Exploiting Monolingual Speech Corpora for Code-mixed Speech Recognition

Karan Taneja[1], *Satarupa Guha*[2], *Preethi Jyothi*[1], *Basil Abraham*[2]

[1]Indian Institute of Technology Bombay, Mumbai, India
[2]Microsoft India Development Center, Hyderabad, India
{karantaneja,pjyothi}@iitb.ac.in, {saguha,baabraha}@microsoft.com

Abstract

One of the main challenges in building code-mixed ASR systems is the lack of annotated speech data. Often, however, monolingual speech corpora are available in abundance for the languages in the code-mixed speech. In this paper, we explore different techniques that use monolingual speech to create synthetic code-mixed speech and examine their effect on training models for code-mixed ASR. We assume access to a small amount of real code-mixed text, from which we extract probability distributions that govern the transition of phones across languages at code-switch boundaries and the span lengths corresponding to a particular language. We extract segments from monolingual data and concatenate them to form code-mixed utterances such that these probability distributions are preserved. Using this synthetic speech, we show significant improvements in Hindi-English code-mixed ASR performance compared to using synthetic speech naively constructed from complete utterances in different languages. We also present language modelling experiments that use synthetically constructed code-mixed text and discuss their benefits.

Index Terms: code-mixed speech recognition, synthetic code-mixed speech from monolingual data

1. Introduction

Code-mixing (CM) is a linguistic phenomena that is prevalent in multilingual communities where speakers alternate between languages within a sentence or a discourse. Given the widespread use of CM among multilingual speakers, automatic speech recognition (ASR) for CM speech is of great interest. However, labeled CM speech is a scarce resource, thus making ASR for CM very challenging.

Although labeled CM speech is not easily available, monolingual labeled corpora are typically available in large quantities for the individual languages in CM speech. Assuming access to a small amount of CM text and no CM speech, can we leverage the monolingual speech data to improve recognition on CM speech? This is the main problem we tackle in this work. Using a small amount of real CM text as our reference, we compute probability distributions corresponding to a) phone pairs that appear at switch points when we transition from one language to another and b) span lengths corresponding to each language. We then construct synthetic CM speech (out of monolingual speech fragments) such that these probability distributions are mostly preserved. With such constraints in place, we aim at creating synthetic CM speech that mimics important characteristics of real CM speech and captures acoustic properties at switch points across languages which are non-existent in monolingual speech.

Specifically, our contributions are summarized as follows:

- We propose two algorithms to create synthetic CM speech that preserves span length distributions and phone transition probability distributions at switch points.
- Using real CM Hindi-English speech for evaluation, we empirically investigate the effect of using synthetic CM speech to train acoustic models.
- We use transcripts from the synthetic CM speech to train language models (LMs) and examine their effect on ASR performance.

Related Work: There is a large body of prior work on ASR for CM speech [1, 2, 3]. This includes techniques specifically targeting the acoustic model [1, 2] and the language model [4, 5, 6] to handle code-mixing in speech. Apart from these cascaded ASR systems, there is also recent work on using end-to-end systems trained on multilingual data to recognize CM speech [7, 8, 9, 10, 11, 12].

Leveraging monolingual sentences for code-mixed language models has been extensively studied in prior work [13, 14, 15, 16, 17]. However, creating synthetic CM speech to improve ASR for CM data is relatively less explored. In [18], complete utterances for each language in the CM speech are concatenated together to create synthetic code-switched utterances that are subsequently used to train an end-to-end ASR system. Recent work has also explored the use of a semi-supervised approach where non-parallel CM speech and text are used by an ASR and TTS system within an autoencoder [19].

2. Our approach

We generate synthetic CM data in two stages. We first fragment monolingual utterances into smaller speech segments and subsequently concatenate these segments using different techniques in order to construct synthetic CM speech.

Segmenting monolingual utterances: From monolingual utterances in Hindi and English, we extract smaller segments with the help of a simple amplitude-based silence detector; these segments mostly end at word boundaries. The transcriptions corresponding to these segments are obtained using forced alignments that are derived from monolingual ASR systems trained for Hindi and English.

2.1. Characterizing CM speech

We first discuss some definitions that will be used subsequently. In a CM sentence, *switch point* is a word boundary where the following word is from a different language than the preceding word. A series of monolingual tokens between two consecutive switch points is called a *language span*. The number of tokens in a language span is called its *span length*. The *span length distribution* is a discrete probability distribution of span lengths in a CM corpus. We additionally define *language span length distribution* as the probability distribution of span lengths corresponding to a particular language in a CM corpus.

Algorithm 1: Algorithm to create a synthetic CM corpus according to given span length and sentence length distributions.

Input: sentence length distribution $\mathcal{D}_{\texttt{sent}}$, span length distributions of two languages $\mathcal{D}^0_{\texttt{span}}$ and $\mathcal{D}^1_{\texttt{span}}$, monolingual fragment corpora $\mathcal{F}^0_\ell$ and $\mathcal{F}^1_\ell$ for the two languages, consisting of (speech, transcript) pairs of length ℓ fragments (for each ℓ in the support of $\mathcal{D}^0_{\texttt{span}}$ and $\mathcal{D}^1_{\texttt{span}}$, respectively); a desired corpus size, n; a bound on number of times a fragment can be used, d.
Output: A corpus of n synthetic CM (speech, transcript) pairs

```
corpus ← empty;
while size of corpus < n do
    trans, speech ← empty;
    sent_length ← sample(D_sent) ;
    current_lang ← sample(Bernoulli(0.5));
    while length of trans < sent_length do
        span_length ← sample(D_span^current_lang);
        (speech_frag, trans_frag) ← uniformly
          from F_span_length^current_lang. (If there are such fragments
          selected less than d times, restrict to them);
        append speech_frag to speech;
        append trans_frag to trans;
        current_lang ← 1 − current_lang;
    end
    add (speech, trans) to corpus
end
return corpus
```

From the perspective of definitions given in [20], we can show that preserving the ratio of number of tokens of each language preserves M-Index and language entropy. Further, preserving the language span length distribution of all languages additionally preserves I-index, burstiness and span entropy. Our first CM corpus, called *SynSL*, is generated to preserve language span length distributions which implicitly preserves the five metrics mentioned above. We note that preserving span length distributions gives appropriate weightage to triphones from the individual languages in CM speech.

A *switch point phone transition* (SPT) is an ordered pair of phones, made up of the last phone of the last word in a span and the first phone of the first word in the following span. Let $\langle s\rangle$ and $\langle /s\rangle$ denote the start- and end-of-sentence tags which are treated as single-phone words, occurring as single-word spans. We shall construct CM sentences involving two languages L_0 and L_1, with phone-sets Π_0 and Π_1 respectively. We shall consider Π_0 and Π_1 to be disjoint (by annotating the phones with the language identifier), except for $\langle s\rangle$ and $\langle /s\rangle$ which are included in both. Then, the set of all possible SPTs is a subset of $(\Pi_0 \times \Pi_1) \cup (\Pi_1 \times \Pi_0)$. The empirical *SPT distribution* over a corpus, denoted by $P(x, y)$ where $(x, y) \in (\Pi_0 \times \Pi_1) \cup (\Pi_1 \times \Pi_0)$, is calculated by aggregating all the SPTs in the corpus, to form a discrete probability distribution. We shall only use the conditional probabilities of the form $P(y|x) = P(x, y)/\sum_{y'} P(x, y')$ (for x with a positive denominator).

We also define *fragment phone transition* (FPT) as an ordered pair of phones made up of first phone of the first word and last phone of the last word of a language span. The set of all possible FPTs is a subset of $(\Pi_0 \times \Pi_0) \cup (\Pi_1 \times \Pi_1)$.

Algorithm 2: Algorithm to create a synthetic CM corpus according to given switch point phone transition and fragment phone transition distributions.

Input: SPT distributions $P(y|x)$ and FPT distributions $Q(y|x)$, for all $x, y \in \Pi_0 \cup \Pi_1$; monolingual fragments; parameters n (required corpus size) and d (a bound on number of times a fragment is used).
Output: A corpus of n synthetic CM (speech, transcript) pairs

```
corpus ← empty;
while size of corpus < n do
    trans, speech ← empty;
    p_s ← sample(P(p_s|⟨s⟩));
    while p_f ≠ ⟨/s⟩ do
        p_e ← sample(Q(p_e|p_s));
        (speech_frag, trans_frag) ← a random
          fragment starting with phone p_s and ending in
          p_e. (If there are such fragments selected less
          than d times, restrict to them);
        append speech_frag to speech;
        append trans_frag to trans;
        p_s ← sample(P(p_s|p_e));
    end
    add (speech,trans) to corpus
end
return corpus
```

We can also define a corresponding empirical *FPT distribution*, $Q(x, y)$ and conditional probabilities $Q(y|x)$, for FPTs $(x, y) \in (\Pi_0 \times \Pi_0) \cup (\Pi_1 \times \Pi_1)$. Our second CM corpus, called *SynPT*, is generated to preserve the two PT distributions (i.e., SPT and FPT) of a given corpus. Though the transition probabilities at switch points are characterized by SPT distribution only, FPT distribution is important for the algorithm to generate the desired SPT distribution as described next.

2.2. Algorithms for generating CM corpora

The algorithm to generate a corpus that preserves SL distribution is given in Algorithm 1. It simply samples span lengths alternately for the two languages, and randomly samples fragments of matching lengths from a given corpus. The minimum length of each sentence is also sampled from a given distribution obtained from real CM text.

Algorithm 2 is used to create a synthetic corpus that matches specified phone transition distributions SPT and FPT. This algorithm models the sampling of fragments as a Markov chain Monte Carlo sampling. Its input is $P(y|x)$ and $Q(y|x)$, the SPT and FPT (conditional) distributions. The sampling procedure consists of alternately sampling the second phone conditioned on the first phone from the SPT and the FPT distributions. This process can be modeled as a single Markov chain, with each state being a phone, annotated with whether it is at the beginning or the end of a span (except for $\langle s\rangle$ and $\langle /s\rangle$ which occur only once each). We add an edge from the state $\langle /s\rangle$ to the state $\langle s\rangle$ with probability 1 (corresponding to starting a new sentence after one ends) so that the entire sampling algorithm can be considered a single random walk, starting from the state $\langle s\rangle$. The Markov chain formed here is ergodic and its stationary distribution will match the PT distributions of the original corpus. Figure 1 shows 30 phone-pairs with the most frequent SPTs in real CM text and in synthetic CM text; we see that the

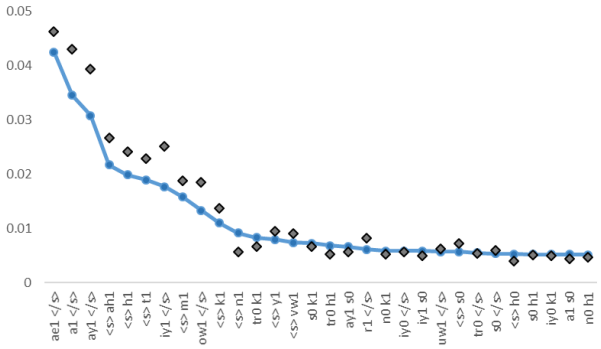

Figure 1: *Probabilities of 30 most frequent SPTs in CMtext (line) and their probabilities is SynPT (dots).*

probabilities are fairly well-matched.

In implementing both the algorithms, we impose a restriction that a fragment can be sampled at most 3 times unless there is no fragment satisfying the given constraint(s) among available fragments. In case of the latter, in Algorithm 1, a span length closest to the desired span length was chosen and in Algorithm 2, a fragment corresponding to a random phone pair was chosen; this exception was invoked rarely for the corpus sizes used. We also require that $P(\langle/s\rangle|\langle s\rangle) = 0$ (i.e. no empty sentences are allowed).

3. Experimental setup

Data description: Table 1 provides details of the datasets employed in this work. We make use of monolingual Hindi (**HI**) and Indian English (**EN**) speech corpora. (An additional 220 hours of English data was used to train the forced alignment model used in creating synthetic speech.) For the purpose of evaluation, we use real CM speech to create a development set (**Dev**) and a test set (**Test**), each amounting to more than 2 hours of speech. We also assume access to 12,600 sentences of real CM text (**CMtext**), containing $183,000$ tokens, that is used to train LMs and guide the creation of our synthetic CM speech. All the data sources are proprietary to Microsoft. We note that all the speech data used in our experiments are conversational in nature, with noisy transcriptions consisting of misspellings, many named entities and English words inconsistently rendered in either Roman or Devanagari scripts.

Implementation details: All our ASR systems were built using the Kaldi toolkit [21]. We used the `wsj/s5` scripts to train tied-state triphone models using MFCC features. The alignments from these models were subsequently used to train time-delay neural network (TDNN) [22] models, using `nnet3` scripts in Kaldi. We used a common phone set of 55 phones and proprietary lexicons. The LMs in all subsequent experiments (unless specified otherwise) are trigram models with Kneser-Ney smoothing, trained using SRILM toolkit [23].

4. Experiments and results

4.1. Impact of synthetic speech on acoustic models

All the systems shown in Table 2 use a smoothed trigram LM trained on CMtext. For the acoustic model, we build a baseline monolingual Hindi TDNN model that is trained on 50 hours of speech randomly chosen from HI. We refer to this system as "HI(50)", shown in Table 2. We also trained a TDNN model using 50 hours from HI and 50 hours from EN, which performed worse than HI(50) with WERs of 64.60 and 65.85 on the dev and test sets, respectively.[1] Hence, in all subsequent experiments in this section, we use monolingual Hindi data as our base corpus to which synthetic data is added. For synthetic CM speech, in addition to SynSL and SynPT described in Section 2.2, we also generate "SynConcat" inspired by prior work [18] where up to three entire utterances in Hindi and English are concatenated together.

Table 1: *Statistics of our data sets*

Corpus	Type	# of hours	# of utterances
HI	Speech	353	42.4k
EN	Speech	133	24.7k
CMtext	Text	-	12.6k
Dev	Speech	2.2	2k
Test	Speech	2.3	2k

In the first set of numbers in Table 2, with HI(50) as our starting point, we show the effect of augmenting the training data with 100 hours of different kinds of synthetic CM speech. We make the following observations: 1) Adding 100 hours of synthetic data improves WERs when compared to HI(50), regardless of how the synthetic speech was generated. These improvements are statistically significant (at $p < 0.001$) using the mapswwe test[2]. 2) Adding 100 hours of SynPT is more beneficial compared to adding 100 hours of SynConcat. The difference in WERs on both dev and test sets using HI(50)+SynPT(100) when compared with HI(50)+SynConcat(100) are statistically significant at $p < 0.001$. Using SynPT also helps improve performance slightly more than using SynSL. These results show that synthetic speech that mimics the characteristics of real CM speech is more useful than naively constructed synthetic speech when starting with a modest baseline (i.e. a monolingual ASR system trained on 50 hours of Hindi speech).

In the second set of numbers in Table 2, we start with a TDNN acoustic model trained on 350 hours of Hindi speech (HI(350)). This gives significantly lower WERs on the dev and test sets compared to HI(50). Next, we examine the effect of augmenting HI(350) with 100 hours of synthetic data. As with HI(50), adding synthetic data to HI(350) continues to improve WERs significantly (at $p = 0.02$ for dev and $p < 0.001$ for test). However, the manner in which synthetic data is generated is less important when starting with HI(350); SynConcat continues to be worse than SynPT albeit by a smaller margin.[3]

The final set of numbers in Table 2 shows how WERs change as a function of the amount of SynPT data used during training. As expected, we see a trend of diminishing returns with small additional improvements in WER as we increase the amount of SynPT beyond 100 hours.

[1] We also trained models with different ratios of combining HI and EN speech data. However, none of these models outperformed HI(50).

[2] `https://github.com/usnistgov/SCTK`

[3] We also created synthetic speech that preserves both SL and PT distributions (SynSL+PT). HI(350)+SynSL+PT(100) gives WERs of 57.05% (dev) and 58.69% (test), which is comparable to SynPT.

Table 2: *WERs using different TDNN-based acoustic models. (Double-lines demarcate three different sets of experiments.)*

Training	Dev	Test
HI(50)	63.01	65.14
HI(50)+SynConcat(100)	60.89	62.44
HI(50)+SynSL(100)	60.22	62.28
HI(50)+SynPT(100)	**59.05**	**60.81**
HI(350)	58.99	60.47
HI(350)+SynConcat(100)	57.91	59.65
HI(350)+SynSL(100)	**57.22**	58.89
HI(350)+SynPT(100)	57.31	**58.73**
HI(350)	58.99	60.47
HI(350)+SynPT(100)	57.31	58.73
HI(350)+SynPT(200)	56.62	58.73
HI(350)+SynPT(350)	**56.28**	**58.29**

4.2. Impact of synthetic text on language models

All the systems in Table 2 used an LM trained on CMtext. In this section, we fix the acoustic model to be the best-performing model from Table 2 i.e. HI(350)+SynPT(350), and investigate different trigram LMs[4] based on (i) monolingual HI and EN text and (ii) synthetic CM text. For monolingual text, we consider Hindi and English transcripts from the HI and EN speech corpora, corresponding to 3.2 million Hindi tokens and 1 million English tokens, respectively. For synthetic CM text, we experiment with transcripts from SynConcat, SynSL and SynPT. We also use a linguistically-motivated technique [17] to generate synthetic CM text starting from monolingual text, which is grammatically valid and constrained by a theory of code-mixing called the equivalence constraint theory (ECT). Using this technique, we generated a total of 24.5k mixed sentences starting from HI which will henceforth be referred to as ECT.

The WERs reported in Table 3 are computed after the reference and predicted sentences are subject to a transliteration post-processing step. This is because a significant number of errors arise from a mismatch in scripts where an English word in the reference transcript that is correctly predicted in the ASR hypothesis is marked as an error because it is rendered in Devanagari and vice-versa. Such a transliteration-based scoring mechanism was also explored in recent work [24]. We used the Bing Transliteration API[5] . This step improved our WERs by 2.97% on average.

Along with WERs, Table 3 also lists a second error metric specific to code-mixing that we call code-mixed WER (CM-WER). The sentence hypothesized by the ASR system is first aligned with the reference transcription to derive a sequence of edits i.e. substitutions, deletions and insertions. If there are M words on both sides of switch points across all reference transcriptions and N edits in the ASR hypotheses corresponding to words surrounding the switch points in the references, then CM-WER $= \frac{N}{M}$. This metric provides an estimate of how accurately the system predicts words at switch points.

The first two rows of Table 3 shows WERs obtained using a trigram LM trained only on Hindi transcripts and a trigram LM trained on both Hindi and English transcripts. We observe that simply adding English sentences to the LM significantly degrades performance. However, English sentences become useful in conjunction with synthetically generated CM text. This is apparent when we compare HI + ECT to HI + EN + ECT and obtain a statistically significant reduction in WER at $p < 0.001$. Adding ECT sentences improves predictions at switch points and the monolingual English sentences further improve prediction of the English segments within the CM utterances.

Table 3: *WERs using different LMs. Numbers in brackets denote CM-WER.*

Training Text	Dev	Test
HI	58.65 (63.81)	60.83 (65.50)
HI+EN	72.36 (77.66)	73.81 (80.62)
HI+ECT	58.39 (62.24)	60.86 (65.41)
HI+EN+ECT	57.72 (60.57)	60.12 (62.92)
HI+EN+SynConcat	57.75 (60.60)	60.15 (63.04)
HI+EN+SynSL	57.51 (60.03)	60.11 (62.75)
HI+EN+SynPT	57.49 (60.00)	60.12 (62.86)
SynSL+SynPT+ECT (S-All)	57.88 (60.05)	60.25 (62.79)
CMtext	55.10 (**52.79**)	57.38 (55.33)
CMtext+S-All	**54.59** (52.92)	**56.97** (**55.05**)

The next three rows of Table 3 show how WERs vary when we train LMs using transcripts from SynConcat(100), SynSL(100) and SynPT(100). Interestingly, LMs trained using this synthetic data (which is agnostic to linguistic constraints) are comparable in performance to the LMs trained using ECT (which is linguistically motivated). These numbers are further comparable to an LM trained exclusively on data from all synthetic sources (except SynConcat) and no monolingual text.

From the last two rows of Table 3, we see that the use of monolingual data and synthetic data sources brings us close to the WERs obtained by an LM trained only on CMtext. However, CM-WER using CMtext is substantially better when compared to any of the other LMs. Linearly interpolating an LM trained on CMtext (scaled by 0.9) with an LM trained on SynSL + SynPT + ECT (scaled by 0.1) provides further improvements over an LM trained on CMtext.

5. Conclusions and future work

In this work, we generate synthetic Hindi-English CM speech that obeys constraints of real CM text and significantly helps improve ASR performance of CM speech. Transcriptions from the synthetic corpus are also effective in improving the quality of language models for CM data. In future work, we will explore how to further improve the quality of synthetically generated speech with the help of text-to-speech systems.

6. Acknowledgements

We are grateful to Niranjan Nayak, Rupesh Mehta, Sandeepkumar Satpal and Ankur Gupta of Microsoft IDC for their support with hardware resources and corpora, and their valuable guidance. We would also like to thank Adithya Pratapa and Sandipan Dandapat for their help with generating ECT based CM text.

[4]Rescoring with recurrent neural network-based LMs did not significantly improve performance.

[5]https://azure.microsoft.com/en-in/services/cognitive-services/translator-text-api/

7. References

[1] Y. Li, P. Fung, P. Xu, and Y. Liu, "Asymmetric acoustic modeling of mixed language speech," in *Proceedings of ICASSP*, 2011, pp. 5004–5007.

[2] D. Imseng, H. Bourlard, M. M. Doss, and J. Dines, "Language dependent universal phoneme posterior estimation for mixed language speech recognition," in *Proceedings of ICASSP*, 2011, pp. 5012–5015.

[3] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz, and H. Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *Proceedings of ICASSP*, 2012, pp. 4889–4892.

[4] H. Adel, N. T. Vu, F. Kraus, T. Schlippe, H. Li, and T. Schultz, "Recurrent neural network language modeling for code switching conversational speech," in *Proceedings of ICASSP*, 2013, pp. 8411–8415.

[5] H. Adel, N. T. Vu, K. Kirchhoff, D. Telaar, and T. Schultz, "Syntactic and semantic features for code-switching factored language models," *Proceedings of IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 431–440, 2015.

[6] S. Garg, T. Parekh, and P. Jyothi, "Dual language models for code switched speech recognition," in *Proceedings of Interspeech*, 2018, pp. 2598–2602.

[7] C. Shan, C. Weng, G. Wang, D. Su, M. Luo, D. Yu, and L. Xie, "Investigating end-to-end speech recognition for Mandarin-English code-switching," in *Proceedings of ICASSP*, 2019, pp. 6056–6060.

[8] K. Li, J. Li, G. Ye, R. Zhao, and Y. Gong, "Towards code-switching ASR for end-to-end CTC models," in *Proceedings of ICASSP*, 2019, pp. 6076–6080.

[9] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. J. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *Proceedings of ICASSP*, 2018, pp. 4904–4908.

[10] S. Watanabe, T. Hori, and J. R. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *Proceedings of ASRU*, 2017, pp. 265–271.

[11] B. Li, Y. Zhang, T. N. Sainath, Y. Wu, and W. Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," *CoRR*, vol. abs/1811.09021, 2018.

[12] Z. Zeng, Y. Khassanov, V. T. Pham, H. Xu, E. S. Chng, and H. Li, "On the end-to-end solution to Mandarin-English code-switching speech recognition," *CoRR*, vol. abs/1811.00241, 2018.

[13] S. Garg, T. Parekh, and P. Jyothi, "Code-switched language models using dual RNNs and same-source pretraining," in *Prooceedings of EMNLP*, 2018, pp. 3078–3083.

[14] G. I. Winata, A. Madotto, C. Wu, and P. Fung, "Learn to code-switch: Data augmentation using copy mechanism on language modeling," *CoRR*, vol. abs/1810.10254, 2018.

[15] C. Chang, S. Chuang, and H. Lee, "Code-switching sentence generation by generative adversarial networks and its application to data augmentation," *CoRR*, vol. abs/1811.02356, 2018.

[16] H. Gonen and Y. Goldberg, "Language modeling for code-switching: Evaluation, integration of monolingual data, and discriminative training," *CoRR*, vol. abs/1810.11895, 2018.

[17] 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 ACL*, 2018, pp. 1543–1553.

[18] H. Seki, S. Watanabe, T. Hori, J. L. Roux, and J. R. Hershey, "An end-to-end language-tracking speech recognizer for mixed-language speech," in *Proceedings of ICASSP*, 2018, pp. 4919–4923.

[19] S. Nakayama, A. Tjandra, S. Sakti, and S. Nakamura, "Speech chain for semi-supervised learning of Japanese-English code-switching ASR and TTS," in *SLT Workshop*, 2018, pp. 182–189.

[20] G. A. Guzmán, J. Ricard, J. Serigos, B. E. Bullock, and A. J. Toribio, "Metrics for modeling code-switching across corpora," in *Proceedings of Interspeech*, 2017, pp. 67–71.

[21] D. Povey, A. Ghoshal *et al.*, "The Kaldi speech recognition toolkit," in *Proceedings of ASRU*, 2011.

[22] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of Interspeech*, 2015, pp. 3214–3218.

[23] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of ICSLP*, 2002, pp. 901–904.

[24] J. Emond, B. Ramabhadran, B. Roark, P. J. Moreno, and M. Ma, "Transliteration based approaches to improve code-switched speech recognition performance," in *SLT Workshop*, 2018, pp. 448–455.