INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

The CSTR System for Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages

Ondřej Klejch, Electra Wallington, Peter Bell

Centre for Speech Technology Research, University of Edinburgh, United Kingdom
{o.klejch, electra.wallington, peter.bell}@ed.ac.uk

Abstract

This paper describes the CSTR submission to the Multilingual and Code-Switching ASR Challenges at Interspeech 2021. For the multilingual track of the challenge, we trained a multilingual CNN-TDNN acoustic model for Gujarati, Hindi, Marathi, Odia, Tamil and Telugu and subsequently fine-tuned the model on monolingual training data. A language model built on a mixture of training and CommonCrawl data was used for decoding. We also demonstrate that crawled data from YouTube can be successfully used to improve the performance of the acoustic model with semi-supervised training. These models together with confidence based language identification achieve the average WER of 18.1%, a 41% relative improvement compared to the provided multilingual baseline model. For the code-switching track of the challenge we again train a multilingual model on Bengali and Hindi technical lectures and we employ a language model trained on CommonCrawl Bengali and Hindi data mixed with in-domain English data, using a novel transliteration method to generate pronunciations for the English terms. The final model improves by 18% and 34% relative compared to our multilingual baseline. Both our systems were among the top-ranked entries to the challenge.

Index Terms: low-resource speech recognition, multilingual speech recognition, code switching

1. Introduction

Increasing the amount of transcribed training data can lead to large improvements in the accuracy of automatic speech recognition (ASR) systems for mainstream languages. However, these large amounts of transcribed training data are not available for the vast majority of world languages given the high cost of transcription. Therefore, there has been a growing interest in developing methods that would allow training of ASR models for low-resource languages with limited transcribed training data.

Multilingual training or cross-lingual transfer have been shown to improve the performance of ASR systems in low-resource languages. For example, transcribed data from well-resource languages can be used to train a multilingual bottleneck feature extractor which can be subsequently used for low-resource languages [1, 2, 3]. Alternatively, a model trained on a well-resourced language can be used in a low-resource setting by replacing its output layer with a new output layer for the target low-resource language [4, 5, 6]. Recently, it has been shown that it is also possible to leverage untranscribed data in the target low-resource language with semi-supervised and self-supervised training [7, 8, 9]. Apart from the scarcity of transcribed data, training ASR systems for low-resource languages is also challenging because speakers of those languages might use words from multiple languages within a single utterance – a phenomenon called code-switching. As an example, speakers of minority languages may use English terms when speaking in their native language. Therefore, there has been a lot of research trying to address this issue by focusing on acoustic modelling [10, 11], language modelling [12, 13], decoding [14] and data augmentation [15].

Over the past few years we have worked on the IARPA Material program, during which we have developed a pipeline for training ASR systems for low-resource languages with limited amounts of transcribed training data. In order to improve the model which is typically trained on 50–100 hours of transcribed training data, we crawl text and audio data from the internet and we employ semi-supervised training to train on this crawled data, which leads to large improvement gains [8]. As a result we are able to train an ASR system for a new language without any knowledge of the target language using only a small amount of transcribed data, thus lowering the barrier for training of ASR systems in new languages. Moreover, given enough computation capacity, we are able to train a reasonable baseline system using the crawled data in a matter of days. In this paper we describe how we used this pipeline to train ASR systems for seven Indian languages in the Multilingual and Code-Switching (MUCS) ASR Challenges at Interspeech 2021 [16].

The main contribution of this paper is our demonstration that it is possible to train a solid ASR system for low-resource languages using small amounts of transcribed training data and data crawled from the internet without any knowledge of the target language: in this case any knowledge of the Indian languages.

2. Multilingual ASR

In Track 1 of the challenge participants were required to build a multilingual model for Gujarati, Hindi, Marathi, Odia, Tamil and Telugu. Furthermore, the participants had to build a language identification system because the blind evaluation data did not contain language information. Here we describe how we approached the problem by training a multilingual model (see Figure 1) with subsequent monolingual fine-tuning and ASR confidence based language identification.

2.1. Acoustic Model Training

Our acoustic model training starts with standard GMM training which produces speaker adaptive GMM models. These are used for alignment of the training data, which is then used for training of a neural network based acoustic model with lattice-free maximum mutual information [17]. We pre-train the acoustic model in a multilingual fashion by pooling training data for all languages and having language specific output layers (see Figure 1). After the multilingual pre-training we split the model into six language-dependent acoustic models and we fine-tune them on corresponding training data. As we discussed earlier,

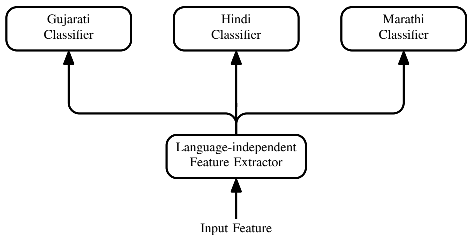

Figure 1: *Example of multilingual acoustic model for Gujarati, Hindi and Marathi with language-independent feature-extractor and language-dependent classifiers. In our experiments we train a multilingual model for all six languages.*

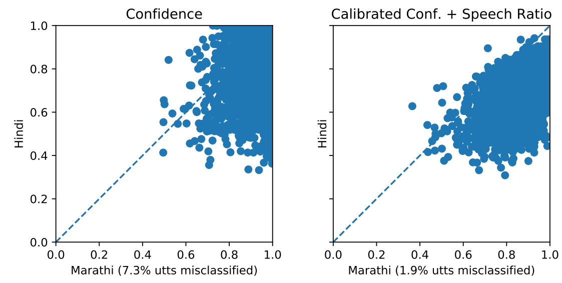

Figure 2: *Language identification of Marathi utterances with Hindi ASR and Marathi ASR with confidence and calibrated confidence + speech ration used for scoring.*

our pipeline uses data crawled from YouTube to improve the performance of ASR systems in low-resource languages with semi-supervised training [18, 8]. Therefore in this challenge we used data crawled from YouTube for semi-supervised multilingual pre-training and the provided transcribed data for subsequent monolingual fine-tuning.

We used a CNN-TDNN architecture which consists of 6 convolutional layers and 12 factored time delayed neural network layers [19] followed by language dependent output layers. The model is trained with lattice-free maximum mutual information (LF-MMI) [17]. The model uses 40-dimensional MFCC features together with i-vectors for speaker-adaptation [20, 21]. The MFCC features are extracted from recordings downsampled to 8kHz and the i-vector extractor is trained on data pooled from all languages. The model is trained with natural gradient [22] and uses SpecAugment [23] and Dropout [24] for regularization.

2.2. Language Model Training

We train a separate language model for each language. Since the provided training data is very limited – especially for Hindi, Marathi and Odia, which contain only 4506, 2543 and 820 unique sentences respectively – we use CommonCrawl data[1] for each language. We preprocess the CommonCrawl data by removing all non-alphanumeric tokens and mapping tokens that contain any character not present in the provided language-specific lexicon to the `<unk>` token. Subsequently we train trigram languages models with Kneser-Ney smoothing [25] on training and CommonCrawl data with the SRILM toolkit [26], we interpolate them with interpolation weights optimized on the provided test data, we select the top 300 thousand words, and we prune the language model using the relative entropy criterion [27] with threshold 10^{-9}. We used Sequitur [28] to produce pronunciations for words that were not present in the provided lexicon.

We also trained recurrent neural network language models (RNN LM) for rescoring of the lattices obtained with the first pass decoding [29]. The RNN LMs were trained with Kaldi on the same data as the n-gram language models. The weights of different data sources were set to the values used for the n-gram language model interpolation as they should reflect how well a given resource matches test conditions.

2.3. Crawling YouTube Data

One of the challenges when working with low-resource languages is the scarcity of transcribed data. In the Material program we usually crawl YouTube videos to obtain data that can be used for semi-supervised training. This process can be divided into two steps: video crawling and filtering. In the first step we search for videos using the most frequent trigrams in the language model as search queries. Once we crawl sufficient amounts of videos (we typically crawl 1000 hours of raw Youtube videos) we need to filter the data. This is because the crawled data may contain recordings from a different language, or else they can be music videos which could negatively affect the performance of the semi-supervised training. We filter the videos in two steps: first we use text-based language identifier CLD 2[2] to remove videos whose description is not written in the correct language. Then we segment the videos with WebRTC VAD[3] and use a seed ASR model to decode the filtered videos. We then compute the mean confidence and speaking rate for each recording and we keep videos with a mean confidence higher than 0.7 and speaking rate above 1.25. Note that in our previous experiments we found that speaking rate is particularly helpful for removing music videos from the crawled data. After the filtering we randomly selected 200 hours of training data for each language for multilingual pre-training.

2.4. Language Identification

The recordings in the blind test data for the multilingual challenge are supplied without a language label. Therefore, in order to use our language-specific ASR models, it was necessary to perform language identification. We used a simple method based on ASR confidence scores: we decoded all recordings with all six language-specific models and then, for each recording, we picked the output with the highest mean word confidence. To improve the language identification performance we calibrated model confidence scores on the provided test set. Furthermore, during data analysis we noticed that speech ratio, computed as the duration of decoded words divided by the total recording duration, helped with language identification. Therefore we computed the final language identification score as a mean of calibrated confidence and speech ratio. The impact of including speech ratio for language identification is illustrated in Figure 2 and the impact on the final WER is shown in Table 2.

We note that this method for language identification is very inefficient because it requires every recording to be decoded with every model, and therefore would not scale to more languages. In the future we aim to replace it with an x-vector based language identification classifier [30] trained on the VoxLingua107 corpus [31].

[1]`http://data.statmt.org/ngrams/raw/`

[2]`https://github.com/CLD2Owners/cld2`

[3]`https://github.com/wiseman/py-webrtcvad`

Table 1: *WER of the multilingual-models evaluated on the test data with oracle language identification.*

	Gujarati	Hindi	Marathi	Odia	Tamil	Telugu	Average
seed multilingual model	16.3	26.7	16.5	33.0	30.1	27.3	24.9
+ monolingual fine-tuning	15.8	25.0	16.6	32.9	29.3	26.1	24.2
+ CommonCrawl LM	15.9	17.2	11.0	28.1	22.6	21.3	19.3
+ RNN LM rescoring	15.4	17.4	**9.9**	26.3	**21.7**	20.1	18.4
semi-supervised multilingual model	21.4	34.3	19.9	33.2	30.8	31.5	28.4
+ monolingual fine-tuning	15.7	24.7	16.8	32.1	29.3	26.3	24.1
+ CommonCrawl LM	14.7	15.1	10.9	26.6	22.7	21.1	18.4
+ RNN LM rescoring	**14.2**	**15.0**	10.1	25.6	22.0	20.0	**17.7**
train + semi-supervised multilingual model	16.8	26.1	16.6	32.1	29.8	26.8	24.6
+ monolingual fine-tuning	15.6	24.4	16.6	32.3	29.2	26.0	23.9
+ CommonCrawl LM	15.2	16.0	10.7	26.9	22.5	20.7	18.6
+ RNN LM rescoring	14.7	16.2	10.0	**25.4**	21.9	**19.7**	17.9

Table 2: *WER of the semi-supervised model with monolingual fine-tuning, CommonCrawl LM and RNN LM rescoring with different language identification methods.*

	Gujarati	Hindi	Marathi	Odia	Tamil	Telugu	Average
baseline	19.3	40.4	22.4	39.1	33.4	30.6	30.7
oracle	14.2	15.0	10.1	25.6	22.0	20.0	17.7
confidence	15.9	17.4	16.1	29.6	23.1	21.6	20.5
confidence + speech ratio	16.7	15.7	12.1	26.8	22.9	22.3	19.3
calibrated confidence	**14.6**	15.2	11.6	27.2	**22.4**	**20.2**	18.5
calibrated confidence + speech ratio	14.7	**15.0**	**10.6**	**25.9**	**22.4**	20.3	**18.1**

2.5. Results

As described in Section 2.1 we start by training a seed multilingual model which is then monolingually fine-tuned. This seed model is subsequently used to decode the crawled YouTube data, which is then used for semi-supervised multilingual training. As we can see in Table 1, monolingual fine-tuning of the multilingual models improves the performance in almost all cases. Furthermore, using the CommonCrawl LM together with RNN LM rescoring yields additional gains. Most interestingly we can see that using the YouTube data improves the average performance by 0.7% compared to using only the provided training data. Based on our past experience, we hoped that incorporating the YouTube data would result in bigger improvements but it is possible that the crawled data is too mismatched from the test data. This is apparent from the results of the semi-supervised model with and without monolingual fine-tuning.

Having found the best performing model we decided to evaluate it without language information. In Table 2 we summarize the results. We see that the baseline system provided by the organizers achieves the WER of 30.7% on average while our best system with the oracle language information achieves 17.7% on average. When using uncalibrated confidence, the language identity of 5.8% of utterances is misclassified and the average WER degrades to 20.5%. By calibrating the confidence on the test data and incorporating the speech ratio information, the language identity of only 0.9% of utterances is misclassified and the model achieves the average WER of 18.1% which is only 0.4% absolute worse than the oracle system and 12.6% absolute better than the provided baseline.

3. Code-Switching ASR

In Track 2 participants had to build an ASR system for transcription of lectures in Bengali and Hindi with a lot of code-switching, especially technical and programming terms. Here we describe how we trained our models for this track.

3.1. Acoustic Model

We trained the acoustic model in the same way as in Track 1 by following the Kaldi Babel recipe for GMM training and then training a multilingual CNN-TDNN model with 40 dimensional MFCC features and i-vectors as inputs. The main difference is that we found this Track's training data to be noisy. Therefore we cleaned the training data with a Kaldi cleanup script [32] that re-segments the training data and removes noisy parts which can have a detrimental effect on the AM training (as we will see in Table 3). We experimented with monolingual fine-tuning as in Track 1 but we did not observe any improvements. Furthermore, we also tried single stage transfer learning [33] using the semi-supervised multilingual model trained in Track 1 but we did not see any improvements compared to the multilingual model trained from scratch on the clean training data.

3.2. Lexicon

The provided lexicon for the code-switching track used the CMU pronunciation dictionary [34] for English terms and a grapheme lexicon for Bengali/Hindi terms. We hypothesized that this might be sub-optimal for discriminative training with LF-MMI because one sound could be represented by an English phoneme and Bengali/Hindi grapheme. Therefore we decided to map all words into a shared phoneme dictionary. We used the Hindi dictionary provided in Track 1 and the Bengali phonemic dictionary from the IARPA Babel program [35]. After that we had to map English phonemes to their Bengali and Hindi counterparts. Even though Flite [36] can produce pronunciations of English terms using an Indic phoneme set [37], we decided not to use it and instead implemented a tool that would automatically map phonemes between two languages with only monolingual lexicons.

We took inspiration from work on machine transliteration [38, 39] and we crawled Wikipedia for data that would allow us to learn the phoneme mapping. Specifically, we crawled pages about famous people listed on the Bengali and Hindi ver-

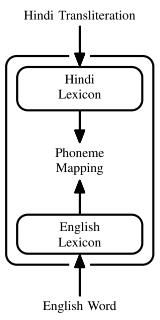

Figure 3: *Mapping English Phonemes into Hindi/Bengali counterparts*

sions of Wikipedia and then looked for the English version of the equivalent page. We took the title of these pages and, if the number of words in both titles matched, we took them as training examples. We hypothesised that if the number of words in both titles matches, the Hindi/Bengali titles are transliterations of English titles. Subsequently, we converted these words into their English and Indic phonetic representations and we trained a simple alignment model with the Baum-Welch algorithm [40]. The alignment model, implemented as an OpenFST transducer [41], allows substitutions between any pair of phonemes, insertions, and deletions, with the exception that insertion cannot be followed by deletion and vice versa. We then used the trained alignment model to extract one-best alignments between the training pairs, we removed pairs with low scores (as the Hindi/Bengali titles probably were not transliterations of the English titles), and we trained a pair language model with OpenGRM [42] which we used for phoneme mapping. This approach for phoneme-to-phoneme mapping is similar to the pair language model approach used for grapheme-to-phoneme conversion [43, 44].

3.3. Language Model Training

The main challenge of building language models for code-switched data is the lack of representative text data. As in Track 1 we used CommonCrawl data to enhance the language model training data. However, the CommonCrawl data is preprocessed with CLD 2 and thus sentences with a lot of foreign terms might be removed. In order to introduce relevant English terms into our training data we downloaded subtitles of English videos from `SpokenTutorial.org`. Since the data in this challenge track is from the Hindi and Bengali language mutations of this channel, this method ensured that we had relevant English terms in our language model. As in Track 1 we trained trigram LMs for first pass decoding and RNN LMs for lattice rescoring.

3.4. Results

The results of the models trained for the code-switching track of the challenge are summarized in Table 3. In the table we compare two sets of models, the first those those using the provided lexicon, which contains grapheme pronunciations for Bengali/Hindi words and phoneme pronunciations for English words, and the second those using the phoneme lexicon obtained by mapping English phonemes into their Bengali/Hindi counterparts. For both sets of models we first trained a multilingual model using both Bengali and Hindi training data. In the next step we trained models on cleaned training data, which led to substantial improvements for both languages (more than 3% and 6% absolute for Bengali and Hindi respectively). Next we resegmented the test data with WebRTC VAD which reduced the WER by another 0.8% and 1.8% respectively. Subsequently, we decoded the test data with the Web LM and rescored the lattices with the RNN LM. For Bengali, the best model did not use RNN LM rescoring as it degraded the final performance. By performing all these steps we managed to reduce the WER of the phoneme based model from 28.5% down to 23.3% for Bengali and from 27.2% down to 18.1% for Hindi. We also tried to perform single stage transfer learning [33] from the multilingual model trained in Track 1, but we did not get any gains on top of the results already reported.

Table 3: *WER of the code-switching models evaluated on the test data with document level scoring.*

	Bengali	**Hindi**
grapheme multilingual model	29.6	27.7
+ training data cleanup	26.5	21.4
+ test data resegmentation	25.6	19.9
+ Web LM	23.4	18.9
+ RNN LM rescoring	24.6	18.4
phoneme multilingual model	28.5	27.2
+ training data cleanup	26.4	21.3
+ test data resegmentation	25.6	19.5
+ Web LM	**23.3**	18.8
+ RNN LM rescoring	24.7	**18.1**

4. Conclusions

In this paper we described our submission to the MUCS 2021 challenge. Our submitted models were trained with our training pipeline developed for low-resource languages during the IARPA Material program. First the models were multilingually pre-trained on automatically crawled data from Youtube and then they were monolingually fine-tuned on the provided training data. Our results in this challenge suggest that it is possible to train good ASR system for low-resource languages with limited amounts of data and no knowledge of the target language. Furthermore, we demonstrated that it is possible to leverage data which is available on-line, for enhancing the acoustic model with semi-supervised training and extending the vocabulary of the language model. In the future, we will focus on improving semi-supervised training and transfer learning so that we can build ASR models for languages with very little or no transcribed data.

5. Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Air Force Research Laboratory (AFRL) contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, AFRL or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This work was also partially supported by EPSRC Project EP/T024976/1 (Unmute).

6. References

[1] S. Thomas, S. Ganapathy, and H. Hermansky, "Cross-lingual and multi-stream posterior features for low resource lvcsr systems," in *Interspeech*, 2010.

[2] F. Grézl, M. Karafiat, and M. Janda, "Study of probabilistic and bottle-neck features in multilingual environment," in *ASRU*, 2011.

[3] K. Veselý, M. Karafiát, F. Grézl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *SLT*, 2012.

[4] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep neural networks," in *ICASSP*, 2013.

[5] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *ICASSP*, 2013.

[6] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *ICASSP*, 2013.

[7] A. Ragni and M. Gales, "Automatic speech recognition system development in the" wild"," in *Interspeech*, 2018.

[8] A. Carmantini, P. Bell, and S. Renals, "Untranscribed web audio for low resource speech recognition." in *Interspeech*, 2019.

[9] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NeurIPS*, 2020.

[10] E. Yılmaz, H. van den Heuvel, and D. Van Leeuwen, "Investigating bilingual deep neural networks for automatic recognition of code-switching frisian speech," *Procedia Computer Science*, vol. 81, pp. 159–166, 2016.

[11] S. Dalmia, Y. Liu, S. Ronanki, and K. Kirchhoff, "Transformer-transducers for code-switched speech recognition," in *ICASSP*, 2021.

[12] Y. Li and P. Fung, "Code-switch language model with inversion constraints for mixed language speech recognition," in *COLING*, 2012.

[13] 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 *ICASSP*, 2013.

[14] E. Yilmaz, S. Cohen, X. Yue, H. Li, and D. van Leeuwen, "Multi-graph decoding for code-switching asr," in *Interspeech*, 2019.

[15] Y. Sharma, B. Abraham, K. Taneja, and P. Jyothi, "Improving low resource code-switched asr using augmented code-switched tts," in *Interspeech*, 2020.

[16] A. Diwan, R. Vaideeswaran, S. Shah, A. Singh, S. Raghavan, S. Khare, V. Unni, S. Vyas, A. Rajpuria, C. Yarra, A. Mittal, P. K. Ghosh, P. Jyothi, K. Bali, V. Seshadri, 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," in *Interspeech*, 2021.

[17] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free MMI," in *Interspeech*, 2016.

[18] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Semi-supervised training of acoustic models using lattice-free MMI," in *ICASSP*, 2018.

[19] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks." in *Interspeech*, 2018.

[20] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE TASLP*, vol. 19, no. 4, pp. 788–798, 2010.

[21] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *ASRU*, 2013.

[22] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of DNNs with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.

[23] 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," *Interspeech*, 2019.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *ICASSP*, 1995.

[26] A. Stolcke, "SRILM-an extensible language modeling toolkit," in *ICSLP*, 2002.

[27] ——, "Entropy-based pruning of backoff language models," in *Proc. of DARPA Broadcast News Transcription and Understanding Workshop, 1998*, 1998.

[28] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, 2010.

[30] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors." in *Odyssey*, 2018.

[31] J. Valk and T. Alumäe, "VoxLingua107: A dataset for spoken language recognition," in *SLT*, 2021.

[32] V. Manohar, D. Povey, and S. Khudanpur, "JHU kaldi system for Arabic MGB-3 ASR challenge using diarization, audio-transcript alignment and transfer learning," in *ASRU*, 2017.

[33] P. Ghahremani, V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Investigation of transfer learning for ASR using LF-MMI trained neural networks," in *ASRU*, 2017.

[34] "The CMU pronunciation dictionary," *http://www.speech.cs.cmu.edu*, 1995.

[35] M. Harper, "IARPA babel program," *http://www.iarpa.gov/Programs/ia/Babel/babel.html.*

[36] A. W. Black and K. A. Lenzo, "Flite: a small fast run-time synthesis engine," in *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 2001.

[37] A. Parlikar, S. Sitaram, A. Wilkinson, and A. W. Black, "The festvox indic frontend for grapheme to phoneme conversion," in *WILDRE: Workshop on Indian Language Data-Resources and Evaluation*, 2016.

[38] K. Knight and J. Graehl, "Machine transliteration," in *EACL*, 1997.

[39] S. Ravi and K. Knight, "Learning phoneme mappings for transliteration without parallel data," in *NAACL-HLT*, 2009.

[40] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[41] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *International Conference on Implementation and Application of Automata*, 2007.

[42] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The OpenGrm open-source finite-state grammar software libraries," in *ACL System Demonstrations*, 2012.

[43] J. R. Novak, N. Minematsu, and K. Hirose, "WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding," in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, 2012.

[44] J. L. Lee, L. F. Ashby, M. E. Garza, Y. Lee-Sikka, S. Miller, A. Wong, A. D. McCarthy, and K. Gorman, "Massively multilingual pronunciation modeling with WikiPron," in *LREC*, 2020.