KARI: KAnari/QCRI’s End-to-End systems for the \textsc{Interspeech} 2021 Indian Languages Code-Switching Challenge

Amir Hussein$^{1,2}$, Shammur Chowdhury$^2$, Ahmed Ali$^2$

$^1$Kanari AI, California, USA
$^2$Qatar Computing Research Institute, HBKU, Doha, Qatar

\texttt{amir@kanari.ai, \{schowdhury, amali\}@hbku.edu.qa}

\section*{Abstract}

In this paper, we present the Kanari/QCRI (KARI) system and the modeling strategies used to participate in the \textsc{Interspeech} 2021 Code-switching (CS) challenge for low resource Indian languages. The subtask involved developing a speech recognition system for two CS datasets: Hindi-English and Bengali-English, collected in a real-life scenario. To tackle the CS challenges, we use transfer learning for incorporating the publicly available monolingual Hindi, Bengali, and English speech data. In this work, we study the effectiveness of two steps transfer learning protocol for low resourced CS data: monolingual pretraining, followed by fine-tuning. For acoustic modeling, we develop an end-to-end convolution-augmented transformer (Conformer). We show that selecting the percentage of each monolingual data effects model biases towards using one language character set over the other in a CS scenario. The models pretrained on well-aligned and accurate monolingual data showed robustness against misalignment between the segments and the transcription. Finally, we develop word-level n-gram language models (LM) to rescore ASR recognition.

\textbf{Index Terms:} code-switching, conformer, end-to-end, speech recognition, transfer learning

\section{1. Introduction}

Code-switching (CS) is one of the most common phenomenon in a multilingual society, where speakers often alter between two or more languages. The CS occurring in spontaneous speech is highly unpredictable and difficult to model. Due to its presence in day-to-day conversation and also in more formal and semi-formal settings like educational lectures and news \cite{1}, this phenomenon has captured the attention of the speech recognition community.

As a result, there has been increasing interest in building automatic speech recognition (ASR) systems for CS including Mandarin-English \cite{2}, Hindi-English \cite{3}, and French-Arabic \cite{4}. Recently proposed approaches to model CS, shifted from using modular Hidden Markov Model-Deep Neural Networks (HMM-DNN) to end-to-end (E2E) ASR approaches as a result of the E2E system’s success in multilingual ASR \cite{5,6}. In \cite{7,8} studies proposed additional language identification task on top of connectionist temporal classification (CTC) Attention (CTC-Attention) \cite{9} architecture for English-Mandarin CS. In \cite{5} authors modeled limited Hindi-English CS using E2E attention model \cite{10} with context-dependent target to word transduction, factorized language model with part-of-speech (POS) tagging and code-switching identification. In \cite{11} authors proposed transformer-based architecture with two symmetric language-specific encoders to capture the individual language attributes for Mandarin-English CS.

In this paper we describe our E2E speech recognition systems designed to participate in the \textsc{Interspeech} 2021 CS task \cite{12}. The main objective of the CS \textsc{Interspeech} 2021 task (subtask2) is to build a robust ASR systems for Hindi-English and Bengali-English \cite{13} spoken content. In this context, we refer Hindi and Bengali as native languages of the speakers and the English as the non-native/second language. Both datasets are collected from spoken tutorials which covered various technical topics. The task is particularly challenging due to the scarcity of publicly available acoustic and lexical Hindi-English and Bengali-English CS resources. Furthermore, given the data was collected from a real-life scenario, the transcripts are not specifically created for ASR use purposes, but for end-user consumption. This introduced different kinds of noise:

1. Misalignments between the transcription and segment start and end times.
2. Inconsistency in the script used to write the same word (some English words were written in the Latin script and the native scripts of Hindi and Bengali).
3. Some English words are merged with the native Hindi/Bengali words as one word.
4. In some cases the transcription of the spoken utterance was found inaccurate or completely wrong.

In this study, we investigate the effectiveness of transfer learning from monolingual languages for low resourced CS data. We show that it is possible to achieve significant improvements in CS task with two steps: 1) balanced pre-training on monolingual languages, and 2) careful fine-tuning for the CS task. In addition, we provide a detailed practical guidelines for the effective E2E conformer pre-training and fine-tuning strategy with limited CS data. In our approach, the acoustic model was built based on the recently introduced end-to-end (E2E) convolution-augmented transformer (conformer) for speech recognition \cite{14}. In \cite{14} authors showed that the conformer outperforms significantly the traditional transformer in most of the ASR tasks. For language modeling (LM) we used word level bi-gram model as it provided best results on.

\begin{itemize}
\item [1] \url{https://navana-tech.github.io/IS21SS-indicASRchallenge/}
\end{itemize}
limited CS data compared to other deep learning methods. To train the LM models, we used publicly available monolingual data in addition to the CS data.

The rest of this paper is organized as follows: Section 2 presents the development of the acoustic model. Section 3 describes the language modeling. Section 4 presents datasets description. The details of the experimental setup, results and their discussion are given in Section 5. Section 6 concludes the findings of our study.

2. Acoustic Modeling

We develop ASR conformer architecture using ESPNET toolkit [14]. The implementation consists of a conformer encoder [13] which is a multi-blocked architecture and a transformer decoder. The encoder consists of several blocks each is a stack of a position-wise feed-forward (FF) module, multi-head self-attention (MHSA) a convolution operation (CONV) module, and another FF module in the end. The self-attention computation of every single head in MHSA can be formulated as:

\[
Atth(q_h, k_h, V_h) = S\left(\frac{Q_h \times K_h^T}{\sqrt{d_k}}\right) \times V_h
\]

where \( S \) is a softmax operation, \( Q = X \times W^q \), \( K = X \times W^k \) and \( V = X \times W^v \) are the queries, keys and values respectively. The \( W^q \) and \( W^k \) are learnable weights. The \( d_{out} \) is the dimension of the attention, and \( d_k \), \( d_v \) are the dimensions of values, keys and queries. To simultaneously attend to information from different representations, outputs of each head are concatenated in MHSA as follows:

\[
\text{MHSA}(Q, K, V) = [\text{head}_1, \ldots, \text{head}_h] W^h \quad (2)
\]

where \( h \) is the number of attention heads in a layer. The MHSA is followed by a convolution module (CONV) which consists of a 1-D convolution layer, gated linear units (GLU) activation batch normalization (BN) layer and a Swish activation as shown in Figure 1.

![Illustration of the conformer CONV module. All 1D-CNN operations are point-wise convolution.](image)

Each module includes layer normalization (LN) and is followed by a layer dropout (D), and a residual connection from the module input as shown in Equation 3.

\[
\begin{align*}
X' &= \frac{1}{2} D(\text{FF}_1(\text{LN}(X))) + X \\
X'' &= D(\text{MHSA}(\text{LN}(X'))) + X' \\
Z &= D(\text{CONV}(\text{LN}(X''))) + X'' \\
Z' &= \frac{1}{2} D(\text{FF}_2(\text{LN}(Z))) + Z
\end{align*}
\]

2.1. Conformer ASR training

During the training, the conformer ASR predicts the target sequence \( Y \) of tokens from acoustic features \( X \). For text tokenization, we used word-piece byte-pair-encoding (BPE) [15]. The total loss function \( L_{asr} \) is a multi-task learning objective that combines the decoder cross-entropy (CE) loss \( L_{ce} \) and the CTC loss [16] \( L_{ctc} \).

\[
L_{asr} = \alpha L_{ctc} + (1 - \alpha) L_{ce} \quad (4)
\]

where \( \alpha \) is a weighting factor with the selected best value of 0.3. In our approach, the conformer is first pretrained with monolingual speech data from both Hindi/Bangali and English with shared vocabulary for both languages. We add around half of the available CS data to make the model familiar with CS examples that mix the two languages. Then, we fine-tune all the model parameters on all the available CS speech with a very small learning rate (\( \frac{1}{100} \) of lr used during the pretraining).

3. Language Modeling

Since the available CS text data is very limited we decided to train word-level n-gram language models (LMs); a 2-gram and a 3-gram LMs. Both n-gram models were trained with the KenLM toolkit [17] on the entire text data described in Section 4. During the decoding, the best transcription is selected by leveraging both the posteriors of an acoustic model (AM) and the perplexity of a language model (LM).

4. Datasets Description

In this section, we describe the details of the provided Interspeech 2021 data for Code-switching subtask. In addition, we also present all the publicly available acoustic and text resources that were used in developing our approach.

4.1. Speech data

The code-switching challenge used Hindi-English and Bengali-English datasets recorded from spoken tutorials covering various technical topics. Both datasets were sampled at 16 kHz with 16 bits encoding. Basic analysis showed that each dataset contains around 45% of non-native words and 55% of Hindi/Bengali native words. In addition to the provided CS speech datasets, we used publicly available Bengali (Bn) [18] dataset, Hindi (Hi) speech from the Interspeech 2021 multilingual challenge, and Tedlium3 [19]. All the speech data was sampled at 16kHz except Hi which was sampled at 8kHz. As a result, we upsampled the Hi audio to 16kHz. Since each Hindi and Bengali monolingual datasets are limited we use different subsets of Tedlium3 ranging from 22.7 hours to 203 hours. More details about the datasets are shown in Table 1.

---

1. https://speech-improvement.dataiku.com/indicASRchallenge/data.html
4. Text data

For CS language modeling we used Hindi-English news paper dataset\(^2\) and Bengali-English wiki dataset\(^2\). Moreover, we used the challenge CS transcription text, Tedlium3 transcription text, and Hindi Wikipedia articles\(^3\).

5. Experiments and Results

5.1. Experimental Setup

We ran our experiments on an HPC node equipped with 4 NVIDIA Tesla V100 GPUs with 16 GB memory, and 20 cores of Xeon(R) E5-2690 CPU.

We first augment the raw speech data with the speed perturbation with speed factors of 0.9, 1.0, and 1.1 [20]. Then, we extract 83-dimensional feature frames consisting of 80-dimensional log Mel-spectrogram and pitch features [21] and apply cepstral mean and variance normalization (CMVN). Furthermore, we augment these features using the specaugment approach [22]. To reduce the noise in the provided CS transcription we performed basic cleaning by removing all punctuation except the symbols that were spoken in the audio {_, /, =, +, %, \*}. In addition, we converted English words to lowercase and separated the numbers and different words that were glued in one word: (attributes [20]).

5.2. Default model hyperparameters

All hyperparameters were obtained using a grid search during the pretraining phase. The E2E conformer-based ASR model was trained using Noam optimizer [23]. Table (2) summarizes the best set of parameters that were found for the conformer architecture. Both models were pretrained for 60 epochs with dropout-rate 0.1, warmup-steps of 20000, and learning rate of 5.

5.3. Pretraining

During the pretraining, the number of selected hours of non-native Tedlium3 was limited by the number of available hours of each native Hindi and Bengali data to avoid data biases. We used two configurations: 1) Equal non-native (Ev) where the percentage of the non-native data is 45% and the native data is 55% (similar to the percentage of each language in the CS data) 2) Small non-native (Sv): the ratio of non-native to the native data is 1 : 4. In addition, we added around half of the provided CS data during the pretraining. The number of selected hours for each configuration is summarized in Table 3.

In Figure 2, we show an example of decoded outputs produced from the Ev and Sv configurations.

| Table 2: Values of E2E conformer hyperparameters obtained from the grid search. |
|-----------------------------------------------|
| **Hyperparameters** | **Values** |
| BPE | 1000 |
| Batch size | 64 |
| Attention heads | 4 |
| CNN module kernel | 15 |
| Encoder layers | 8 |
| Decoder layers | 4 |
| \(d\) | 512 |
| FF units | 2048 |

Table 3: Number of hours of the monolingual (Hindi/Bengali), Tedlium3, and the CS used in pretraining phase for Ev and Sv configurations.

| Configuration | Native | Tedlium3 | CS |
|---------------|--------|----------|----|
| Sv (Hindi)    | 95     | 22.7     | 50 |
| Ev (Hindi)    | 95     | 86       | 50 |
| Sv (Bengali)  | 211.6  | 57.6     | 20.5 |
| Ev (Bengali)  | 211.6  | 200.5    | 20.5 |

Figure 2: Examples from Hindi-English CS-Dev set decoded by E2E conformer model pretrained with Ev and Sv configurations.

It can be noted from Example 1 that the transcription from pretraining with Ev configuration is better in identifying English words than the Sv configuration which is more biased to Hindi characters. This is expected as with the Sv configuration the Hindi speech is around 4 times that of English speech. We note here that both transcriptions are phonetically correct. In addition, pretraining with Ev configuration resulted in a more robust

\(^2\)https://www.kaggle.com/pk13055/code-mixed-hindienglish-dataset
\(^3\)https://www.kaggle.com/abyasrafi/bwnwiki
\(^5\)https://www.kaggle.com/disishbig/hindi-wikipedia-articles-172k
model to misalignments since the size of the well-aligned pretraining data is larger than the Sv configuration. On the other hand, from Example 2 we can see that pretraining with Sv produces more accurate predictions in the Hindi language. The provided evaluation set in the challenge was very noisy in a level that increased the WER when the model produced better quality transcription that was captured with a manual investigation. To select the best-pretrained model we created our own development set consisting of 2.6 hours of each Hindi and Bengali monolingual sets and 2.6 hours of CS evaluation set. The results of the pretraining models for both Ev and Sv configurations are summarized in Table 4.

Table 4: WER results on the locally created development set (l-Dev) from 2.6 hours of native Hindi (Hi) and Bengali (Bn), non-native Tedlium3 (Ted3), and the code-switching development set (CS-Dev).

| Configuration         | Native | Ted3 | CS-Dev | l-Dev |
|-----------------------|--------|------|--------|-------|
| Baseline (Hi-En) [12] | -      | -    | 37.7   | -     |
| Baseline (Bn-En) [12] | -      | -    | 37.2   | -     |
| Sv (Hi-En)            | 33.1   | 19.6 | 28.3   | 26.9  |
| Ev (Hi-En)            | 35.1   | 12.7 | 27.5   | 25.6  |
| Ev (Hi-En)+2gram      | 34.2   | 13.3 | 28.3   | 23.1  |
| Sv (Bn-En)            | 15.2   | 17.7 | 28.2   | 20.2  |
| Ev (Bn-En)            | 20.4   | 16.8 | 27.7   | 22.8  |
| Ev (Bn-En)+2gram      | 18.4   | 16.1 | 28.2   | 19.9  |

It can be seen that the pretrained models with only half of the provided CS data, significantly outperformed the challenge baselines on average by 15% in relative WER on CS development set. It is worth noting that for the Hi-En baseline the development set is very noisy, and we found that the quality of the pretrained model is significantly better and more accurate than the noisy reference. As a result we think that pretraining the model on a well aligned and accurate script from monolingual data resulted in robustness against inaccurate segment alignments and incorrect reference transcription presented in the CS training data. Finally, re-scoring with 2gram LM model corrects some spellings and helps better selecting the characters set for the corresponding language as shown in Figure 3. However, we noticed that in some examples the LM re-scoring introduced some deletions.

**Example 3**

| Ref. | अब tagged अभिधारण पर क्लिक करें अभिधारण करें |
|------|-------------------------------------------|
| Translated | Now click on the tagged icon again |
| Hyp. Adapted | अब tagged icon पर क्लिक करें |
| Hyp. Adapted+2gram | अब tagged icon पर क्लिक करें |

**Figure 3:** Examples from Hindi-English CS-Dev set decoded by models pretrained with Equal non-native (Ev) configuration.

5.4. Model fine-tuning

The best results were obtained from fine-tuning the entire network with a very small learning rate of 0.1. Since the provided test set was very noisy, we decided to report model fine-tuning results obtained from the final blind submissions. In the challenge, the systems were evaluated using the conventional word error rate (WER) and the transliterated WER (T-WER) as shown in Table 5. The T-WER counts an English word in the reference text as being correctly predicted if it is in English or in a transliterated form in the native script.

Table 5: WER% & Transliterated WER (T-WER)% results on Hi-En and Bi-En final blind set.

|                  | Hi-En | Bi-En | AVG WER | AVG T-WER |
|------------------|-------|-------|---------|-----------|
| Baseline [12]    | 25.5  | 32.8  | 31.7    | 29.2      |
| Sv (adapted)     | 21.9  | 26.3  | 25.3    | 23.7      |
| Ev (adapted)     | 20.3  | 25.8  | 24.5    | 23.8      |
| Ev (adapted)     | 22.4  |       |         |           |

It can be seen that the EV finetuned (adapted) configuration resulted in best results which confirms our findings from the pretraining phase. The rescoring with LM model corrected some mistakes however it also introduced some deletions. Due to limited number of submissions we did not consider the system with LM for final submission.

5.5. Practical considerations for E2E conformer transfer learning from monolingual to CS

Monolingual conformer pretraining for CS: The monolingual pretraining for CS is very sensitive and can easily be biased to one language character set due to the phonetic overlap between the two languages. Hence for successful pretraining for CS task, we recommend choosing the percentage of each monolingual data close to their expected percentage in the CS data.

Language modeling for ASR rescoring: our results suggest that 2-gram model provided the best re-scoring for E2E conformer ASR in the CS scenario compared to other deep learning techniques. The rescoring with LM corrects words spelling and helps choosing the correct language character set, however, sometimes it introduces deletions.

Conformer finetuning: Fine-tuning pretrained conformer by following the conventional freezing approach degraded the performance. In fact, our results suggest that freezing any blocks in the encoder, decoder or both (encoder+decoder) resulted in worse performance. The best results were obtained from fine-tuning the entire E2E network with very small learning rate of 0.1 and no warmup steps.

6. Conclusions

In this paper, we have presented and evaluated our transfer learning approach for the E2E conformer-based ASR system (KARI), designed to participate in the Interspeech 2021 Code-switching challenge. The two steps transfer learning showed significant improvements and robustness against segment misalignment and script inconsistencies. In addition, we showed the effect of the percentage of each selected monolingual data for pretraining, on the CS ASR performance. Finally, we provided practical guidelines that will provide guidance to the practitioners and researchers in developing ASR systems for CS in a real-world scenario. In future work, we
plan to explore the applicability of other transfer learning methods for code-switching that includes self-supervised and multi-task learning approaches.

7. References

[1] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, “A survey of code-switched speech and language processing,” arXiv preprint arXiv:1904.00754, 2019.

[2] Y. Li and P. Fung, “Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints,” in ICASSP, 2013.

[3] G. Seeram and R. Sinha, “Exploration of end-to-end framework for code-switching speech recognition task: Challenges and enhancements,” IEEE Access, vol. 8, pp. 68146–68157, 2020.

[4] D. Amazouz, M. Adda-Deckor, and L. Lamel, “Addressing code-switching in French/Algerian Arabic speech,” in Interspeech, 2017.

[5] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, “Multilingual speech recognition with a single end-to-end model,” in 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2018, pp. 4904–4908.

[6] A. Datta, B. Ramabhadran, J. Emond, A. Kannan, and B. Roark, “Language-agnostic multilingual modeling,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 8239–8243.

[7] N. Luo, D. Jiang, S. Zhao, C. Gong, W. Zou, and X. Li, “Towards end-to-end code-switching speech recognition,” arXiv preprint arXiv:1810.13091, 2018.

[8] 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 ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 6056–6060.

[9] S. Kim, T. Hori, and S. Watanabe, “Joint ctc-attention based end-to-end speech recognition using multi-task learning,” in 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2017, pp. 4835–4839.

[10] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016, pp. 4960–4964.

[11] X. Zhou, E. Yilmaz, Y. Long, Y. Li, and H. Li, “Multi-encoder-decoder transformer for code-switching speech recognition,” arXiv preprint arXiv:2006.10414, 2020.

[12] A. Diwan, R. Vaideeswaran, S. Shah, A. Singh, S. Raghavan, S. Khare, V. Unni, S. Vyas, A. Rajpuria, C. Yarra, A. Mittal, P. K. Gosh, P. Jyothi, K. Bali, V. Sheshadri, S. Sitaram, S. Bharadwaj, J. Nanavati, R. Nanavati, K. Sankaranarayanan, T. Seeram, and B. Abraham, “Multilingual and code-switching asr challenges for low resource indian languages,” arXiv preprint arXiv:2104.00925, 2021.

[13] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al., “Conformer: Convolution-augmented transformer for speech recognition,” arXiv preprint arXiv:2005.08100, 2020.

[14] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi, et al., “Recent developments on espnet toolkit boosted by conformer,” arXiv preprint arXiv:2010.13956, 2020.

[15] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” arXiv preprint arXiv:1808.06226, 2018.

[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proceedings of the 23rd international conference on Machine learning, 2006.

[17] K. Heafied, “Kenlu: Faster and smaller language model queries,” in Proceedings of the sixth workshop on statistical machine translation, 2011, pp. 187–197.

[18] O. Kjartansson, S. Sarin, K. Pipatsrisawat, M. Jansche, and L. Ha, “Crowd-Sourced Speech Corpus for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali,” in Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU), Gurugram, India, Aug. 2018, pp. 52–55. [Online]. Available: http://dx.doi.org/10.21437/SLTU.2018-11.

[19] F. Hernandez, V. Nguyen, S. Ghamnay, N. Tomashenko, and Y. Estève, “Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation,” in International Conference on Speech and Computer (ICSC), 2018, pp. 198–208.

[20] T. Ko, V. Poddinti, D. Povey, and S. Khudanpur, “Audio augmentation for speech recognition,” in Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[21] P. Ghalshremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, “A pitch extraction algorithm tuned for automatic speech recognition,” in IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2014, pp. 2494–2498.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” arXiv preprint arXiv:1904.08779, 2019.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in neural information processing systems, 2017, pp. 5998–6008.