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

Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages

Anuj Diwan, Preethi Jyothi

Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India
{anujdiwan,pjyothi}@cse.iitb.ac.in

Abstract

This work presents a seemingly simple but effective technique to improve low-resource ASR systems for phonetic languages. By identifying sets of acoustically similar graphemes in these languages, we first reduce the output alphabet of the ASR system using linguistically meaningful reductions and then reconstruct the original alphabet using a standalone module. We demonstrate that this lessens the burden and improves the performance of low-resource end-to-end ASR systems (because only reduced-alphabet predictions are needed) and that it is possible to design a very simple but effective reconstruction module that recovers sequences in the original alphabet from sequences in the reduced alphabet. We present a finite state transducer-based reconstruction module that operates on the 1-best ASR hypothesis in the reduced alphabet. We demonstrate the efficacy of our proposed technique using ASR systems for two Indian languages, Gujarati and Telugu. With access to only 10 hrs of speech data, we obtain relative WER reductions of up to 7% compared to systems that do not use any reduction.

Index Terms: ASR for low-resource languages, end-to-end ASR models.

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) systems are becoming an increasingly popular choice for ASR modeling [1]. E2E systems directly map speech to sequences of graphemes or subword units derived from graphemes. This direct treatment of the ASR problem makes E2E modeling an attractive choice for low-resource and high-resource languages alike. However, E2E ASR systems are very data intensive and consequently tend to underperform on low-resource languages for which labelled speech data is scarce. Identifying techniques that can help boost E2E performance for low-resource languages is of great interest. We offer one such approach that we refer to as *Reduce and Reconstruct* (RNR).

To motivate RNR, we start with reiterating the goal of E2E models. These models aim at learning direct mappings from speech to graphemes. In low-resource settings, there may not be sufficient amounts of data to learn these mappings reliably. This issue could be alleviated by meaningfully reducing the size of the output vocabulary using a simple rule-based system that is linguistically motivated. This is the *Reduce* step in RNR. The resulting alphabet should be compact, while also being sufficiently discriminative across speech sounds. With this reduced alphabet in place, we now have an easier task to be learned by the E2E models. Since the predictions from the E2E models will be in the reduced alphabet, one would require an additional reconstruction module (i.e. the *Reconstruct* step in RNR) to recover the original grapheme sequence. We tackle the reconstruction problem using WFSTs, since the *Reduce* step is a deterministic mapping from the original grapheme alphabet to the reduced alphabet which can be appropriately modelled using WFSTs. We note here that RNR is well-suited for E2E models that learn direct mappings from speech to graphemes. For traditional cascaded ASR systems, a reduced alphabet might actually be counterproductive with introducing more confusability across words in the lexicon.

Consider the following examples that illustrate the utility of RNR. Suppose our reference text is “call the bus" and we train an ASR model with a reduced output vocabulary $\Re$ that uses a single character ζ to represent both “c" and “k" in the original English alphabet. If the ASR model is able to accurately predict “ζall", then reconstruction will trivially map “ζall" to “call" since “kall" is not a valid word in English. A more interesting case is when there is more than one valid word that a reduced word form can map to. Say our reference text is “game is on" and $\Re$ has a single character κ that maps to “c", “k" and “g". If the ASR model correctly predicts κ, “κame" could be mapped to either “came" or “game". However, the language model in the reconstruction module should accurately pick “game is on" as the more likely prediction given its higher likelihood.

In this work, we consider two low-resource Indian languages, Gujarati and Telugu, to demonstrate the use of RNR. RNR can be easily applied to any language (like Telugu, Gujarati, etc.) that is largely phonetic with one-to-one mappings between phones and graphemes. We first identify sets of acoustically similar graphemes (graphemes that represent acoustically similar phones) in each language to produce an appropriate reduction, train an ASR system with the reduced vocabulary and finally reconstruct the best ASR prediction using an FST-based reconstruction module.

2. Related Work

Our line of work is closely related to error correction models for ASR that explicitly correct errors made by the ASR system [2]. Several such efforts for error correction of ASR systems have been carried out in prior work; [3] presents a review. Past work has looked at reordering ASR hypotheses using machine translation-inspired techniques [4, 5], leveraging contextual information using recurrent models and rescoring confusion networks generated by the ASR system [6, 7, 8, 9, 10, 11]. Our proposed framework is different in that it is not strictly a postprocessing technique and modifies the ASR system itself to use a reduced alphabet. More recent work has looked at correcting errors made by E2E models by training either sequence-to-sequence or transformer-based models [12, 13, 14].

Another different but related line of work investigates the ways in which sounds from different languages can be merged effectively when building multilingual ASR systems [15]. Attempts at redefining the phone set in a data-driven manner have also been made for monolingual ASR systems [16]. Recent work has looked at the influence of merging phones for code-switched speech recognition [17] and building language-agnostic multilingual systems by mapping graphemes in Indian languages to a single script [18].

http://dx.doi.org/10.21437/Interspeech.2021-644

3. RNR: Reduce and Reconstruct

We first devise a many-to-one reduction mapping where each grapheme in the original alphabet is mapped to a reduced grapheme, thus creating a new reduced alphabet $\Re$. An ASR model is trained using transcriptions in $\Re$. The 1-best decoded output from this model is fed as input to a reconstruction module that is trained to recover transcriptions in the original alphabet from the reduced alphabet. We note that using the n-best decoded outputs ($n = 100$) from the ASR model, for reconstruction, resulted in exactly the same performance as just using the 1-best, so we use the latter in all our experiments.

3.1. Reduction Functions

We aim to group together graphemes that are acoustically similar and replace each such set with a reduced grapheme. We manually design such reduction functions by referring to the phonology of Gujarati and Telugu. This task is simple for phonetic languages, since they have mostly one-to-one mappings between graphemes and phonemes.[1]

Both Gujarati and Telugu have five different types of plosives that vary in their place of articulation (labial, alveolar, retroflex, palatal and velar). Each of these plosives can be voiced or unvoiced and further, aspirated or unaspirated yielding a total of 20. For a given place of articulation, we merge the graphemes corresponding to all four plosives into a single grapheme in $\Re$.[2] The graphemes corresponding to the twenty plosives are reduced to five graphemes in $\Re$. There are five graphemes corresponding to nasal sounds in Gujarati and Telugu, which we reduce down to a single grapheme. In both languages, long and short vowel variants corresponding to a particular sound are merged into a single grapheme. All other graphemes are left as-is in $\Re$. This reduction will henceforth be referred to as ρ_1. This reduces the grapheme alphabet size to 27 from 63 for Gujarati and to 27 from 70 for Telugu.

3.2. FST-based Reconstruction

Our reconstruction model is implemented using a cascade of FSTs. This structure is somewhat similar to the decoder in [21], although the specific FSTs used in our work and the motivation for their design are very different. The input to the reconstruction model is an ASR hypothesis consisting of a sequence of reduced graphemes. Let this input be represented as a linear chain acceptor H. H is transduced to an output FST O using a series of FST compositions, followed by an invocation of the $\mathrm{shortestpath}$ algorithm on the composed FST:

$$O = \mathrm{shortestpath}(H \circ S \circ E \circ L \circ G)$$

The FSTs S, E, L and G (described below) are weighted using negative log probability costs, thus making $\mathrm{shortestpath}$ appropriate. The output labels of O will correspond to the best reconstructed word sequence in the original alphabet. An important way in which our FST-based approach differs from the standard $HCLG$ FST pipeline [22] is that we do not require a pronunciation lexicon and operate only with grapheme sequences.

S: Reduction FST. S is a 1-state machine that takes reduced graphemes as input and produces original graphemes as its output. S contains zero-cost transitions corresponding to pairs $(g, \hat{g})$ where g is a reduced grapheme in $\Re$ and $\hat{g}$ is its counterpart in the original alphabet. $H \circ S$ would exhaustively produce all possible reconstructions of the reduced word forms in H.

E: Edit distance FST. By examining the ASR errors, we observe that many predicted words differ from the correct words in only a small number of graphemes. To accommodate this, we design an edit distance FST E that takes a space-separated grapheme sequence as input and produces as output all reduced word forms that are within an edit distance of d from each word in the input. The allowable edits are substitutions, insertions and deletions. Each edit incurs an additive cost λ. Both d and λ are tunable hyperparameters. $H \circ S \circ E$ will now produce all possible reconstructions of the reduced word forms in H with zero cost and also other words with cost λe ($1 \leq e \leq d$) that are at an edit distance e from the exact reconstructions.

L: Dictionary FST. The dictionary FST maps sequences of graphemes to sequences of words. This machine is similar in structure to a pronunciation lexicon FST (that maps sequences of phonemes to sequences of words) [22]. However, note that the similarity is only structural; we map graphemes to words, so lexicons are not required. The input vocabulary of L comprises graphemes from the original alphabet and the output vocabulary of L corresponds to words from the training vocabulary of the ASR system. We also include an `unk` word in the output vocabulary to accommodate out-of-vocabulary words that might appear in the evaluation data. Any grapheme sequence can map to the `unk` word, but this is associated with a very large penalty η. This prevents in-vocabulary words from opting for paths involving `unk`.

G: Language model FST. Finally, we have an N-gram language model acceptor that rescores word sequences from $H \circ S \circ E \circ L$. The N-gram language model is trained on the training set transcriptions of the full speech data.[3]

4. Experiments and Results

Dataset Details. We use the Microsoft Speech Corpus (Indian Languages) dataset [23] [24]. The Gujarati and Telugu speech corpora comprise 39.1 hours and 31.3 hours of training speech, respectively. The dev/test splits for Gujarati and Telugu contain 500/3075 and 500/3040 utterances, respectively. The OOV rates for Gujarati and Telugu on the test set are 5.2% and 12.0%, respectively.

Experimental Setup. ASR systems for both languages are built using the ESPNet Toolkit [25]. We use 80-dimensional log-mel acoustic features with pitch information. Our ASR model is a hybrid LSTM-based CTC-attention model [26]. All our ASR systems use BPE tokenization [27], with BPE-based output vocabularies of size 5000. We use a CTC weight of 0.8 and an attention weight 0.2 with a dropout rate of 0.2. For both languages, we use 4 encoder layers, 1 decoder layer and location-based attention with 10 convolutional channels and 100 filters. For Gujarati, we use 512 encoder units, 300 de-

[1] In this work, while we evaluate on two languages with phonetic orthographies, it should be possible to use RNR with any language for which there is sufficient linguistic expertise to deterministically map its character inventory down to a reduced "pseudo-phone" set.

[2] Motivated by prior work on phoneme confusions in ASR (e.g., [19]), we used place of articulation as the main dimension along which to collapse phonemes. We leave for future work more detailed investigations of determining the best way in which the reduced alphabet can be constructed [20].

[3] We observed no additional benefits in performance with using larger text corpora to train G.

Table 1: *Reduced WERs (r-WER) on Gujarati and Telugu for different training set durations.*

Duration	Reduction	r-WER (Guj)		r-WER (Tel)	
		Dev	**Test**	**Dev**	**Test**
Full	identity	41.5	43.2	44.1	46.8
	ρ_1	36.5	39.6	39.3	42.8
	ρ_1-rand	41.3	42.3	44.2	47.9
10 hr	identity	60.2	68.6	64.1	71.4
	ρ_1	53.9	63.6	56.9	66.5
	ρ_1-rand	63.2	71.8	60.8	69.4

coder units and an attention dimension of 320. For Telugu, we use 768 encoder units, 450 decoder units and an attention dimension of 250. We use a beam decoder during inference with a beam width of 45. All models are trained on an Nvidia GeForce GTX 1080 Ti GPU.

All the reconstruction FSTs are implemented using the OpenFST toolkit [28]. The G FST represents a Kneser-Ney smoothed 4-gram LM that is trained using the SRILM toolkit [29]. An edit distance of $d = 3$ and a cost of $\lambda = 5$ were the best values obtained for the WERs using the full training set duration, by tuning on the dev set.

4.1. ASR Experiments

Table 1 lists reduced WERs for both Gujarati and Telugu on the dev and test sets using different reduction functions and trained on two different train durations; "Full" refers to the complete train set and "10 hr" refers to a randomly sampled 10 hour subset. Recall that for a given reduction function, we train an ASR system with the reduction applied to the ground truth text. Since the above WERs are computed between the hypotheses and the *reduced* ground truth text, and because the output alphabet is different for each reduction function, these are not standard WERs but are rather *reduced* WERs.

Identity refers to the baseline system with an unaltered grapheme set. ρ_1 is the reduction function discussed in Section 3. ρ_1-rand is designed to show the importance of a linguistically meaningful reduction. In ρ_1-rand, graphemes are randomly merged together while ensuring that the size of the final alphabet matches the size of the alphabet after applying the reduction ρ_1.

The baseline identity WERs are comparable to previously published results [30] using LSTM-based E2E architectures for the Microsoft Speech Corpus (Indian Languages) dataset. The r-WER for ρ_1 is lower than the r-WER from the identity system, which shows that the ASR system performs an easier task while training on the reduced alphabet. However, the r-WER of ρ_1-rand is significantly worse than ρ_1, confirming our claim that linguistically motivated reductions are key to derive performance improvements.

4.2. FST-based Reconstruction Experiments

Table 2 shows the WERs for both Gujarati and Telugu using our FST-based reconstruction model with the two training durations. Recall that d is the edit distance permitted by the edit distance FST and λ is the associated edit cost. $d = 0$ thus means only allowing for words in the original alphabet that can be exactly recovered from the reconstructed word forms. We find that in this scenario, reconstruction with the ρ_1 mapping for $d = 0$ significantly outperforms both the baseline and the identity reduction. As we increase d from 0 to 3, the power of reconstruction increases due to the E FST and we observe large reductions in WER for both ρ_1 and identity. Even with this best reconstruction system, the ρ_1 mapping has significantly lower WERs than the corresponding identity mapping. In the $d = 3, \lambda = 5$ setting, averaged across both languages, we observe a 3.2% relative test WER decrease for the Full duration and a 4.8% relative test WER decrease for the 10-hr duration. The benefits of our approach are more pronounced in the 10 hour setting, reaffirming its utility for low-resource languages.

4.3. Reconstruction after RNNLM rescoring

We examine whether RNR is effective even after using an RNNLM rescoring during decoding. We train a 2-layer RNNLM (with 1500 hidden units each) on the training transcriptions of the full speech data to rescore the predictions from the ASR systems during beam decoding. Table 3 lists WERs for both Gujarati and Telugu using the best reconstruction system ($d = 3, \lambda = 5$) for both training durations. With the RNNLM in place, we observe a significant improvement in performance of the baseline system that uses no reconstruction. In the 10-hr setting, our approach using ρ_1 performs significantly better than the baseline and the identity mapping for both languages, yielding up to 7% relative reduction in test WERs. In the full setting, Gujarati does not benefit from RNR while Telugu still yields improvements on test WER. It is clear that the improvements owing to RNR are more pronounced in very low-resource settings. This trend was observed in Table 2 as well.

Table 3: *WERs for Gujarati and Telugu using $d = 3, \lambda = 5$, in conjunction with RNNLM rescoring*

Duration	Reduction	WER (Guj)		WER (Tel)	
		Dev	**Test**	**Dev**	**Test**
Full	Baseline	37.4	34.0	37.9	40.0
	identity	**36.2**	**31.8**	37.7	39.2
	ρ_1	37.1	32.2	**36.5**	**38.1**
10-hr	Baseline	56.2	63.2	56.9	63.8
	identity	55.5	62.3	56.2	62.5
	ρ_1	**52.0**	**58.2**	**51.2**	**59.1**

Table 4: *Results for Gujarati 10-hr using a conformer model*

Reduction	r-WER (Guj)	
	Dev	**Test**
identity	57.7	61.1
ρ_1	56.4	59.5
ρ_1-rand	59.6	62.4

(a) *r-WER on ASR predictions*

d	λ	Reduction	WER (Guj)	
			Dev	**Test**
Baseline			57.7	61.1
0	10	identity	57.7	61.1
		ρ_1	57.9	60.4
3	10	identity	**57.1**	60.5
		ρ_1	57.6	**59.9**

(b) *WERs after reconstruction*

4.4. Using conformers

To demonstrate that RNR is effective across different E2E architectures, we also train a conformer-based ESPNet model [31]

Table 2: *WERs from the FST-based reconstruction model for Gujarati and Telugu for two training durations.*

d	λ	Reduction	WER (Guj)		WER (Tel)	
			Dev	**Test**	**Dev**	**Test**
		Baseline	41.5	43.2	44.1	46.8
0	5	identity	41.8	43.4	45.1	47.7
		ρ_1	40.4	41.9	42.1	45.7
3	5	identity	37.9	37.8	40.6	42.5
		ρ_1	**37.8**	**36.5**	**38.5**	**41.2**

(a) *Full training duration.*

d	λ	Reduction	WER (Guj)		WER (Tel)	
			Dev	**Test**	**Dev**	**Test**
		Baseline	60.2	68.6	64.1	71.4
0	5	identity	60.3	68.6	64.4	71.6
		ρ_1	56.2	64.9	58.4	67.8
3	5	identity	56.8	64.9	59.2	66.1
		ρ_1	**53.2**	**61.2**	**54.3**	**63.6**

(b) *10-hr training duration.*

for Gujarati 10-hours shown in Table 4. We use 2 encoder layers and 1 decoder layer, each with 350 units and 4 attention heads. We use a CTC weight of 0.3 with a dropout rate of 0.1. Following similar trends as in the hybrid models, we find ρ_1 to be much more effective as a reduction compared to ρ_1-rand and reconstruction with ρ_1 yields WER improvements over the identity system on the test set.

5. Discussion and Qualitative Analysis

5.1. Choice of reduction function

One might wonder about the impact of the reduction function on ASR performance. In Table 1, we show how a randomly chosen ρ_1-rand can be detrimental to the ASR system. Additionally, we employ a third reduction function ρ_2 that is less compressive than ρ_1. ρ_2 reduces pairs of aspirated/unaspirated plosives into a single grapheme and does not merge long and short vowel variants, resulting in a $\Re$ of size 48 for Gujarati and size 55 for Telugu. With ρ_2, on the 10-hr training set with a $(d = 0, \lambda = 5)$ system, we obtain WERs of 67.8% and 67.9% on the Gujarati and Telugu test sets, respectively, which is worse compared to 64.9% and 67.8% obtained with ρ_1. This shows that a reduction function like ρ_1, that is more compressive than ρ_2, is more beneficial with RNR.

5.2. Effect of reduction function on correcting ASR errors

Consider the ASR system (from Section 4.1) trained on the 10-hr training subset. Of the substitution errors made by the identity system, we compute the percentage of errors that were corrected by the reduced predictions. We first align the identity hypothesis text to the reference text to give an alignment a_1. We similarly align the reduced hypothesis text to the reduced reference text; call it a_2. Let X be the number of character substitutions $a \rightarrow b$ (where $a \neq b$) in the identity system alignment a_1. Among these, we count the number of substitutions Y for which the corresponding character alignment in a_2 is correctly predicted and compute $\frac{Y}{X} \times 100\%$. Of all the substitution errors made by the identity system, 16.29% (for Gujarati) and 16.92% (for Telugu) of the errors were corrected by the reduced predictions. This shows that the reduced system is able to fix many substitution errors incurred by the identity system.

5.3. Test set perplexities before and after reduction

As an information-theoretic measure of reduction , we compute the perplexity (ppl) of the test set using a trigram LM trained on the training set, in both the original and reduced vocabularies in Table 5. The ppl reductions show that our reduced vocabulary makes the language model-based predictions more accurate; the larger drop in ppl for Telugu also correlates with the larger improvement in WERs for Telugu compared to Gujarati.

Table 5: *Test set perplexities using a trigram LM*

Reduction	Test ppl (Guj)	Test ppl (Tel)
identity	115.05	768.66
ρ_1	108.13	706.32

5.4. Illustrative Examples

Figure 1: *An illustrative example each from Gujarati and Telugu using FST reconstruction,* $\mathbf{d} = 3, \lambda = 5$. *$R$ and I are the reference and identity transcriptions.*

Figure 1 shows two examples of RNR. The first example highlights nasal/plosive sounds in Gujarati that were correctly recognized by ρ_1, while the identity system recognizes a different nasal/plosive sound that is acoustically confusable and part of the reduction map in ρ_1. The second example shows a long vowel in Telugu mistakenly recognized as a short vowel. Here again, ρ_1 merges these two graphemes and hence accurately recovers the correct form.

6. Conclusions

This work proposes an effective reduce-and-reconstruct technique that can be used with any ASR system for low-resource phonetic languages. We demonstrate its utility for two Indian languages and show that as the available training data decreases, our approach yields greater benefits, making it well-suited for low-resource languages. Future work will investigate the use of more powerful Transformer-based [32] reconstruction models and automatically learning a reduction mapping based on errors made by the ASR system.

7. Acknowledgements

The last author is grateful to IBM Research, India (i.e., the IBM AI Horizon Networks - IIT Bombay initiative) for their support.

8. References

[1] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[2] E. K. Ringger and J. F. Allen, "Error correction via a post-processor for continuous speech recognition," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1. IEEE, 1996, pp. 427–430.

[3] R. Errattahi, A. E. Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," in *Procedia Computer Science*, 2018.

[4] H. Cucu, A. Buzo, L. Besacier, and C. Burileanu, "Statistical error correction methods for domain-specific ASR systems," in *International Conference on Statistical Language and Speech Processing*. Springer, 2013, pp. 83–92.

[5] L. F. D'Haro and R. E. Banchs, "Automatic correction of ASR outputs by using machine translation," *proceedings Interspeech 2016*, pp. 3469–3473, 2016.

[6] A. Sarma and D. D. Palmer, "Context-based speech recognition error detection and correction," in *Proceedings of HLT-NAACL 2004: Short Papers*. Association for Computational Linguistics, 2004, pp. 85–88.

[7] S. Jung, M. Jeong, and G. G. Lee, "Speech recognition error correction using maximum entropy language model," in *Eighth International Conference on Spoken Language Processing*, 2004.

[8] Y.-C. Tam, Y. Lei, J. Zheng, and W. Wang, "ASR error detection using recurrent neural network language model and complementary ASR," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 2312–2316.

[9] R. Nakatani, T. Takiguchi, and Y. Ariki, "Two-step correction of speech recognition errors based on n-gram and long contextual information." in *INTERSPEECH*, 2013, pp. 3747–3750.

[10] E. Byambakhishig, K. Tanaka, R. Aihara, T. Nakashika, T. Takiguchi, and Y. Ariki, "Error correction of automatic speech recognition based on normalized web distance," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[11] Y. Fusayasu, K. Tanaka, T. Takiguchi, and Y. Ariki, "Word-error correction of continuous speech recognition based on normalized relevance distance," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[12] J. Guo, T. N. Sainath, and R. J. Weiss, "A Spelling Correction Model for End-to-end Speech Recognition," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 5651–5655, 2019.

[13] O. Hrinchuk, M. Popova, and B. Ginsburg, "Correction of Automatic Speech Recognition with Transformer Sequence-To-Sequence Model," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 7074–7078.

[14] S. Zhang, M. Lei, and Z. Yan, "Automatic spelling correction with transformer for ctc-based end-to-end speech recognition," 2019.

[15] L. Melnar and J. Talley, "Phone merger specification for multilingual ASR: the Motorola polyphone network," in *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS'03)*, 2003, pp. 1337–1340.

[16] R. Singh, B. Raj, and R. M. Stern, "Structured redefinition of sound units by merging and splitting for improved speech recognition," in *Sixth International Conference on Spoken Language Processing*, 2000.

[17] S. Sivasankaran, B. M. L. Srivastava, S. Sitaram, K. Bali, and M. Choudhury, "Phone Merging For Code-Switched Speech Recognition," in *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 11–19. [Online]. Available: https://www.aclweb.org/anthology/W18-3202

[18] 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)*, May 2020, pp. 8239–8243.

[19] B. T. Meyer, M. Wächter, T. Brand, and B. Kollmeier, "Phoneme confusions in human and automatic speech recognition," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[20] T. M. Bailey and U. Hahn, "Phoneme similarity and confusability," *Journal of Memory and Language*, vol. 52, no. 3, pp. 339–362, 2005.

[21] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.

[22] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[23] Microsoft, "Microsoft Speech Corpus (Indian languages)," February 2020. [Online]. Available: https://msropendata.com/datasets/7230b4b1-912d-400e-be58-f84e0512985e

[24] B. Srivastava, S. Sitaram, R. Mehta, K. Mohan, P. Matani, S. Satpal, K. Bali, R. Srikanth, and N. Nayak, "Interspeech 2018 Low Resource Automatic Speech Recognition Challenge for Indian Languages," 08 2018, pp. 11–14.

[25] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-End Speech Processing Toolkit," in *Interspeech*, 2018, pp. 2207–2211. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1456

[26] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[27] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016. [Online]. Available: http://dx.doi.org/10.18653/v1/P16-1162

[28] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-State Transducer Library," in *Proceedings of the 12th International Conference on Implementation and Application of Automata*, ser. CIAA'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 11–23.

[29] A. Stolcke, "SRILM-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[30] V. M. Shetty, J. MetildaSagayaMaryN., and S. Umesh, "Improving the performance of transformer based low resource speech recognition for indian languages," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8279–8283, 2020.

[31] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," 2020.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf