INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Style Variation as a Vantage Point for Code-Switching

Khyathi Raghavi Chandu, Alan W Black

Language Technologies Institute
Carnegie Mellon University
kchandu@cs.cmu.edu, awb@cs.cmu.edu

Abstract

Code-Switching (CS) is a prevalent phenomenon observed in bilingual and multilingual communities, especially in digital and social media platforms. A major problem in this domain is the dearth of substantial corpora to train large scale neural models. Generating vast amounts of quality synthetic text assists several downstream tasks that heavily rely on language modeling such as speech recognition, text-to-speech synthesis etc,. We present a novel vantage point of CS to be style variations between both the participating languages. Our approach does not need any external dense annotations such as lexical language ids. It relies on easily obtainable monolingual corpora without any parallel alignment and a limited set of naturally CS sentences. We propose a two-stage generative adversarial training approach where the first stage generates competitive negative examples for CS and the second stage generates more realistic CS sentences. We present our experiments on the following pairs of languages: Spanish-English, Mandarin-English, Hindi-English and Arabic-French. We show that the trends in metrics for generated CS move closer to real CS data in the above language pairs through the dual stage training process. We believe this viewpoint of CS as style variations opens new perspectives for modeling various tasks in CS text.

Index Terms: code-switching, style transfer, non-parallel data, adversarial training

1. Introduction

Code-Switched [1] text is prevalent in semi-formal and informal communication platforms. A major challenge in addressing this widely observed form of mixing languages is the scarcity of curated data, thereby making it a low resource setting [2]. However, there are plenty of monolingual corpora available for each of the participating languages. We present a novel standpoint to transfer knowledge from monolingual corpora without additional annotations such as language ids or parse trees. The recent advances in cross-lingual pretrained language models [3, 4] call out for vast amounts of code-switched data. Hence, our work on automatic generation of CS text is a crucial precursor for several downstream tasks including Text to Speech, Automatic Speech Recognition etc.,

We propose a novel vantage point for CS as stylistic variation between the participating embedded and matrix languages. For the scope of this paper, we define the style variations between languages to be extrinsic properties such as surface lexical forms and intrinsic properties such as underlying grammar, word order etc,. We address this problem with adversarial training in two stages: (1) *Stage 1:* transfer the style of each of the monolingual participating languages into the content of the other language; (2) *Stage 2:* transfer the style of incorrectly switched to naturally switched sentences. Hence, the four styles in play here are the following: (1) l_m: matrix language style (2) l_e: embedded language style (3) l_a: incorrect/artificial code-switching style (4) l_n: natural code-switching style. The goal is to traverse smoothly across these styles without affecting the content. The first stage generates negative examples facilitating the discriminative training for the second stage. This dual stage training eliminates the need for additional linguistic annotations, such as lexical language ids used by several contemporary works. We present our results on four pairs of languages.

2. Related Work

Constraint Theory based Generation: [5] combined syntactic constraints by predicting language boundary to reconstruct CS text. [6] and [7] present techniques based on Equivalence Theory [8] and Matrix Language Frame Theory [9] to create grammatically valid CS text. While these methods demonstrate the use of expert knowledge to assist generation, the same is difficult to replicate and scale to other languages.

Language Informed Modeling: Prior works rely on annotations of language spans in multi-task setup [10] or using dual RNN to handle each language [11]. Generated sentences are used to pretrain the model which essentially is augmenting the original data with generated data. As a result, [11] train on a lot of CS data to generate new CS text. Our approach relies primarily on a lot of monolingual data in Stage 1 and some amount of CS data in Stage 2. Capturing syntactic and language switching signals prove effective in a hierarchical VAE architecture [12]. [13] proposed a GAN based approach to generate language id tags and discriminate whether it is a valid sequence. The similarity between this and our works is that the fundamental architecture is a GAN. Lexical level translation is needed by [13] which cannot be done by a simple word lookup but depends on the context. The language id based lookup for translation may not perform well in all cases especially when transliteration is also needed.

Dependence on Parallel data: [14] proposed a seq-to-seq model with copy mechanism limiting the method to rely on parallel monolingual translations of CS text. However, more often there might be limitations on gathering parallel data not only across languages but in this case also with CS text. There has been some prior work in style transfer techniques for non-parallel data such as [15, 16].

Our approach eliminates the need for drafting constraint theories, additional annotations for language ids and parallel data. This enables scalability to new language pairs attributed to the availability of monolingual corpora and limited CS text.

3. Datasets

Each of the participating monolingual utterances (belonging to $\mathbb{M}$ (matrix language) and $\mathbb{E}$ (embedded language)) are treated as two distinct styles. Note that the sentences are not aligned either at phrase or sentence levels. We explored CS for four

http://dx.doi.org/10.21437/Interspeech.2020-2574

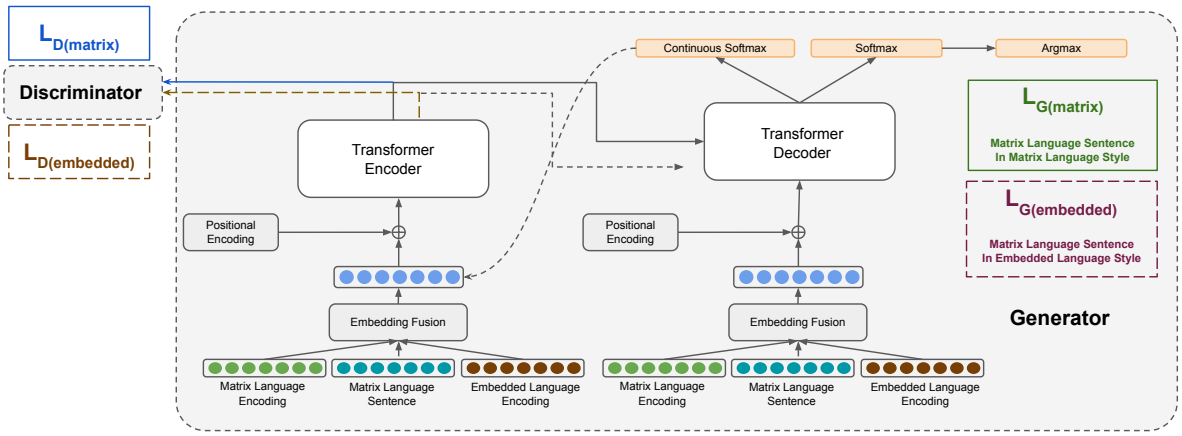

Figure 1: *Transformer based GAN architecture for generating CS text. Note: The same architecture is used for two stages. Matrix language sentence is the embedding of the text and language encoding is the embedding of the language.*

language pairs as presented in Table 1.

The reasons behind selecting these language pairs are multi-step. We select Hinglish and Spanglish since they are widely spoken. The usage of Hindi in Hinglish is commonly romanized, bringing in a new cross-script variety. Spanglish and Hinglish thus have very close scripts as opposed to Mandarin-English where the scripts are different. While the word order for English, Spanish, Mandarin and French is subject-verb-object (SVO), the same for Hindi is subject-object-verb (SOV) and Arabic is verb-subject-object (VSO). These differences facilitate the stylistic attributes to the mixing of these languages.

Language	Monolingual (Stage 1)	Code-Switched (Stage 2)
Spanish English	[17] [19]	[18]
Mandarin English	[20] [19]	[21]
Hindi English	[22] [22]	[22]
Arabic French	[23] [25]	[24]

Table 1: *Monolingual and Code-Switched Datasets used for training Stage 1 and Stage 2*

4. Model Description

The two problems we address are repealing the need for annotations (such as language id) on CS data and maximizing the utilization of monolingual data. Both the issues are addressed using a two stage generative adversarial training paradigm with a transformer based autoencoder. Unavailability of parallel sentences is tackled by preserving semantics of the original sentence of one language and mixing the attributes of the other language without disentangling the representation into these two properties. Following are the two stages involved:

Stage 1: The embedded and matrix languages are mixed in arbitrary ways to generate CS text. This stage simply uses the corpora from each language as an individual style.

Stage 2: The sentences generated after Stage 1 were not supervised via any real CS sentences. Hence, they are used as negative examples (with style l_a) against limited amount of CS text (with style l_n) to generate naturally switched sentences.

The model architecture remains same for both stages except for variation in training data. Figure 1 presents our GAN setup for Stage 1. The following subsections present the flow of our model by instantiating for Stage 1 for readability. The same process is applied for Stage 2 with the difference of using positive and negative examples of CS sentences.

4.1. Generator

The generator (G_θ) in our architecture comprises of transformer based encoder and decoder. In Stage 1, our transformer encoder takes in the matrix language sentence ($s_m \in \mathbb{M}$) along with the matrix language encoding or style (l_m) and produces a latent representation ($z_{m,m}$).

$$z_{m,m} = TransEnc(s_m, l_m) \forall s_m \in \mathbb{M} \quad (1)$$

We use this $z_{m,m}$ along with the original matrix language sentence s_m and the matrix language encoding l_m to reconstruct the original sentence s_m. Greedy decoding is performed that uses argmax which is non-differentiable to compute the loss for reconstructing the original sentence ($L_{G_\theta(matrix)}$).

$$L_{G_\theta(matrix)} = -\sum_{s_m \in \mathbb{M}} log(Pr(s_m|s_m, l = m)) \quad (2)$$

Next, the same matrix language sentence s_m is considered along with the embedded language encoding l_e to produce a latent representation ($z_{m,e}$).

$$z_{m,e} = TransEnc(s_m, l_e) \forall s_m \in \mathbb{M} \quad (3)$$

This $z_{m,e}$ is used to reconstruct the original sentence s_m. This means that the model is attempting to reconstruct the content of the original sentence while varying the style i.e, language encoding. The loss corresponding to this reconstruction is ($L_{G_\theta(embedded)}$).

$$L_{G_\theta(embedded)} = -\sum_{s_m \in \mathbb{M}} log(Pr(s_m|s_m, l = e)) \quad (4)$$

Similarly, corresponding counterparts using $z_{e,e}$ and $z_{e,m}$ are generated.

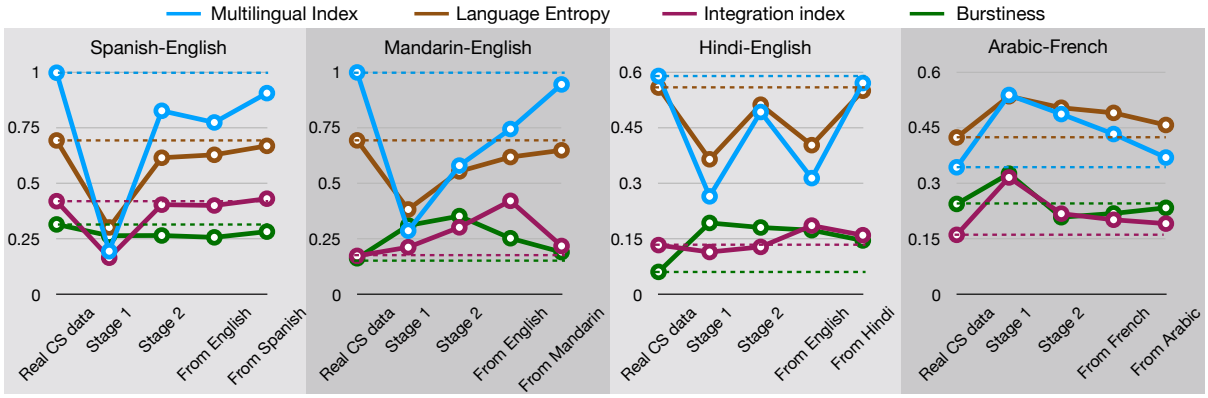

Figure 2: *Trends in metrics for evaluating the generation of CS text for four language pairs in dual stage training. The dotted line of each color benchmarks the corresponding metric for real CS data.*

4.2. Discriminator

The discriminator (D_ϕ) is a classifier that predicts whether the current distribution is closer to the original latent space or the generated latent space. The purpose of generator reconstructing the original sentence s_m with matrix language encoding l_m (contributing to $L_{G_\theta(matrix)}$) is solely to make sure that the generator is retaining the content of the original sentence, and has no contribution towards training the discriminator. On the other hand, the generation of the sentence s_m with embedded language encoding l_e, say $s_{m,e}$ essentially establishes our end goal. In the GAN architecture, we now have two choices i.e, sampling a sentence from: (1) original distribution $s_{m,m}$ i.e, the matrix language sentence with the matrix language encoding or (2) distribution from the generator $s_{m,e}$ i.e, the matrix language sentence with the embedded language encoding. The positive examples to train the discriminator come from real sentences which are trained by maximizing the probability for predicting that it belongs to label *m*.

$$L_{D_\phi(matrix)} = -\sum log(Pr(m|z_{m,m}, l = m)) \quad (5)$$

A crucial problem for training GANs in text domain is the non-differentiable function of argmax that is performed in decoding. There are three prominent solutions to address this problem including REINFORCE [26], Gumbel-Softmax [27], manipulating the latent space. We proceed with the third option by performing a continuous softmax of the words, thus eliminating the need to perform argmax. Let the vocab size be $\mathcal{V}$ and the embedding dimension be $\mathcal{H}$. Instead of discretely making a selection of the embedding over the vocabulary space to select each word, we perform continuous softmax. The final softmax layer in decoder provides us with a vector of size $1 \times \mathcal{V}$. Multiplying this with the embedding weights ($\mathcal{V} \times \mathcal{H}$) results in $1 \times \mathcal{H}$ vectors for each word. Note that in the case of argmax, we make a discrete selection of the word, whereas, in the case of continuous softmax, we arrive at a soft representation of the weighted combination of properties of the words across different words in the vocabulary. Therefore the latter does not enforce this soft representation to be a word. This partially decoded representation now passes through the transformer encoder to arrive at a latent representation to be fed into the discriminator and the overall training objective are:

$$L_{D_\phi(embedded)} = -\sum log(Pr(e|z_{m,e}, l = e)) \quad (6)$$

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim\mathbf{p}_{data}}[log\mathbf{D}_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[log(1 - \mathbf{D}_\phi(\mathbf{G}_\theta(\mathbf{z}))]$$

4.3. Dual Stage Training Setup:

The task of generating CS text not only entails mixing languages but also mixing them appropriately. This means that our discriminator performs two tasks of discriminating between: (1) the participating languages, owing to the asymmetry between their interactions, such as matrix and embedded languages (Stage 1) and (2) incorrectly and correctly switched languages (Stage 2). Hence we dissolve the training procedure into two stages with dedicated objectives.

In Stage 1, the sentences from $s_m \in \mathbb{M}$ are transferred to the style of l_e ($s_{m,e}$). At any given point of time, there is only one sentence and one language style that is encoded as presented in equations 1 and 3. We conflated this information while presenting in Figure 1. For example, let the subscript 'r' be reconstructed sentence from an original source sentence. In addition to the sentence, there is one style that is taken as input. This style could be either (only one among) l_m or l_e styles which is encoded. Let the source and the matrix language styles be s_m and s_e respectively. The discriminator labels the following representations in the corresponding ways: (i) Label 0: (a) s_m with l_m style; (b) 's'_{mr} with l_m style; (ii) Label 1: (a) s_e with l_e style; (b) 's'_{er} with l_e style; In this way, we train a model to generate $s_{m,e}$ and $s_{e,m}$ which are matrix language sentences in the embedded language style and embedded language sentences in the matrix language style. Since there is no supervision from naturally CS sentences, we delegate this responsibility to the Stage 2 of training with the same architecture. We used $s_{m,e}$ as negative examples of CS sentences for Stage 2 of training. We have also experimented with a random subset of $s_{m,e}$ and $s_{e,m}$ as negative examples. This performed worse than the former setting since $s_{m,e}$ has the underlying grammatical structure of $\mathbb{M}$, thereby generating stronger negative examples for adversarial training. In Stage 2, the negative examples generated from Stage 1 belongs to incorrect or artificial CS style (l_a) because the words are mixed arbitrarily from both the languages. The style from real CS data is the real or natural CS style (l_n) in which we want to finally generate CS text. The same model is trained in the same way described above again with these two styles in this second stage. Note that the training objective is also the same in the second stage as the first stage. The goal

of the first stage is to generate robust negative examples of CS style to train the model in the second stage.

Hyperparameter setup: We used 3 layers of transformer encoders and decoders with a maximum sequence length of 45 words. The word embedding dimension is 256 with 300 iterations of pre-training the generator before training our GAN. In Stage 1, there is minimal overlap of vocabulary between the languages. This is in contrary to data in typical style transfer datasets which have overlapping vocabulary spaces. Hence the discriminator learns much faster than generator in our case. To combat this, the learning rate in Stage 1 for the generator is 1e-3 and the discriminator is 1e-4. For Stage 2, the generator and discriminator are initialized with models learnt in Stage 1 thereby transferring the knowledge of each of the languages. However, to quickly adapt to the parameter space of Stage 2, we use slanted triangular learning rate [28] with a short linear increase period followed by longer decay period. Adam optimizers are used throughout the model. We plan to release our code, models and generated samples upon acceptance.

5. Evaluation

We present our results on four pairs of languages from Table 1. We evaluate trends in different metrics of CS proposed by [29] in our dual stage training. Consolidated results are presented in Figure 2. Metrics that we look into are *multilingual-index*, *language entropy*, *integration-index* and *burstiness*. In Figure 2, 'Stage 1' contains generated sentences $s_{m,e}$. The model learnt in Stage 2 has options to generate from negative examples or original text of each language as source. 'Stage 2' contains sentences generated using negative examples from Stage 1 as source. Similarly, 'From $<$*lang*$>$' uses the corresponding language as source to transfer style to real CS. We observe that the metrics move closer to real CS in 'Stage 2' as compared to 'Stage 1'. Within 'Stage 2', metrics are closer to real CS data when the source text belongs to $\mathbb{M}$ in comparison to $\mathbb{E}$ or $s_{m,e}$ from Stage 1. We plan to explore properties of syntactic and semantic mixing in each stage in our future work.

5.1. Human Evaluation

We conduct human evaluation in the form of preference testing between groups of sentences with 10 human subjects for generated Hinglish each with 10 batches of sentences. Each batch has 2 sets of sentences (each set having 5 sentences). The first set is from Stage 1 and the second set is from Stage 2 which are jumbled randomly. The preference testing is done at the set level. We ask the human evaluators to select a set that *'seems more natural'*. The instances are grouped lengthwise i.e, among 10 instances: *short-length* in the range of 3-5 words (2 sets), *medium-length* in the range of 6-10 words (5 sets), *long-length* in the range of 11-15 words (3 sets).

Overall, the sentences generated from Stage 2 are preferred 74% of the times. Dissecting them length wise Stage 2 is preferred 60% in short-length, 84% in medium-length and 67% in long-length. This shows that, though Stage 2 generation is clearly preferred for all ranges of sentence lengths, it is better in generating longer sentences in comparison to shorter sentences.

5.2. Qualitative Analysis

The following are some of the common forms of errors observed in the generated text for Hinglish when trained on the blogging data collected from [10] on which similar trends in results from Hinglish in Fugure 2 were observed.

• *Gender Disagreement:* For instance, consider the sentence 'kyunki ye scam bhi ho sakti hai' (Meaning: because this can also be a scam). The gender of the direct object which is 'scam' should agree with the inflection of the verb 'sakti'. Hence this should have been 'sakta'. The gender of the word 'scam' (which is a borrowed word from embedded language English) is unknown in the matrix language, so it would be presumed to be masculine. But this sentence used a feminine verb phrase.

• *Incorrect Case markers:* 'agar aap bhi ye post pasand aaye toh aap puch sakte hai' (Meaning: If this post is pleasing to you also, then you can ask). The words 'to you' is supposed to be in dative case which is 'aap ko' in the first clause of the sentence. However 'aap' which is in the nominative case is generated.

• *Semantically incorrect - random mixing:* Sometimes, the model also generates completely random mix of words. For instance, this is one of the sentence from the output: 'am bahut hi achchi jankari aapko pata hi hoga' (Loosely Translated Meaning: very good information you must be knowing). The sentence does not convey a coherent meaning. The word order of the sentence is also jumbled and does not strictly belong to either of the matrix or the embedded languages.

• *No mixing:* In some cases, the entire sentence is built from words belonging to the same language. For instance, 'the best way to improve your application for the reasons listed below'.

• *Incorrect sub-word mixing:* Though the incorrect case markers and gender disagreement are syntactic errors, this seems to happen due to the modeling at word level. Sub-word level modeling of text is a promising direction to address this error category especially for morphologically rich languages.

The category of first two error types described above are syntactic. Our current model is purely data driven from the surface forms. This motivates the utility of inducing syntax while generation. This also encourages the case to model sub-word level modeling especially for morphologically rich languages. The semantically incorrect sentences seem to be generated due to random mixing of the words from both the languages. The loosely translated meaning of the sentence is not utterly senseless but when framed in CS fashion does not make sense. In addition, there is an inherent challenge while dealing with multiple datasets that lead upto domain variation. For instance, the vocabulary or the style on social media platforms such as Twitter is very different from the domain of conversations. Although we have carefully selected the datasets to belong to similar domains, this might often not be feasible which invites the domain invariant modeling of text.

6. Conclusion

We present a novel perspective of viewing CS as style variants between participating languages. We believe this viewpoint opens new avenues for dealing with mixed language text. The main contributions of the paper are threefold. Firstly, we eliminate the need for explicit language identification using two stage adversarial training. Secondly, our approach transfers from bountiful monolingual resources and relies on limited CS data to generate new CS sentences. Thirdly, we present our experiments on dual stage transformer based GAN model for generating four pairs of CS languages: Spanish-English, Mandarin-English, Hindi-English and Arabic-French. In future, we would like to compare the performance of this technique with other style transfer models and also explore the possibility of end-to-end training of both the stages. We are also continuing working on utilizing the generated data to pre-train models to perform downstream tasks.

7. References

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

[2] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *arXiv preprint arXiv:1904.00784*, 2019.

[3] M. Artetxe and H. Schwenk, "Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 597–610, 2019.

[4] A. Conneau and G. Lample, "Cross-lingual language model pretraining," in *Advances in Neural Information Processing Systems*, 2019, pp. 7057–7067.

[5] Y. Li and P. Fung, "Code-switch language model with inversion constraints for mixed language speech recognition," in *Proceedings of COLING 2012*, 2012, pp. 1671–1680.

[6] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, and K. Bali, "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)*, 2018, pp. 1543–1553.

[7] G. Lee, X. Yue, and H. Li, "Linguistically motivated parallel data augmentation for code-switch language modeling," *Proc. Interspeech 2019*, pp. 3730–3734, 2019.

[8] S. Poplack, "Sometimes i'll start a sentence in spanish y termino en espanol: toward a typology of code-switching1," *Linguistics*, vol. 18, no. 7-8, pp. 581–618, 1980.

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

[10] K. Chandu, T. Manzini, S. Singh, and A. W. Black, "Language informed modeling of code-switched text," in *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, 2018, pp. 92–97.

[11] S. Garg, T. Parekh, and P. Jyothi, "Code-switched language models using dual rnns and same-source pretraining," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3078–3083.

[12] B. Samanta, S. Reddy, H. Jagirdar, N. Ganguly, and S. Chakrabarti, "A deep generative model for code-switched text," *arXiv preprint arXiv:1906.08972*, 2019.

[13] C.-T. Chang, S.-P. Chuang, and H.-Y. Lee, "Code-switching sentence generation by generative adversarial networks and its application to data augmentation," *arXiv preprint arXiv:1811.02356*, 2018.

[14] G. I. Winata, A. Madotto, C.-S. Wu, and P. Fung, "Code-switched language models using neural based synthetic data from parallel sentences," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 271–280.

[15] Z. Yang, Z. Hu, C. Dyer, E. P. Xing, and T. Berg-Kirkpatrick, "Unsupervised text style transfer using language models as discriminators," in *Advances in Neural Information Processing Systems*, 2018, pp. 7287–7298.

[16] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Advances in neural information processing systems*, 2017, pp. 6830–6841.

[17] D. Graff, S. Huang, I. Cartagena, K. Walker, and C. Cieri, "Fisher spanish transcripts ldc2010t04," *Web Download, Philadelphia, USA*, 2010.

[18] M. Deuchar, P. Davies, J. R. Herring, M. Parafita Couto, and D. Carter, "Building bilingual corpora," 2014.

[19] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic, "Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 5016–5026.

[20] L. Tian, D. F. Wong, L. S. Chao, P. Quaresma, F. Oliveira, and L. Yi, "Um-corpus: A large english-chinese parallel corpus for statistical machine translation," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, 2014.

[21] D.-C. Lyu, T.-P. Tan, E. S. Chng, and H. Li, "Seame: a mandarin-english code-switching speech corpus in south-east asia," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[22] P. Mathur, R. Shah, R. Sawhney, and D. Mahata, "Detecting offensive tweets in hindi-english code-switched language," in *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, 2018, pp. 18–26.

[23] Z. Song, S. M. Strassel, H. Lee, K. Walker, J. Wright, J. Garland, D. Fore, B. Gainor, P. Cabe, T. Thomas *et al.*, "Collecting natural sms and chat conversations in multiple languages: The bolt phase 2 corpus." in *LREC*. Citeseer, 2014, pp. 1699–1704.

[24] R. Cotterell, A. Renduchintala, N. Saphra, and C. Callison-Burch, "An algerian arabic-french code-switched corpus," in *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, 2014, p. 34.

[25] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5. Citeseer, 2005, pp. 79–86.

[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[28] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.

[29] G. A. Guzmán, J. Ricard, J. Serigos, B. E. Bullock, and A. J. Toribio, "Metrics for modeling code-switching across corpora." 2017.