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

1 Synthetic Code-Mixed Generation

1.1 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[1] for the European languages, namely French, German and Spanish. For Indic languages, namely Hindi, Bengali, Malayalam, Tamil and Telugu, we obtain the parallel corpus from the multilingual parallel corpus directory[2] based on the open parallel corpus[3]. We show the detailed statistics of the generated code-mixed corpus in Table 1.

1.2 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. (2019), 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:

$$C_u(x) = \frac{N(x) - max(\ell_i \in \ell\{w_{\ell_i}(x)\})}{N(x)}, \quad (1)$$

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 1.

2 Results and Analysis

2.1 Network Training

The neural code-mixed generation network is trained to minimize the negative log-likelihood of the training data. We follow the most widely used method to train a decoder RNN for sequence generation, called the "teacher forcing" algorithm (Williams and Zipser, 1989). We define $y^* = \{y_1^*, y_2^*, \ldots, y_m^*\}$ as the ground-truth output sequence for a given input sequence E. The maximum-likelihood training objective is the minimization of the following loss:

$$\mathcal{L}_{mle} = -\sum_{t=1}^{m} \log p(y_t^* | y_1^*, \ldots, y_{t-1}^*, E) \quad (2)$$

2.2 Error Analysis

We also perform thorough analysis of the errors produced by the system (*en-hi*) and the way to mitigate those errors. The errors are categorized in the following types:

1. **Reference Inaccuracy:** The error in word alignment propagates and leads to the inaccurate reference code-mixed sentence. Since, we use synthetic reference code-mixed sentence to train our code-mixed generator it causes errors in the generated code-mixed sentence too. This issue can be minimized by advancing the underlying alignment algorithm.

[1] https://www.statmt.org/europarl/
[2] http://lotus.kuee.kyoto-u.ac.jp/WAT/indic-multilingual/index.html
[3] http://opus.nlpl.eu/

Algorithm 1 Code-Mixed Text Generation

```
Input: a parallel sentence (en-sentence, x-sentence)
Output: an equivalent code-mixed sentence (en-x-sentence)
procedure getCodeMixedText(en-sentence, x-sentence)
    en-tokens ← tokenize(en-sentence)                          ▷ Tokenize the English sentence
    x-tokens ← tokenize(x-sentence)                            ▷ Tokenize the language-x sentence
    alignment ← getAlignment(en-sentence, x-sentence)          ▷ Learn the alignment matrix
    phrases ← extractPhrase(en-tokens, x-tokens, alignment)    ▷ Phrase Extraction
    en-x-tokens ← x-tokens                                     ▷ Initialize the code-mixed sentence
    pos ← getPartsOfSpeechTags(en-tokens)                      ▷ Parts-of-speech tagging of English sentence
    ner ← getNERTags(en-tokens)                                ▷ NER tagging of English sentence
    noun-phrases ← getNounPhrase(en-tokens)                    ▷ Extraction of noun phrases
    for (entity, entity-type) in ner do                        ▷ Looping for each entity in English sentence
        if entity-type in [`PER', `LOC',`ORG'] and entity in phrases then
            aligned-phrase = getAlignedPhrase(phrases, entity)
            en-x-tokens ← en-x-tokens.replace(aligned-phrase, entity)
        end if
    end for
    for nphrase in noun-phrase do                              ▷ Looping for each noun phrase in English sentence
        aligned-phrase = getAlignedPhrase(phrases, nphrase)
        en-x-tokens ← en-x-tokens.replace(aligned-phrase, nphrase)
    end for
    for (token, pos-type) in pos do                            ▷ Looping for each token of English sentence
        if pos-type == `ADJ' and token in phrases then
            aligned-phrase = getAlignedPhrase(phrases, token)
            en-x-tokens ← en-x-tokens.replace(aligned-phrase, token)
        end if
    end for
    en-x-sentence ← ' '.join(en-x-tokens)                      ▷ Join each token to form the code-mixed sentence
    return en-x-sentence
end procedure
```

2. **Missing/Incorrect Words:** This is one of the common error type, where the model generated incorrect words/phrase. The missing or incorrect words cause fluency problem in the generated code-mixed sentence. We also observe that the majority of the missing words are *function words* while incorrectly generated words belong to the *content words* category. E.g.
   Generated: इस book समस्त Copyright हमारे पास हैं ।
   Gold: इस book के समस्त Copyright हमारे पास हैं ।
   (**Trans**: *Copyright of this book is owned by us.*) Here the function word के *(of)* is missing from the generated code-mixed text.
   This error can be reduced by employing the advance language model (like GPT (Radford et al., 2019), MASS (Song et al., 2019)) as decoder in the network.
3. **Factual Inaccuracy:** The model sometimes generates the factually incorrect named entities. We also observe that this type of error occur in often longer sentences, where the model is confused to copy/generate the relevant entity in the given context. E.g.
   Generated: Bluetooth stack के प्रयोग से BlueZ management|
   Gold: BlueZ stack के प्रयोग से Bluetooth management |
   (**Trans**: *Bluetooth management using the BlueZ stack.*) Here the entities *'Bluetooth'* and *'BleuZ'* are misplaces in the generated code-mixed text. The factual inaccuracy can be tackled with the inclusion of knowledge graph (Zhu et al., 2020), which will help the

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	178,453,371/2,000/2,000	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 1: 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**.

model to generate the factually correct entity at the decoding step.

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). E.g.
   Generated: एक Bill और एक Act के बीच क्या अंतर है |
   Gold: एक Bill और एक Act के बीच क्या difference है |
   (**Trans**: *What is the difference between a bill and an act?*) Here the noun word '*difference*' could not be generated by the model instead it generate the word 'अंतर'. However, according to code-mixed theory the noun word '*difference*' should be mixed to generate the code-mixed sentence.
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 training 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. It sense that, those errors can be further reduced with sentence simplification (Dong et al., 2019) or text splitting of the longer input sentence.

References

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402.

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

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.

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.

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.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

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)*, pages 271–280, Hong Kong, China. Association for Computational Linguistics.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Boosting factual correctness

of abstractive summarization with knowledge graph.
arXiv preprint arXiv:2003.08612.