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.

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].
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 \( \mathcal{R} \). An ASR model is trained using transcriptions in \( \mathcal{R} \). 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 \( \mathcal{R} \).\(^2\) The graphemes corresponding to the twenty plosives are reduced to five graphemes in \( \mathcal{R} \). 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 \( \mathcal{R} \). This reduction will henceforth be referred to as \( \rho_1 \). 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 reduction model is an ASR hypothesis consisting of a sequence of grapheme in the original alphabet is mapped to a reduced grapheme, so 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 \( \text{unk} \) 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 \( \text{unk} \) word, but this is associated with a very large penalty \( \eta \). This prevents in-vocabulary words from opting for paths involving \( \text{unk} \).

\( G \): Language model FST. Finally, we have an \( N \)-gram language model acceptor that rescores word sequences from \( H \circ S \circ E \circ O \). 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 \( G \).
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. $\rho_1$ is the reduction function discussed in Section 3. $\rho_{1\text{-rand}}$ is designed to show the importance of a linguistically meaningful reduction. In $\rho_{1\text{-rand}}$, graphemes are randomly merged together while ensuring that the size of the final alphabet matches the size of the alphabet after applying the reduction $\rho_1$.

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 $\rho_1$ 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 $\rho_{1\text{-rand}}$ is significantly worse than $\rho_1$, 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 $\lambda$ 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 $\rho_1$ 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 $\rho_1$ and identity. Even with this best reconstruction system, the $\rho_1$ 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 $R\lambda R$ 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 $\rho_1$ 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 $R\lambda R$ while Telugu still yields improvements on test WER. It is clear that the improvements owing to $R\lambda R$ are more pronounced in very low-resource settings. This trend was observed in Table 2 as well.

4.4. Using conformers

To demonstrate that $R\lambda R$ is effective across different E2E architectures, we also train a conformer-based ESPNet model [31] for Gujarati 10 hr using a conformer model.

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

<table>
<thead>
<tr>
<th>Duration</th>
<th>Reduction</th>
<th>WER (Guj)</th>
<th>WER (Tel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>Dev</td>
<td>Test</td>
</tr>
<tr>
<td>Full</td>
<td>identity</td>
<td>37.4</td>
<td>34.0</td>
</tr>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>36.2</td>
<td>31.8</td>
</tr>
<tr>
<td>10-hr</td>
<td>identity</td>
<td>56.2</td>
<td>63.2</td>
</tr>
<tr>
<td></td>
<td>$\rho_1$</td>
<td>55.5</td>
<td>62.3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Reduction</th>
<th>r-WER (Guj)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev</td>
</tr>
<tr>
<td>identity</td>
<td>57.7</td>
</tr>
<tr>
<td>$\rho_1$</td>
<td>56.4</td>
</tr>
<tr>
<td>$\rho_{1\text{-rand}}$</td>
<td>59.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d \lambda$</th>
<th>Reduction WER (Guj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>identity $\rho_1$</td>
</tr>
<tr>
<td>0</td>
<td>57.7</td>
</tr>
<tr>
<td>1</td>
<td>identity $\rho_1$</td>
</tr>
<tr>
<td>1</td>
<td>57.6</td>
</tr>
</tbody>
</table>

(a) r-WER on ASR predictions
(b) WERs after reconstruction

4.4. Using conformers

To demonstrate that $R\lambda R$ is effective across different E2E architectures, we also train a conformer-based ESPNet model [31]
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 ρ1-rand can be detrimental to the ASR system. Additionally, we employ a third reduction function ρ2 that is less compressive than ρ1. ρ2 reduces pairs of aspirated/unaspirated plosives into a single grapheme and does not merge long and short vowel variants, resulting in a R of size 48 for Gujarati and size 55 for Telugu. With ρ2, on the 10-hr training set with a (d = 0, λ = 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 ρ1. This shows that a reduction function like ρ1, that is more compressive than ρ2, 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 a1. We similarly align the reduced hypothesis text to the reduced reference text; call it a2. Let X be the number of character substitutions a → b (where a ≠ b) in the identity system alignment a1. Among these, we count the number of substitutions Y for which the corresponding character alignment in a2 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.

5.4. Illustrative Examples

Figure 1 shows two examples of R\(R\). The first example highlights nasal/plosive sounds in Gujarati that were correctly recognized by ρ1, while the identity system recognizes a different nasal/plosive sound that is acoustically confusable and part of the reduction map in ρ1. The second example shows a long vowel in Telugu mistakenly recognized as a short vowel. Here again, ρ1 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


