the more errors on the *en-ta* and *en-te* language pairs. It can be understand by the fact that, we had comparatively lesser number of samples of these language pairs to train the system. This error can be reduced by training the system with sufficient number of samples.

(6) Others: We categorize the remaining errors in others category. The other type of errors include repeated word, inadequate sentence generation, extra word generation etc. We also observe that majority of the error occurred when the input sentence were relatively longer than 12 words.

We randomly take a sample of size 100 from the generated En-Hi code-mixed text and categorize them using the six different aforementioned error types. We found that top-3 frequent errors (Missing/Incorrect Words, Reference Inaccuracy, Code-Mixed Inaccuracy) come under 27.21%, 23.37%, and 17.44% respectively.

6 Conclusion

In this paper, we have proposed a neural network based effective method coupled with the linguistic and pre-trained feature representation along with the transfer learning to generate the code-mixed sentences. To train and evaluate the proposed approach, we have introduced a linguistically motivated approach for code-mixed sentence generation using the parallel sentences of any particular language pair. Our experimental results and in-depth analysis show that the feature representation and transfer learning together effectively improve the model performance and the quality of the generated code-mixed sentence. We have shown the effectiveness of the proposed approach on eight different language pairs. In future work, we plan to explore the unsupervised neural approach for code-mixed text generation.

Acknowledgment

Asif Ekbal gratefully acknowledges the Young Faculty Research Fellowship (YFRF) Award supported by the Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, and implemented by Digital India Corporation (formerly Media Lab Asia).

References

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE/ACM transactions on audio, speech, and language Processing*, 23(3):431–440.

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013a. Recurrent neural network language modeling for code switching conversational speech. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8411–8415. IEEE.

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013b. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 206–211.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*, pages 221–237.

Susan Berk-Seligson. 1986. Linguistic constraints on intrasentential code-switching: A study of spanish/hebrew bilingualism. *Language in society*, 15(3):313–348.

Irshad Ahmad Bhat, Manish Shrivastava, and Riyaz Ahmad Bhat. 2016. Code mixed entity extraction in indian languages using neural networks. In *FIRE (Working Notes)*, pages 296–297.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Khyathi Raghavi Chandu, Manoj Chinnakotla, Alan W Black, and Manish Shrivastava. 2017. Webshodh: A code mixed factoid question answering system for web. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 104–111. Springer.

Monojit Choudhury, Kalika Bali, Sunayana Sitaram, and Ashutosh Baheti. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 65–74.

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.

Anne-Marie Di Sciullo, Pieter Muysken, and Rajendra Singh. 1986. Government and code-mixing. *Journal of linguistics*, 22(1):1–24.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32*, pages 13063–13075. Curran Associates, Inc.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392. Association for Computational Linguistics.

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

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

Hila Gonen and Yoav Goldberg. 2018. Language modeling for code-switching: Evaluation, integration of monolingual data, and discriminative training. *arXiv preprint arXiv:1810.11895*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018a. A deep neural network based approach for entity extraction in code-mixed Indian social media text. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Deepak Gupta, Ankit Lamba, Asif Ekbal, and Pushpak Bhattacharyya. 2016a. Opinion mining in a code-mixed environment: A case study with government portals. In *Proceedings of the 13th International Conference on Natural Language Processing*, pages 249–258.

Deepak Gupta, Pabitra Lenka, Asif Ekbal, and Pushpak Bhattacharyya. 2018b. Uncovering code-mixed challenges: A framework for linguistically driven question generation and neural based question answering. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 119–130, Brussels, Belgium. Association for Computational Linguistics.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2016b. A Hybrid Approach for Entity Extraction in Code-Mixed Social Media Data. *MONEY*, 25:66.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. SMPOST: Parts of Speech Tagger for Code-Mixed Indic Social Media Text. *arXiv preprint arXiv:1702.00167*.

Deepak Kumar Gupta, Shubham Kumar, and Asif Ekbal. 2014. Machine learning approach for language identification & transliteration. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 60–64.

Taofik Hidayat. 2012. An analysis of code switching used by facebookers (a case study in a social network site). *Student essay for the study programme Pendidikan Bahasa Inggris (English Education) at STKIP Siliwangi Bandung, Indonesia*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386, Varna, Bulgaria. INCOMA Ltd.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

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 on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Raj Nath Patel, Prakash B Pimpale, and M Sasikumar. 2016. Recurrent neural network based part-of-speech tagger for code-mixed social media text. *arXiv preprint arXiv:1611.04989*.

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

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

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

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1543–1553.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding Language Preference for Expression of Opinion and Sentiment: What do Hindi-English Speakers do on Twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1131–1141.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. *arXiv preprint arXiv:1906.08972*.

David Sankoff and Shana Poplack. 1981. A formal grammar for code-switching. *Research on Language & Social Interaction*, 14(1):3–45.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72.

Thamar Solorio and Yang Liu. 2008a. Learning to predict code-switching points. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 973–981.

Thamar Solorio and Yang Liu. 2008b. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019a. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

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

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019b. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 271–280, Hong Kong, China. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Synthetic Code-Mixed Generation

A.1 Dataset Statistics

We create the synthetic datasets for eight different language pairs: English-Hindi (en-hi), English-Bengali (en-bn), English-Malayalam (en-ml), English-Tamil (en-ta), English-Telugu (en-te), English-French (en-fr), English-German (en-de) and English-Spanish (en-es). We used the Europarl parallel corpus (Koehn, 2005) v7[16] for the European languages, namely French, German and

[16] https://www.statmt.org/europarl/

Algorithm 1 Code-Mixed Text Generation

```
1:  Input: a parallel sentence (en-sentence, x-sentence)
2:  Output: an equivalent code-mixed sentence (en-x-sentence)
3:  procedure GETCODEMIXEDTEXT(en-sentence, x-sentence)
4:      en-tokens ← tokenize(en-sentence)                          ▷ Tokenize the English sentence
5:      x-tokens ← tokenize(x-sentence)                            ▷ Tokenize the language-x sentence
6:      alignment ← getAlignment(en-sentence, x-sentence)          ▷ Learn the alignment matrix
7:      phrases ← extractPhrase(en-tokens, x-tokens, alignment)    ▷ Phrase Extraction
8:      en-x-tokens ← x-tokens                                     ▷ Initialize the code-mixed sentence
9:      pos ← getPartsOfSpeechTags(en-tokens)                      ▷ Parts-of-speech tagging of English sentence
10:     ner ← getNERTags(en-tokens)                                ▷ NER tagging of English sentence
11:     noun-phrases ← getNounPhrase(en-tokens)                    ▷ Extraction of noun phrases
12:     for (entity, entity-type) in ner do                        ▷ Looping for each entity in English sentence
13:         if entity-type in [`PER', `LOC',`ORG'] and entity in phrases then
14:             aligned-phrase = getAlignedPhrase(phrases, entity)
15:             en-x-tokens ← en-x-tokens.replace(aligned-phrase, entity)
16:         end if
17:     end for
18:     for nphrase in noun-phrase do                              ▷ Looping for each noun phrase in English sentence
19:         aligned-phrase = getAlignedPhrase(phrases, nphrase)
20:         en-x-tokens ← en-x-tokens.replace(aligned-phrase, nphrase)
21:     end for
22:     for (token, pos-type) in pos do                            ▷ Looping for each token of English sentence
23:         if pos-type == `ADJ' and token in phrases then
24:             aligned-phrase = getAlignedPhrase(phrases, token)
25:             en-x-tokens ← en-x-tokens.replace(aligned-phrase, token)
26:         end if
27:     end for
28:     en-x-sentence ← ‘ ’.join(en-x-tokens)                      ▷ Join each token to form the code-mixed sentence
29:     return en-x-sentence
30: end procedure
```

Language Pairs	# Parallel Sentences	# Code-Mixed Sentences	Train/Dev/Test	SPF	CMI
en-es	1,965,734	200,725	196,725/2,000/2,000	68.59	28.80
en-de	1,920,209	192,131	188,131/2,000/2,000	68.41	28.26
en-fr	2,007,723	197,922	193,922/2,000/2,000	68.12	28.40
en-hi	1,561,840	252,330	248,330/2,000/2,000	62.92	23.49
en-bn	337,428	167,893	163,893/2,000/2,000	67.61	25.41
en-ml	359,423	182,453	178,453/2,000/2,000	81.84	28.13
en-ta	26,217	12,380	11,380/500/500	78.74	28.16
en-te	22,165	10,105	9,105/500/500	76.19	28.69

Table 6: Statistics of parallel corpus and generated synthetic code-mixed sentences along with the training, development and test set distributions. We also show the complexity of the generated code-mixed sentence in terms of **SPF** and **CMI**.

Spanish. For Indic languages, namely Hindi, Bengali, Malayalam, Tamil and Telugu, we obtain the parallel corpus from the multilingual parallel corpus directory[17] based on the open parallel corpus[18].

[17] http://lotus.kuee.kyoto-u.ac.jp/WAT/indic-multilingual/index.html

[18] http://opus.nlpl.eu/

We show the detailed statistics of the generated code-mixed corpus in Table 6.

A.2 Code-mixed Complexity

We measure the complexity if the generated code-mixed text in terms of the following metrics:

Switch-Point Fraction (SPF) Switch-point are the point in a sentence where the language of each side of the words are different. Following Pratapa et al. (2018); Winata et al. (2019b), we compute the SPF as the number of switch-points in a sentence divided by the total number of word boundaries. A sentence having more number of switch points are more complex as it contains many interleaving words in different languages.

Code-mixing Index (CMI) It is used to measure the amount of code mixing in a corpus by accounting for the language distribution. The sentence level CMI score can be computed with the following formula:

$$C_u(x) = \frac{N(x) - max(\ell_i \in \ell\{w_{\ell_i}(x)\})}{N(x)}, \quad (7)$$

where $N(x)$ is the number of tokens of utterance x, w_{ℓ_i} is the word in language ℓ_i. We compute this metric at the corpus-level by averaging the values for all sentences. We have reported the SPF and CMI values for all the language pairs in Table 6.