INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

BACHELOR THESIS

Multilingual, Code-switching and Low-Resource NLP and ASR

Author:
Anuj Jitendra Diwan
170070005
Department of Computer Science

Supervisor:
Prof. Preethi Jyothi
Assistant Professor
Department of Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology

in the

Department of Computer Science and Engineering

September 10, 2021

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Abstract

Multilingual, Code-switching and Low-Resource NLP and ASR

by
Anuj Jitendra Diwan
170070005
Department of Computer Science

Many advances in NLP and Speech are powered by availability of data. Only a small fraction of languages that have access to large quantities of data (high-resource languages) benefit from these advances [1]. Data that captures a linguistic phenomenon like code-switching is scarcely available. Developing less data-intensive techniques for low-resource and code-switching languages thus demands radically new approaches. In fact, developing such techniques is extremely important for making technology more inclusive for multilingual speakers of diverse languages. In this thesis, we describe the progress and outcomes of four projects to improve speech recognition for low-resource languages:

1. **Transliteration-based Transfer:** Developing a novel transfer learning strategy that pretrains an ASR model using speech data from a high-resource language with its text transliterated to the target low-resource language.

2. **Indian Languages' Interspeech 2021 Special Session:** Organizing the 'Multilingual and code-switching ASR Challenges for low resource Indian languages' Special Session at Interspeech 2021. I was one of the primary technical contributors and the first author on the challenge description paper.

3. **Bridging Scripts by Grounding in Speech:** Using monolingual labelled speech data in a high-resource language and a low-resource language to build a high quality transliteration system i.e. grounding transliteration in speech. This can be used to improve the low-resource ASR system via high- to low-resource transfer learning.

4. **Reduce and Reconstruct:** Incorporating linguistic knowledge into low-resource speech recognition by developing linguistically inspired reduced vocabularies.

Parts of this thesis were submitted and accepted as papers in Interspeech 2021. These papers are [2] [3] [4].

Acknowledgements

I am extremely grateful to my advisor, Prof. Preethi Jyothi, for being such a wonderful mentor, guiding and motivating me throughout this journey. I am very thankful to her for introducing me to the field of Speech Recognition and Natural Language Processing and for encouraging me to work on this important and socially relevant research problem of developing less data-intensive techniques for low-resource languages. I also thank her for allowing access to the CSALT computational cluster, where I was able to run all my experiments.

I am also thankful to Prof. Sunita Sarawagi for the many enlightening discussions and interesting ideas over the course of the projects. My machine learning knowledge has vastly expanded because of her. I also thank her for allowing access to her group's computational cluster (with cute 3-lettered names like `ant` and `bee`) where I was able to run large-scale experiments.

I also thank Ashish Mittal, Shreya Khare, Samarth Bharadwaj, Tejas Dhamecha at IBM Research; my colleagues Tarang Jain, Vinit S. Unni, Ankita Singh at IIT Bombay; and organizers of the Multilingual ASR Interspeech 2021 Special Session for an amazing and fulfilling collaboration.

I finally thank my parents and grandparents for supporting and motivating me throughout this journey, as most of this thesis was completed during the work-from-home Autumn 2020 and Spring 2021 online semesters.

Contents

Chapter 1

Introduction

Many advances in NLP and Speech are increasingly powered by availability of data. For example, acoustic models for state-of-the-art speech recognition systems are usually trained on hundreds of hours of task-specific training data [5]. However, a majority of the 7000 languages of the world do not have access to vast quantities of data. Therefore, only a small fraction of languages that have access to such data (high-resource languages) benefit from these state-of-the-art advances [1]. Adding to the data disparity, data that captures a linguistic phenomenon like code-switching is scarcely available. Developing less data-intensive techniques that are capable of learning low-resource and code-switching language tasks demands radically new approaches. In this thesis, we explore four distinct approaches to improve speech recognition for low-resource languages.

Chapters 2, 3, 4 and 5 explore each approach in detail. Chapter 6 explores other work that was done as part of the BTP. We conclude in Chapter 7. Appendix A contains notes summarizing the various papers that were read during the B.Tech Project.

1.1 Low Resource ASR: The surprising effectiveness of High Resource Transliteration

Although a target low-resource language may have limited data, using transfer learning techniques, it is possible to use larger quantities of data from related/unrelated high-resource languages and transfer this knowledge to the low-resource language task.

We propose a novel transfer learning strategy that pretrains a given speech recognition model using labelled speech data from a high-resource language, but with its text transliterated to the target low-resource language, followed by finetuning on the limited low-resource labelled speech data. This transliteration encourages increased sharing between output spaces of the two languages and results in significant improvements.

In Chapter 2, we explore this approach and evaluate it using a transformer ASR architecture and the state-of-the-art wav2vec2.0 ASR architecture. We use English as the high-resource language and six languages Hindi, Gujarati, Telugu, Bengali, Amharic and Korean as low-resource targets. With access to 1 hour of target speech, we obtain relative WER reductions of up to 8.2% compared to existing transfer-learning approaches.

This work was done in collaboration with IBM Research and submitted to Interspeech 2021 [2].

1.2 Multilingual and Code-switching ASR Challenges for Low Resource Indian Languages

Promoting public participation and encouraging research in one's subfield is an important task that garners interest and fuels innovation in one's subfield. To that end, I joined the organizing committee and am one of the primary technical contributors for the 'Multilingual and Code-switching ASR Challenges for Low Resource Indian languages' Special Session at Interspeech 2021.

The Special Session consisted of two speech recognition tasks for low-resource Indian language ASR and code-switching ASR. As part of this challenge, 600 hrs of speech data in 6 Indian languages and 2 code-mixing Indian languages were released and around 40 submissions were made.

In Chapter 3, I describe the timeline and technical details of this challenge. I also detail my own contributions, that were completed as part of this BTP, towards the organization of this challenge.

This challenge was organized in collaboration with many researchers from different institutions, both in academia and industry. A list of the members of the organizing team can be found at `https://navana-tech.github.io/IS21SS-indicASRchallenge/team.html` and within this thesis chapter.

1.3 Bridging Scripts by Grounding in Speech

In Chapter 2, we explored transfer learning via transliteration. To do this, we assumed the existence of a transliteration system between the two languages. In this work, we remove that assumption. We explore the problem of learning a transliteration system from a language A to a language B in a weakly supervised fashion by using the audio domain as a grounding space for 'grounding' text in the two languages.

Formally, assuming we are given labelled monolingual speech data in the two languages, and that no other resource is available for the low-resource language, we devise a novel two-way NMT-like method that can produce a transliteration system from language A to language B.

In Chapter 4, we present Bridge-H2L, a method of converting labeled ASR data from a high resource language such as English to a target low resource language with limited labeled data. Bridge-H2L uses a novel two-way neural machine translation model to convert high-resource phonemes to low resource graphemes and back.

This work was done in collaboration with IBM Research.

1.4 Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages

If one has access to limited quantities of data, incorporating domain knowledge has been shown to induce useful biases, improve generalization and mitigate overfitting by learning more effectively from the limited data. Linguistic knowledge indeed provides such benefits for NLP tasks [6].

Inspired by the observation that many languages (for example, all Indian languages) have acoustically similar graphemes, we develop a two-module pipeline that first uses an ASR system to transcribe speech into a linguistically-inspired reduced vocabulary followed by a reconstructor that performs a reconstruction into the original vocabulary. This approach transfers the task of disambiguating acoustically confusable graphemes from the speech recognition step (where it is difficult) to the textual reconstruction step (where it is easier).

In Chapter 5, we explore this approach and demonstrate this technique's efficacy for two Indian languages, Gujarati and Telugu, using two types of reconstructors. 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.

An earlier version of this work was completed as an RnD Project under Prof. Preethi Jyothi and is available as a preprint [4]. Work done during the B.Tech. Project was incorporated into an updated version that was submitted to Interspeech 2021.

Chapter 2

Low Resource ASR: The surprising effectiveness of High Resource Transliteration

Cross-lingual transfer of knowledge from high-resource languages to low-resource languages is an important research problem in automatic speech recognition (ASR). We propose a new strategy of transfer learning by pretraining using large amounts of speech in the high-resource language but with its text transliterated to the target low-resource language. This simple mapping of scripts explicitly encourages increased sharing between the output spaces of both languages and is surprisingly effective even when the high-resource and low-resource languages are from unrelated language families. The utility of our proposed technique is more evident in very low-resource scenarios, where better initializations are more beneficial. We evaluate our approach using a transformer ASR architecture and the state-of-the-art wav2vec2.0 ASR architecture. We use English as the high-resource language and six languages Hindi, Gujarati, Telugu, Bengali, Amharic and Korean as low-resource targets. With access to 1 hour of target speech, we obtain relative WER reductions of up to 8.2% compared to existing transfer-learning approaches.

2.1 Introduction

End-to-end (E2E) systems have emerged as a de facto modeling choice for ASR in recent years and demonstrate superior performance compared to traditional cascaded ASR systems. However, E2E systems entail highly resource-intensive training with large amounts of labeled speech to perform well. This requirement tilts the balance in favour of high-resource languages like English for which large labeled speech corpora are publicly available. In contrast, for a majority of the world's languages, only limited amounts of transcribed speech are available. Improving the performance of E2E ASR systems for such low-resource languages by effectively making use of large amounts of labeled speech in high-resource languages is of great interest to the speech community.

Transfer learning techniques for speech recognition aim at effectively transferring knowledge from high-resource languages to low-resource languages and have been extensively studied [7]–[14]. A popular paradigm for transfer learning in E2E systems is to pretrain a model on labeled speech from one (or more) high-resource languages and then fine-tune all or parts of the model on speech from the low-resource language. Often the high-resource and low-resource languages utilize very different grapheme vocabularies. Such disparities in the output vocabularies have been predominantly handled in prior work by either training only the encoder layers or training both the encoder and decoder layers in the E2E ASR system using the high-resource language. In the latter case, the output softmax layer are on the high resource graphemes and need to be replaced with a new one corresponding to the target low-resource language before fine-tuning on speech from the low-resource language. In these approaches, the sharing across languages is latent and not controllable in the output space when the language-specific graphemes are disjoint.

In this work, we propose a strategy of increased sharing in the output grapheme space by transliteration of high resource language transcription to the low-resource language. With English as the high-resource language, we adapt to six different low-resource world languages. For these languages, off-the-shelf transliteration libraries that can convert any English text to graphemes in the languages are easily available, since transliteration is a popular input typing tool for the large number of speakers of minority languages.

We use transliteration as a first step to convert transcriptions of large English speech corpora into text in the script of the target language. The E2E model is then pretrained using these transliterated transcriptions for English speech, followed by finetuning the model using limited amounts of speech in the target language. This seemingly simple technique helps the model learn a good initialization for the target language and is shown to be more effective than standard transfer learning techniques on a range of languages. Forcing English transcriptions to adopt the same script as the target language enables better sharing of model parameters, across both encoder and decoder layers. Our method is able to provide gains even with off-the-shelf, imperfect transliteration libraries since the transliterated data is used only during pre-training. In contrast the reverse approach of transliterating low resource languages to English as proposed in [15] is significantly worse since it requires a final possibly lossy transliteration back to the native language.

2.2 Related Work

Improving low resource ASR by exploiting labeled data from high resource languages has been an active research area starting from traditional HMM-based models [7] to modern neural systems [8]–[14], [16]. While some recent systems have attempted transfer using accoustic models with a shared phone layer [11] or separate phoneme layers [10] or union of the two [17], our focus here is on the more popular end-to-end systems (E2E) that predict graphemes at the last layer. Transfer learning on E2E systems using labelled data of high-resource languages have been attempted in three settings: (1) Separate softmax layers over grapheme vocabulary of each language that are trained jointly [9], (2) Pre-training on high-resource graphemes followed by fine-tuning a separate grapheme softmax on the target low-resource language [8], [12], and (3) Training a shared softmax layer by taking a union of all grapheme vocabularies, usually applied when languages share graphemes [13]. In all these approaches the sharing across languages is latent and not controllable explicitly in the output space when the language-specific grapheme vocabularies are disjoint.

We attempt to remedy that by transliterating the high-resource graphemes (English) to the low-resource graphemes. While transliteration has been used extensively in improving machine translation, information retrieval and cross-lingual applications, little work has focused on improving speech recognition performance. Recently, [15] proposed the reverse transliteration from Indian languages to English and show improvement over a normal multilingual model. In this paper we show that our direction of transliteration from English provides much higher gains, and the reverse direction is often worse than earlier transfer learning approaches that do not attempt to share graphemes. Transliteration-based approaches are also relevant for code-switched ASR [18]. Concurrently with our work, [5] also proposed to pre-train multilingual models by transliterating low resource graphemes amongst one another. However, they obtain graphemes as predictions by an initial low resource ASR model when input high resource audio. The initial low resource ASR model is trained with its own limited data, and is likely to make highly noisy predictions. Pre-training with a model's own noisy predictions could introduce negative feedback. In contrast we propose a less error-prone method of harnessing the gold transcripts of high resource language via pre-existing transliteration libraries.

Another recent promising direction is learning transferable latent speech representations by pre-training a model from unlabelled speech [14]. Our transliteration of labelled data can be used to

further fine-tune even such pre-trained models, and we show significant gains on a recent self-supervised wav2vec2.0 [19] pre-trained model.

2.3 Proposed approach

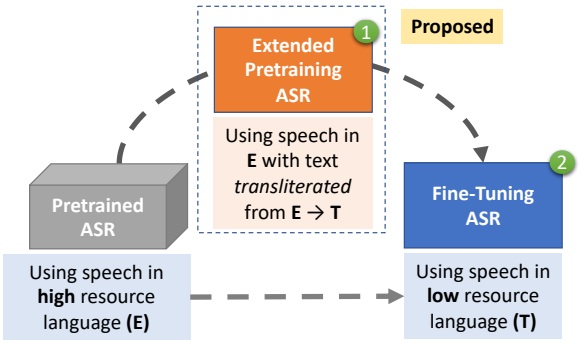

FIGURE 2.1: Training procedure for low resource languages using the proposed transliteration method

The overall training procedure of our proposed approach is shown in Figure 2.1. The training consists of two stages: pretraining followed by finetuning. During pretraining, we transliterate the high-resource language text to the target low-resource language and train the ASR model using the original audio data and this transliterated text. Next, we finetune the pretrained ASR model on the target language data, after reinitializing the output layers since the output vocabulary has changed.

en: ground without overbrimming *ipa*: ɡɹˈaʊnd wɪðˌaʊt ˌəʊvəbɹˈɪmɪŋ
hi: ग्राउंड विदऔत ओवर्ब्रिमिंग *ipa*: ɡɾaːˈʊɳd wˌɪdˈɔːt ˌoːʊrbɾɪmˈĩg
gu: ગ્રાઉન્ડ વિથાત ઓવર્બ્રીમિંગ *ipa*: ɡɾaːˈʊnd wɪtʰˈaːt ˌoːʊrbɾimˈĩg
bn: গ্রাউন্ড উইথাউত ওভারব্রিমিং *ipa*: gɾˈaunˌɔ ˈuitʰˌaut̪o ˈoːbʰaɾˌɔbɾim,iŋ
te: గ్రౌండ్ విత్‌అవుట్ ఒవెర్బ్రిమ్మింగ్ *ipa*: gɾˈuːnɖ ʋˈitaːʊʊt̪ ˈoʊʊɾbɾˌimming
ko: 그라운드 위트하우트 오버르브리밍 *ipa*: kɯɾaunda ɥitʰɯɦautʰɯ ɔbʌɾɯbɯɾimiŋ
am: ግሮውንድ ውትሀውት ኦቨር·በርምምግ *ipa*: gɨrounɨd wɨtɨhout ɔvərɨbɨrɨmɨmɨŋɨg

FIGURE 2.2: Examples of transliteration to all 6 languages for a sample English sentence.

Transliteration is utilized to convert text from one script or language to another, often preserving sounds across languages. By supervising using transliterated text during the pretraining stage, we expect that better sound to text mappings for the target language are implicitly learned. We intend to use existing transliteration tools which supports the proposed transliterations. There are two common transliteration approaches; rule-based and machine translation. The rule-based approach relies on character mappings between the two scripts whereas machine translation approaches learn from parallel training data. To show that the proposed method works with even a simple transliteration system and is therefore easily extended to several other low resource languages, we use existing simple off the shelf systems. For the four Indian languages we used `indic-trans` [20], for Korean we used the Microsoft Azure API[1], and for Amharic, we use the Google Transliterate API via the

[1]https://docs.microsoft.com/en-us/azure/cognitive-services/translator/reference/v3-0-transliterate

`google-transliterate-api`[2] pip package. In fact, developing a custom phoneme-based transliteration system did not yield any improvements over using off-the-shelf systems, so we stuck with the latter. Figure 2.2 shows examples of English text transliterated to corresponding target languages.

2.4 Experiments

We evaluate our proposed approach, **Eng2Tgt**, on six languages: Hindi, Telugu, Gujarati,Bengali, Korean, Amharic and over two model architectures (ESPNet and Wav2vec 2.0) and contrast with two existing transfer-learning approaches.[3]

2.4.1 Baselines

We devise three natural baselines that ablate the choices made in our proposed approach: 1) **NoPre.** We train the transformer ASR models from scratch on the target language datasets i.e., randomized initialization without any pretraining. 2) **EngPre.** We first pretrain the ASR model on untransliterated English i.e. using the original Latin script. Next, we finetune on the Indian language datasets. This baseline is an established method of transfer learning as discussed in Section 2.2. 3) **Tgt2Eng.** This is an approach adapted from the transliterated-based multilingual modelling for multiple languages proposed in [15]. Although they address multilingual modelling, we adapt this work to an English-centric transliteration approach to create a competitive baseline for our transliteration-based approach. We first pretrain the ASR model on English speech that uses the original Latin script. We transliterate the low-resource transcriptions to English with the help of the same transliteration modules used in our approach. Finally, we finetune on the Latin-script low-resource transcriptions. Predictions from this system will be in the Latin script. Therefore, we transliterate the hypotheses back to the low-resource language's for a fair comparison.

2.4.2 Datasets

For high-resource pretraining, we use the 100-hour Librispeech English dataset [21]. For the four Indian languages among the low-resource target languages, we used the Microsoft Speech Corpus (Indian Languages) dataset [22] for Gujarati and Telugu, the Hindi ASR Challenge dataset [23] for Hindi, and the OpenSLR Large Bengali dataset [24] for Bengali. We use the Zeroth Korean [25] dataset for Korean and the ALFFA Amharic [26] dataset for Amharic. Note that for Korean, the data setup was performed using the Zeroth-Korean Kaldi [27] recipe. This recipe uses a morphological segmentation tool called `morfessor` [28] to morphologically segment the provided text. All of the datasets mentioned above are monolingual, without any examples of code mixing with English. Table 2.1 presents detailed statistics per language. To further simulate the low resource setting we downsample the available training set to 10 hours and 1 hour.

2.4.3 Transformer-based E2E ASR

Experimental Setup. The Transformer ASR systems are built using the ESPNet Toolkit [29] that offers recipes to train hybrid CTC-attention end-to-end systems [30]. A Byte-Pair Encoding(BPE) [31] tokenization with a vocabulary size of 5000 is used to tokenize the transcriptions during pretraining and finetuning and 80-dimensional log-mel audio features (with pitch information) are used. The Transformer [32]-based hybrid CTC-attention model uses a CTC weight of 0.3 and an attention weight of 0.7 during both pretraining and finetuning. A 12-layer encoder network, and a 6-layer decoder network is used, each with 2048 units, with a 0.1 dropout rate. Each layer has eight 64-dimensional

[2]https://pypi.org/project/google-transliteration-api/

[3]For Amharic and Korean, we only report wav2vec2.0 WERs; the WERs from the Transformer model were unstable possibly due to poor seeds and require further investigation.

	Train	Dev	Test
Hindi	40.2	4.97	5.02
Telugu	30.0	2.55	2.44
Gujarati	39.7	2.00	3.15
Bengali	40.0	5.00	5.00
Korean	49.7	2.00	1.19
Amharic	18.0	2.00	0.73

TABLE 2.1: Size of the various datasets in hours.

Duration	Method	Hin	Tel	Guj	Ben
Full	NoPre	16.3	29.5	19.2	36.2
	EngPre	**15.6**	26.3	17.6	27.2
	Tgt2Eng	25.2	86.4	44.2	75.5
	Eng2Tgt (Ours)	**15.6**	**25.9**	**17**	**26.2**
10 hour	NoPre	65.5	87.1	55.2	93.4
	EngPre	29.4	51.9	**33.4**	57.1
	Tgt2Eng	40.1	91.3	55.8	85.6
	Eng2Tgt (Ours)	**28**	**48.5**	34.4	**56.4**

TABLE 2.2: Word Error Rate (WER) across different transliteration schemes on the Transformer ASR system

attention heads that are concatenated to give a 512-dimensional attention vector. During both pretraining and finetuning, models were trained for a maximum of 80 epochs with an early-stopping patience of 8 using the noam optimizer from [32] with a learning rate of 10 and 25000 warmup steps. Label smoothing and preprocessing using spectral augmentation is also used. (Note that while fine-tuning on the target language we transfer both encoder and decoder weights. This yielded better performance than transferring only the encoder weights, as observed in [33].) We average the best 5 models for decoding based on multiple criteria, i.e. validation loss, validation accuracy, or simply the last 5 models, with a beam size of 10 or 20 and a CTC weight of 0.3 or 0.5.

Results. Table 2.2 lists word error rates (WERs) on the test sets of each language for all four methods and two training durations: Full that uses the entire training set and a 10 hour subset.

From the results in Table 2.2 we make a number of observations. First, compared to the no pretraining baseline (NoPre), all three methods of harnessing the English high resource corpus provide significant gains. The gains are particularly striking for the 10 hour low resource setting. For every setting, the reverse transliteration approach (Tgt2Eng) is worse than existing fine-tuning (EngPre) that retains each language in its original grapheme space. In contrast, our approach is better than both these approaches in most cases, with larger observed gains in the low-resource 10-hours setting.

We offer an intuitive explanation for why one may expect Eng2Tgt to do better than EngPre. Unlike Eng2Pre, in Eng2Tgt the pretraining commits to target character labels appearing in transliterations of the English data. The fine-tuning phase then only needs to focus on learning new sounds that are missing in the English speech data. While simplistic, this explanation is somewhat borne out by the CER statistics. Of the top five characters in which Eng2Tgt gains the largest CER reductions over EngPre, most of them have very low frequencies of less than 0.5% in the transliterated corpora.

Some examples of erroneous transliterations for the Indian languages are shown in Figure 2.3; when a word is first transliterated into Latin script, it gets transliterated back into a different word in the target script. These errors occur because multiple Indian words can transliterate to the same

चीफ -> chif -> चिफ निर्णयों -> nirnyon -> निर्नयोन
આર્થિક -> aarthik -> આર્તિક પીપોદર -> pepodar -> પેપોદર

FIGURE 2.3: Erroneous transliterations

	Method	Hin	Tel	Guj	Ben	Kor	Amh
10	SelfSup	23.8	35.7	25.2	29.4	21.79 (14.3)	**26.54**
	EngPre	24.0	37.6	25.0	32.3	**13.16 (9.4)**	26.78
	Ours	**23.6**	**34.5**	**23.2**	**28.2**	**13.16** (9.6)	27.32
1	SelfSup	28.9	42.1	57.1	**83.1**	99.87 (83.3)	**52.30**
	EngPre	29.9	48.1	62.1	92.3	66.36 (40.8)	53.75
	Ours	**28.5**	**41.5**	**55.2**	88.9	**62.08 (37.2)**	53.29

TABLE 2.3: WERs on the wav2vec2.0 ASR system with 10 and 1 hour of target data. For Korean, CERs are in parentheses.

English word; many orthographic differences in these Indian languages (e.g., long and short vowels, aspirated and unaspirated consonants, etc.) are lost when transliterated to English. The superior performance of our Eng2Tgt approach demonstrates that even with imperfect transliteration systems, it is possible to design effective transfer strategies. Our approach relies on the transliterated text only during the pre-training step, and the final finetuning with the target language is able to unlearn errors induced due to the imperfect transliteration. On the other hand, Tgt2Eng produces predictions in the Latin script, where the final transliteration back to the Indian language induces large errors in the output transcripts that remain uncorrected.

2.4.4 wav2vec2.0-based ASR

Experimental Setup. We show the promise of our proposed transliteration-based pretraining on the recently introduced self-supervised encoder architecture wav2vec2.0 [19]. We use the pretrained wav2vec2.0 model estimated using unsupervised pretraining on the complete Librispeech dataset and refer to this system as SelfSup. With SelfSup as our starting point, we report numbers for both EngPre (involving supervised pretraining with 100 hours of English speech) and Eng2Tgt that applies transliteration to the 100 hours of English data before pretraining. (Tgt2Eng was excluded due to its poor performance on the Transformer ASR experiments.) For finetuning on our datasets, we use the same training recipe as described in [19] for the 10 hr and 1 hr settings. We perform inference using the default hyperparameters as described in [19] with a 4-gram KenLM[34] for all languages. Since we start with a powerful pretrained model, we show results in Table 2.3 for very low-resource scenarios i.e. using 10-hour and 1-hour training subsets.

Results. SelfSup offers a much stronger baseline compared to NoPre in Table 2.2. We observe that our proposed pre-training with transliterated data provides gains even on a system like wav2vec that leverages powerful pretrained models. Our method also outperforms EngPre in most settings. For most languages in both settings, the EngPre baseline is worse than even the SelfSup baseline, despite being pretrained on more (Latin-script English) data. A major exception is Amharic. We will investigate reasons for this difference in Section 2.4.5.

Language	am	bn	hi	te	gu
Transliteration PER	89	90	76	82	72
KL dist phones	8.2	13.6	10.2	11.4	15.6

TABLE 2.4: Second row shows acoustic consistency of transliteration measured as the phone error rate between transliterated and English text. Third row shows the KL divergence between the unigram phone distribution of English and native language.

	Full	10 Hours
Hin2Tgt	**18.3**	**35.4**
Eng2Tgt40	21.3	38.1

TABLE 2.5: WERs for Gujarati comparing Eng2Tgt pretraining using 40 hrs of transliterated Hindi vs. transliterated English.

2.4.5 Analysis and discussions

Here we attempt to understand the conditions under which our proposed approach is likely to help. Two properties need to *simultaneously* hold for pretraining with transliterated English to be useful: (1) The transliteration library needs to be acoustically consistent, and (2) The overlap in the sounds (phones) of the high and low-resource language has to be high, that is the phonological distance between the two languages should be low. We analyze how well these two properties hold.

Acoustic consistency of transliterations We measure acoustic consistency of the various transliteration libraries using the following PER (phone error rate)-based method. First we convert the Latin-script Librispeech English training set text to IPA using an English g2p tool, `epitran` [35] and also convert the transliterated English to IPA using native-language g2p tools. We use `epitran` [35] for Hindi, Telugu, Bengali and Amharic and UIUC's `g2ps`[4]. We could not find a reliable Korean g2p tool. We then compute the PER between these two IPA sequences ignoring space tokens. Table 2.4 shows the PER of various languages. We observe that Amharic has a relatively high error compared to languages such as Hindi and Gujrati.

Phonological similarity of low resource language with English Many different methods have been proposed for measuring the phonological similarity of two languages [36]. Here we resort to a comparison of a simple unigram distribution of the phones in English and native language. The third row of Table 2.4 shows the results. We observe that for languages like Hindi and Telugu where the KL distance in phone distribution is small *and* the transliteration PER is low, we get consistent gains across different architectures and training data sizes. This analysis provides a partial explanation of our observed numbers but for languages like Amharic more investigation is required.

Effect of related languages Table 2.5 shows WERs for Gujarati in the full and 10 hour training settings on the Transformer system. Using our approach, we compare performance when using a language related to the target language (e.g. Hindi) (Hin2Tgt) during pretraining as opposed to English. The transliterated English data during pretraining was reduced to 40 hours to match the amount of transliterated Hindi pretraining data (Eng2Tgt40). We observe there's a significant improvement in performance across both training settings with using Hindi instead of English during pretraining. Using both transliterated English and Hindi data during pretraining for the 10-hour Gujarati task further reduces WERs from 55.8% to 32.4%. This opens up interesting directions to explore as future work.

[4] https://github.com/uiuc-sst/g2ps

2.5 Future Work

Future work will explore ideas such as:

- Using this transliteration idea for multilingual ASR. More specifically, given N languages, transliterating the ground truth texts of each language to every other language, producing an $N \times N$ data augmentation that can be used to train the multilingual network.
- Using multilingual systems that have language independent and shared layers. Weights for giving higher priority to related languages can be automatically learned.
- Extending this idea to languages with no transliteration systems. This can be done by using intermediate phone layers. One can build a pipeline consisting of a universal phone layer (similar to [37]) followed by language specific phone to grapheme layers. This can be trained if pronunciation dictionaries/Romanizations for the low-resource language are available. If the phone layer is sufficiently well-trained, only labelled speech data for the low-resource language is needed.

2.6 Conclusion

This work explores the surprisingly effective role of transliteration in training an E2E ASR system. We propose a simple transliteration-based transfer learning technique easily adaptable to other low-resource languages and demonstrate the utility of our proposed approach on two state-of-the-art ASR systems, showing significant improvements in performance over established transfer-learning approaches despite using an imperfect transliteration system.

2.7 Collaborators

Researchers who collaborated on this project:

Shreya Khare[†,1], Ashish Mittal[†,1,2], **Anuj Diwan**[†,2], Sunita Sarawagi[2], Preethi Jyothi[2], Samarth Bharadwaj[1]

where † contributed equally.

Affiliations:

[1] : IBM Research ; [2] : IIT Bombay

Chapter 3

Multilingual & Code-switching ASR Challenges for Low Resource Indian Languages

I am one of the organizers and one of the primary technical contributors for the 'Multilingual and Code-switching ASR Challenges for Low Resource Indian languages' Special Session at Interspeech 2021. The organizational and technical work for this Special Session was carried out as part of this BTP. In this chapter, I first provide a brief description of the Special Session and its timeline. Then, I present the technical paper that describes the technical details of the datasets released as part of this Special Session, the baseline descriptions, and the inference pipeline descriptions. Finally, I provide details regarding some of my major duties and technical contributions.

3.1 Brief Description and Timeline

The 'Multilingual and Code-switching ASR Challenges for Low Resource Indian Languages' Special Session at Interspeech 2021 is an Automatic Speech Recognition challenge for low-resource languages and code-switching. The official website for the challenge is `https://navana-tech.github.io/IS21SS-indicASRchallenge/`. It consists of two subtasks:

- **Subtask 1**: This subtask involves building a *multilingual ASR system* in six Indian languages: Hindi, Marathi, Odia, Telugu, Tamil, and Gujarati using training data released as part of this challenge. The blind test set consist of recordings from a subset (or all) of these six languages, with no labelled language IDs. Thus, a multilingual ASR system that can work well on all of these languages will have good performance. More details are available in the following sections.
- **Subtask 2**: This sub-task involves building a code-switching ASR system separately for Hindi-English and Bengali-English code-switched pairs. The blind test set will comprise recordings from these two code-switched language pairs. This code-switching data is novel in the sense that it is drawn from technical lectures and hence contains technical words, very natural code-switching and interesting subchallenges (such as handling spoken punctuation). I worked extensively on this subtask.

Around 40 teams participated in the two subtasks and competed fiercely to obtain the best WERs until the challenge's conclusion. The final leaderboard is available on the challenge website.

The challenge proceeded as follows:

1. *Dec 2020 - Jan 10, 2021* : Conceptualizing the challenge and figuring out the datasets to be used for the code-switching Subtask 2. The data for subtask 2 was obtained, with appropriate permissions, from technical lectures delivered as part of the Spoken Tutorial program at IIT Bombay (`https://spoken-tutorial.org/`). This data was cleaned and converted to an appropriate audio format.

2. *Jan 10, 2021 - Jan 30, 2021* : Since the audio transcription was at the granularity of a lecture, the transcription was aligned to the audio and thereafter spliced into segments of smaller lengths (around 15-20 seconds). Because the data is found data, numerous artefacts exist, such as language mixing, incomplete audio, etc. (More details in the following sections). These artefacts were identified and catalogued. The training and validation data contains these artefacts, since an important part of the challenge is being able to train a system that is robust to such data errors.

3. *Jan 30, 2021 - Feb 10, 2021* : Registration procedure, FAQs, and challenge rules were discussed amongst the organizers, finalized, and released on the challenge website. The final train, dev and test set data was released on the website as well as on OpenSLR (`http://openslr.org/103/` and `http://openslr.org/104/`).

4. *Feb 11, 2021 - Mar 2, 2021* : Baseline recipes for both subtasks were built. For subtask 1, a Hybrid Kaldi-based system [27] was built. For subtask 2, two baselines were built: a Hybrid Kaldi-based system and an End-to-end Espnet-based system [29]. These baselines were released as a public GitHub repository (`https://github.com/navana-tech/baseline_recipe_is21s_indic_asr_challenge`) with detailed READMEs on how to use the published code.

5. *Mar 2, 2021 - Mar 16, 2021* : Blind test data was prepared for both subtask 1 and subtask 2. For subtask 1, some blind test data was collected from native speakers. For subtask 2, the blind test data was cleaned of all artefacts. It was first re-transcribed by native speakers. Thereafter, a colleague and I manually traversed the entire dataset and corrected various errors, details of which are provided in the following sections. A new evaluation pipeline involving transliterated WERs was also devised for subtask2.

6. *Mar 17, 2021 - Apr 2, 2021* : An automatic submission pipeline was setup and blind test data was released to participants. The leaderboard was auto-populated with submissions. Finally, a technical paper describing the entire challenge was written, released on arXiV (`https://arxiv.org/abs/2104.00235`) and submitted to Interspeech 2021.

3.2 Technical Abstract

Recently, there is an increasing interest in multilingual automatic speech recognition (ASR) where a speech recognition system caters to multiple low resource languages by taking advantage of low amounts of labelled corpora in multiple languages. With multilingualism becoming common in today's world, there has been increasing interest in code-switching ASR as well. In code-switching, multiple languages are freely interchanged within a single sentence or between sentences. The success of low-resource multilingual and code-switching ASR often depends on the variety of languages in terms of their acoustics, linguistic characteristics as well as the amount of data available and how these are carefully considered in building the ASR system. In this challenge, we would like to focus on building multilingual and code-switching ASR systems through two different subtasks related to a total of seven Indian languages, namely Hindi, Marathi, Odia, Tamil, Telugu, Gujarati and Bengali. For this purpose, we provide a total of ∼600 hours of transcribed speech data, comprising train and test sets, in these languages, including two code-switched language pairs, Hindi-English and Bengali-English. We also provide baseline recipes[1] for both the subtasks with 30.73% and 32.45% word error rate on the multilingual and code-switching test sets, respectively.

[1] https://github.com/navana-tech/baseline_recipe_is21s_indic_asr_challenge

3.3 Introduction

India is a country of language continuum, where every few kilometres, the dialect/language changes [38]. Various language families or genealogical types have been reported, in which the vast number of Indian languages can be classified, including Austro-Asiatic, Dravidian, Indo-Aryan, Tibeto-Burman and more recently, Tai-Kadai and Great Andamanese [39], [40]. However, there are no boundaries among these language families; rather, languages across different language families share linguistic traits, including retroflex sounds, absence of prepositions and many more resulting in acoustic and linguistic richness. According to the 2001 census, 29 Indian languages have more than a million speakers. Among these, 22 languages have been given the official language status by the Government of India [41], [42]. Most of these languages are low resource and do not have a written script. Hence, speech technology solutions, such as automatic speech recognition (ASR), would greatly benefit such communities [43]. Another common linguistic phenomenon in multilingual societies is code-switching [44], typically between an Indian language and (Indian) English. Understanding code-switching patterns in different languages and developing accurate code-switching ASR remain a challenge due to the lack of large code-switched corpora [45], [46].

In such resource-constrained settings, exploiting unique properties and similarities among the Indian languages could help build multilingual and code-switching ASR systems. Prior works have shown that multilingual ASR systems that leverage data from multiple languages could explore common acoustic properties across similar phonemes or graphemes [13], [43], [47]–[49]. This is achieved by gathering a large amount of data from multiple low-resource languages. Also, multilingual ASR strategies are effective in exploiting the code-switching phenomena in the speech of the source languages [50]. However, there is an emphasis on the need for the right choice of the languages for better performance [51], as significant variations between the languages could degrade the ASR performance under multilingual scenarios [13]. In such cases, a dedicated monolingual ASR could perform better even with lesser speech data than a multilingual [49], [52], [53] or code-switching ASR.

TABLE 3.1: Details of multilingual ASR train (Trn), test (Tst) and blind test (Blnd) data – size, channel compression (Ch.comp), number of unique sentences (Uniq sent), number of speakers (Spkrs) and vocabulary size in words (vocab). All six languages' audio files are single-channel and encoded in 16-bit with a sampling rate of 8kHz except for train and test set of Telugu, Tamil and Gujarati, at 16kHz.

	Hindi			Marathi			Odia			Telugu			Tamil			Gujarati		
	Trn	Tst	Blnd	Trn	Tst	Blnd	Trn	Tst	Blnd	Trn	Tst	Blnd	Trn	Tst	Blnd	Trn	Tst	Blnd
Size (hrs)	95.05	5.55	5.49	93.89	5	0.67	94.54	5.49	4.66	40	5	4.39	40	5	4.41	40	5	5.26
Ch.comp	3GP	3GP	3GP	3GP	3GP	M4A	M4A	M4A	M4A	PCM	PCM	PCM	PCM	PCM	PCM	PCM	PCM	PCM
Uniq sent	4506	386	316	2543	200	120	820	65	124	34176	2997	2506	30329	3060	2584	20257	3069	3419
Spkrs	59	19	18	31	31	-	-	-	-	464	129	129	448	118	118	94	15	18
Vocab (words)	6092	1681	1359	3245	547	350	1584	334	334	43270	10859	9602	50124	12279	10732	39428	10482	11424

Considering the factors above, in this challenge, we have selected six Indian languages, namely, Hindi, Marathi, Odia, Telugu, Tamil and Gujarati, for multilingual ASR; and two code-switched language pairs, Hindi-English and Bengali-English, for code-switching ASR. Unlike prior works on multilingual ASR, the languages selected 1) consider the influences of three major language families – Indo-Aryan, Dravidian and Austro-Asiatic, which influences most of the Indian languages [41], 2) cover four demographic regions of India – East, West, South and North, and, 3) ensure continuum across languages. It is expected that a multilingual ASR built on these languages could be helpful to extend to other low-resource languages [43]. Further, most of the multilingual ASR works in the literature have considered languages other than Indian languages. Works that consider the Indian languages, however, use data that is either not publicly available or limited in size [42], [43], [52], [54], [55]. This is similarly true for code-switched speech, and prior work has predominantly focused on

Hindi-English speech [56]–[60]. This challenge significantly contributes in this context, as we provide a larger corpus (∼600 hours of transcribed speech from different domains) compared to the existing publicly available data for Indian languages.

The challenge comprises two subtasks. *Subtask1* involves building a multilingual ASR system in six languages: Hindi, Marathi, Odia, Telugu, Tamil, and Gujarati. The blind test set comprises recordings from all the six languages. *Subtask2* involves building a code-switching ASR system separately for Hindi-English and Bengali-English code-switched pairs. The blind test set comprises recordings from these two code-switched language pairs. Baseline systems are developed considering hybrid DNN-HMM models for both the subtasks and an end-to-end model for *Subtask2*. Baseline word error rates (WERs) averaged over languages on the test set and blind test set are found to be 30.73% & 32.73%, respectively, for *subtask1*. Similarly, WERs, averaged between two code-switching language pairs, for *Subtask2* are 33.35% & 28.52, 29.37% & 32.09% and 28.45% & 34.08% on test & blind sets with GMM-HMM, TDNN and end-to-end systems, respectively.

3.4 Details of the two Subtasks

3.4.1 Subtask1: Multilingual ASR

Subtask1 is for developing robust multilingual systems in six Indian languages using ∼450 hours of data released as a part of the challenge.

Dataset Description

Table 3.3 shows the data details for the Multilingual ASR specific to each language. The audio files of Odia are collected in the domains of agriculture, healthcare and finance from four districts as a representative of four different dialect regions – Sambalpur (North-Western Odia), Mayurbhanj (North Eastern Odia), Puri(Central and Standard Odia) and Koraput (Southern Odia). The Telugu, Tamil and Gujarati data are taken from Interspeech 2018 low resource ASR challenge for Indian languages, for which, the data was provided by SpeechOcean.com and Microsoft [61]. Further, in all the six languages, the percentage of out-of-vocabulary (OOV) between train & test and train & blind test are found to be in the range 17.2% to 32.8% and 8.4% to 31.1%, respectively. Also, the grapheme set in the data follows the Indian language speech sound label set (ILSL12) standard [62]. The total number of graphemes are 69, 61, 68, 64, 50 and 65 respectively for Hindi, Marathi, Odia, Telugu, Tamil and Gujarati, out of which a total number of diacritic marks in the respective languages are 16, 16, 16, 17, 7 and 17.

Characteristics of the dataset

The Hindi, Marathi and Odia data are collected from the respective native speakers using a reading task. For the data collection, the speakers and the text are selected to cover different language variations for better generalizability. The speakers of Hindi and Marathi belong to a high-literacy group. On the other hand, the speakers of Odia belong to a semi-literate group. The text data of Hindi and Marathi is collected from storybooks. In addition to the speaker and text variability, nativity, pronunciations and accent variations are also present in the datasets. The Odia data went through a manual check, while Hindi and Marathi speech data were passed through a semi-automatic validation process using the ASR pipeline (For more details, please refer to Section 1 of the supplementary material). The train and test sets of Telugu, Tamil and Gujarati are considered as-is for this challenge; however, for the blind test set, the measurement set is used and part of it is modified with speed perturbations randomly between 1.1 to 1.4 (with increments of 0.05), and/or adding one noise randomly from white, babble, and three noises chosen from the Musan dataset [63] at various the signal-to-noise

TABLE 3.2: Details of code-switching ASR train (Trn), test (Tst) and blind test (Blnd) data – size, uniq sent, spkrs and vocab.

	Hin-Eng			Ben-Eng		
	Trn	Tst	Blnd	Trn	Tst	Blnd
Size (hrs)	89.86	5.18	6.24	46.11	7.02	5.53
Uniq sent	44249	2890	3831	22386	3968	2936
Spkrs	520	30	35	267	40	32
Vocab (words)	17830	3212	3527	13645	4500	3742

ratio randomly selected from a set between 18dB to 30dB at the step of 1dB. This modification is done randomly on 29.0%, 23.8% and 34.1% of Telugu, Tamil and Gujarati measurement data, respectively.

Evaluation criteria

From the channel compression schemes in Table 3.3, it is observed that there is a mismatch in the channel compression between train/test and blind test for Marathi. Thus, for the evaluation on the blind test, we consider both the channel matched and mismatched scenarios, for which WER across languages within each scenario is calculated. Thus, we get two WERs: 1) averaged WER across all six languages (channel mismatched scenario), 2) averaged WER across all six languages except Marathi (channel matched scenario).

3.4.2 Subtask2: Code-switching ASR

Subtask2 is on developing code-switching ASR on the Hindi-English and Bengali-English language pairs taken from spoken tutorials.

Dataset Description

The tutorials in the *Subtask2* data cover a range of technical topics, and the code-switching predominantly arises from the technical content of the lectures. The segments file in the baseline recipe provides sentence time-stamps. These time-stamps were used to derive segments from the audio file to be aligned with the transcripts given in the text file. Table 3.2 shows the details of the data considered for *Subtask2*. All the audio files in both datasets are sampled at 16 kHz, 16 bits encoding. The test-train sentence overlap in Hindi-English and Bengali-English data are 33.9% and 10.8%, whereas the blind test-train sentence overlaps are 2.1% and 2.9%, respectively. Speaker information for both these datasets was not available. However, we do have information about the underlying tutorials from which each sentence is derived. We assumed that each tutorial comes from a different speaker; these are the numbers reported in Table 3.2. The percentage of OOV words encountered in the test and blind-test for Hindi-English is 12.5% & 19.6% and for Bengali-English is 22.9% & 27.3% respectively.

Characteristics and Artefacts in the Dataset

The transcriptions in the *Subtask2* data include mathematical symbols and other technical content. It is to be noted here that these tutorials were not explicitly created for ASR but end-user consumption as videos of tutorials in various Indian languages; specifically, in our case, the transcriptions were scripts for video narrators. Thus, there are the following sources of noise in the transcriptions – 1) misalignments between transcription and its respective segment start and end times, 2) inconsistencies in the transcriptions' language for the same audio segment, 3) punctuation's enunciation in the

speech, 4) language mixing within a word, 5) incomplete audio at the begin or the end of an utterance, and 6) merged English words without word boundary markings (For more details, please refer to Section 2 of the supplementary material).

Evaluation criteria:

To handle the transcriptions' language's inconsistencies during the evaluation, we consider transliterated WER (T-WER) besides the standard WER. To ensure that remaining noises are eliminated, we perform manual validation on the blind test set data. While the standard WER only counts an ASR hypothesis as correct if it is an exact match with the word in the reference text, T-WER counts an English word in the reference text as correctly predicted if it is in English transliterated form in the native script. To compute T-WER, we manually annotate the blind test reference text such that every English word only appeared in the Latin script. Following this, we transliterate every English word in the reference transcriptions using Google's transliteration API and manually edit them to remove valid Hindi words and fix any transliteration errors. This yielded a list of English to native script mappings and used this mapping file in the final T-WER to map English words to their transliterated forms.

3.5 Details of baseline schemes

3.5.1 Experimental setup

Multilingual ASR

Hybrid DNN-HMM: ASR model is built using the Kaldi toolkit with a sequence-trained time-delay neural network (TDNN) architecture using the lattice-free MMI objective function [64]. We consider an architecture comprising 6 TDNN blocks with a dimensionality of size 512.
Lexicon: A single lexicon is used containing the combined vocabulary of all six languages. For each language, the lexicon's entries are obtained automatically, considering a rule-based system that maps graphemes to phonemes. For the mapping, we consider the Indian speech sound label set (ILSL2) [62].
Language model (LM): A single LM is built considering the text transcriptions belonging to the train set from all six languages. For the LM, we consider a 3-gram language model developed in Kaldi using the IRSTLM toolkit. Since the LM has paths that contain multiple languages, the decoded output could result in code-mixing across the six languages.

In addition to the multilingual ASR, we also provide monolingual ASR systems' performance considering language-specific training data, lexicon and LM built with language-specific train text transcriptions.

Code-switching ASR

Hybrid DNN-HMM: The ASR model is built using the Kaldi toolkit with the same model architecture for both Hindi-English and Bengali-English language pairs. We use MFCC acoustic features to build speaker-adapted GMM-HMM models. Similar to *Subtask1*, we also build hybrid DNN-HMM ASR systems using TDNNs comprising 8 TDNN blocks with dimension 768.
End-to-end ASR: The hybrid CTC-attention model based on Transformer [65] is used with a CTC weight of 0.3 and an attention weight of 0.7. A 12-layer encoder network and a 6-layer decoder network is used, each with 2048 units, with a 0.1 dropout rate. Each layer contains eight 64-dimensional attention heads, which are concatenated to form a 512-dimensional attention vector. Models are trained for a maximum of 40 epochs with early-stopping patience of 3 using the Noam optimizer from [65] with a learning rate of 10 and 25000 warmup steps. Label smoothing and preprocessing using spectral augmentation is also used. The top 5 models with the best validation accuracy are

averaged, and this averaged checkpoint is used for decoding. Decoding is performed with a beam size of 10 and a CTC weight of 0.4.
Lexicon: Two different lexicons are used, each for Hindi-English and Bengali-English language pair. For each lexicon, the pronunciations are generated as follows for the entire vocabulary in the respective training set. If the word is in the Devanagari/Bengali script, we consider the respective pronunciation as the word's character sequence. This is because both languages have phonetic orthographies. To obtain pronunciations for English words, we use an open-source g2p package (`https://github.com/Kyubyong/g2p`). This package provides pronunciations for numericals, retrieves pronunciations from CMUDict dictionary [66] for words that appear in its vocabulary and predict new pronunciations for words that do not. We also obtain pronunciations for the punctuations by mapping to their corresponding English words.
Language model: Two separate language models are built for each language pair. We consider a trigram language model with Kneser-Ney discounting for each LM training using the SRILM toolkit developed in Kaldi [67].

3.5.2 Baseline results

Multilingual ASR

Table 3.3 shows the WERs obtained on test and blind test sets for each of the six languages along with averaged WER across all six languages. The table shows that the WER obtained with multilingual ASR is lower for Tamil. Though the WER from the multilingual ASR system is higher in the remaining languages compared to their monolingual counterpart, it does not require any explicit language identification (LID) system. Further, it is known that multilingual ASR is effective in obtaining a better acoustic model (AM) by exploring common properties among the multiple languages. However, the multilingual ASR performance also depends on the quality of the language model, which, in this work, could introduce noise due to code-mixing of words. In order to know these variabilities, we analyse the multilingual ASR considering the code-mix in the decoded output and the AM likelihoods separately.

TABLE 3.3: Performance (WER in %) of multilingual and monolingual ASRs on test (Tst) and blind (Blnd) test. Averaged WER across five languages on the blind test for Multi and Mono are 33.47% and 29.98% respectively.

		Hindi	Marathi	Odia	Tamil	Telugu	Gujarati	Avg
Multi	Tst	40.41	22.44	39.06	33.35	30.62	19.27	30.73
	Blnd	37.20	29.04	38.46	34.09	31.44	26.15	32.73
Mono	Tst	31.39	18.61	35.36	34.78	28.71	18.23	27.85
	Blnd	27.45	20.41	31.28	35.82	29.35	25.98	28.38

TABLE 3.4: Averaged languages' (in column) code-mix word percentage in the decoded output from the multilingual ASR for the utterances belonging to a language (in row).

	Hindi	Marathi	Odia	Tamil	Telugu	Gujarati
Hindi	82.3	0.4	0.2	0.8	1.9	14.4
Marathi	16.7	71.8	2.8	0.6	2.0	8.8
Odia	0.4	0.1	96.1	0.7	1.3	1.5
Tamil	0.1	0.0	0.0	98.4	1.0	0.4
Telugu	0.2	0.1	0.0	0.7	97.9	1.2
Gujarati	0.3	0.1	0.0	1.0	0.8	97.7

Analysis: Table 3.4 shows the amount of code-mix across the languages by averaging the percentage of words per sentence of a language in the column in the decoded output of the utterances belonging to the language in the row. The higher values in diagonal entries in the table indicate the multilingual ASR's effectiveness in decoding the target language's utterance. However, the off-diagonal

TABLE 3.5: AM log-likelihoods on test sets from monolingual and multilingual ASRs using forced-alignment.

	Hindi	Marathi	Odia	Tamil	Telugu	Gujarati
Multi	2.5 (0.3)	2.5 (0.3)	2.6 (0.3)	2.3 (0.2)	2.2 (0.2)	2.0 (0.2)
Mono	1.9 (0.2)	1.7 (0.2)	2.4 (0.3)	2.2 (0.2)	2.1 (0.2)	1.8 (0.1)

values of averaged percentage of words are also significant, which could be cause for higher WER with the multilingual ASR system compared to the monolingual ASR systems. Further, to know the effectiveness of AM only, we compute the AM likelihoods considering the forced-alignment process with multilingual and monolingual ASR models. These are shown in Table 3.5. The higher likelihoods with multilingual ASR indicate its benefit over monolingual AM. Thus, the multilingual ASR performance could improve with an effective LM.

Code-switching ASR

Table 3.6 shows WERs for both the Hindi-English and Bengali-English datasets. As mentioned in Section 3.4.2, there are misalignments between the transcriptions and the timestamps in some of the training files. We present results using the original alignments that we obtained with the transcriptions (labelled as UnA). In an attempt to fix the misalignment issues, we also force-align the training files at the level of the entire tutorial with its complete transcription and recompute the segment timestamps. We retrain our systems using these realigned training files (labelled as ReA). As expected, we observe that the averaged ReA WERs are consistently better than the UnA WERs. While the Kaldi TDNN-based system gives better WERs for the test set, the speaker adapted triphone GMM-HMM model performs the best on the blind test set.

Table 3.7 shows the corresponding WERs and T-WERs for the realigned (ReA) blind test sets. T-WER, being a more relaxed evaluation metric, is always better than WER. The Hindi-English code mixed data yields improved WERs evidently due to fewer OOVs and larger amounts of training data. The corresponding values for the blind-set also improve further as we calculate the transliterated scores as discussed in 3.4.2.

TABLE 3.6: WERs from GMM-HMM, DNN-HMM and end-to-end ASR systems for Hin-Eng and Ben-Eng test (Tst) and blind-test (Blnd) sets. (ReA) and (UnA) refers to re-aligned and unaligned audio files, respectively.

	Kaldi-Based				End-to-End	
	GMM-HMM		TDNN		Transformer	
	Tst	Blnd	Tst	Blnd	Tst	Blnd
Hin-Eng (UnA)	44.30	25.53	36.94	28.90	27.7	33.65
Ben-Eng (UnA)	39.19	32.81	34.31	35.52	37.2	43.94
Avg (UnA)	41.75	29.17	35.63	32.21	32.45	38.80
Hin-Eng (ReA)	31.56	24.66	28.40	29.03	25.9	31.19
Ben-Eng (ReA)	35.14	32.39	30.34	35.15	31.0	36.97
Avg (ReA)	33.35	28.52	29.37	32.09	28.45	34.08

Analysis: Table 3.8 shows the relative errors of English words in the reference transcripts either being substituted or deleted with respect to the total errors. The Hindi-English and Bengali-English blind test transcriptions contain 34.2% and 33.6% English words, respectively. The relative errors in Table 3.8 (all greater than 50%) show that the errors on English words are relatively more compared to Bengali/Hindi words. While the Kaldi-based GMM-HMM (tri4b) models give the best WERs on the blind test sets in Table 3.7, it has the highest relative error rates compared to the end-to-end and TDNN architectures as shown in Table 3.8.

TABLE 3.7: WER and T-WER values for Kaldi and end-to-end based architectures obtained after using aligned (ReA) segments for both Hin-Eng and Ben-Eng.

	Kaldi-Based				End-to-End	
	GMM-HMM		TDNN		Transformer	
	WER	T-WER	WER	T-WER	WER	T-WER
Hin-Eng	24.66	22.72	29.03	26.20	31.19	29.80
Ben-Eng	32.39	31.42	35.15	33.39	36.97	36.00
Avg	28.52	27.07	32.09	29.79	34.08	32.9

TABLE 3.8: Relative substitution/deletion error rates (Err) and transliterated error rates (T-Err) of English words for Kaldi and end-to-end based architectures for Hin-Eng and Ben-Eng.

	Kaldi-Based				End-to-End	
	GMM-HMM		TDNN		Transformer	
	Err	T-Err	Err	T-Err	Err	T-Err
Hin-Eng	62.23	59.13	60.41	56.43	57.96	56.32
Ben-Eng	58.44	56.72	55.16	52.77	56.29	54.97

3.6 Conclusion

This paper presents the dataset details and baseline recipe and results for Multilingual and code-switching ASR challenges for low resource Indian languages as a special session in Interspeech 2021. This challenge involves two subtasks dealing with 1) multilingual ASR and 2) code-switching ASR. Through this challenge, the participants have the opportunity to address two critical challenges specific to multilingual societies, particularly in the Indian context – data scarcity and the code-switching phenomena. Through this challenge, we also provide a total of ∼600 hours of transcribed speech data, which is a reasonably large corpus for six different Indian languages (especially when compared to the existing publicly available datasets for Indian languages). Baseline ASR systems have been developed using hybrid DNN-HMM and end-to-end models. Furthermore, carefully curated held-out blind test sets are also released to evaluate the participating teams' performance.

3.7 Supplementary Material

3.7.1 Semi-automatic validation of Hindi and Marathi data in Subtask1

The automatic validation is done separately for Hindi and Marathi considering the following two measures – 1) WER obtained from ASR [31] and 2) likelihood scores obtained from decoded lattice from ASR. In both cases, ASR is trained separately for Hindi and Marathi, considering noisy data from each language (before validation) of ∼500hrs. The WER based data validation has been used in prior work [68]. We believe that the WER-based criteria discard the audios containing insertions, deletions, and/or substitution errors while reading the stimuli. On the other hand, unlike prior works, we include lattice-based likelihood criteria for discarding very noisy audios or the audios with many incorrect pronunciations. In this work, we consider all those audios whose lattice likelihoods are above 3.49 and WER is equal to 0.0% for Hindi. Similarly, those audios were chosen for the Marathi data for which the WER is less than or equal to 12.5% and lattice likelihood is greater than 3.23. The choice of threshold is found to achieve ∼ 100hrs of data separately for Hindi and Marathi by ensuring higher likelihood and lower WERs in the selected audios. Further, the selected data is split into a train (Trn) and test (Tst) without sentence overlap and with out-of-vocabulary (OOV) rates at about 30% between train and test sets.

3.7.2 Sources of noise in the transcriptions of the Subtask2 data:

Misalignments: Each spoken tutorial came with transcriptions in the form of subtitles with corresponding timestamps. These timestamps were more aligned with how the tutorial videos proceeded rather than with the underlying speech signal. This led to misalignments between the transcription and segment start and end times specified for each transcription. While this is an issue for training and development segments, transcriptions corresponding to the blind test segments were manually edited to exactly match the underlying speech signal and fix any transcription errors.

Inconsistent script usage: There were multiple instances of the same English word appearing both in the Latin script and the native scripts of Hindi and Bengali in the training data. Given this inconsistency in script usage for English words, the ASR predictions of English words could either be in the native script or in the Latin script. To allow for both English words and their transliterations in the respective native scripts to be counted as correct during the final word error rate computations, we introduce a transliterated WER (T-WER) metric along with the standard WER metric.

Punctuations: Since our data is sourced from spoken tutorials on coding and computer science topics, there are many punctuations (e.g., semicolon, etc.) that are enunciated in the speech. This is quite unique to this dataset. There were also many instances of enunciated punctuations not occurring in the text as symbols but rather as words (e.g., 'slash' instead of '/', etc.). There were many non-enunciated punctuations in the text as well (exclamation, comma, etc.). We manually edited the blind test transcriptions to remove non-enunciated punctuations and normalized punctuations written as words into their respective symbols (e.g. + instead of plus).

Mixed words: In the Bengali-English transcriptions, we saw multiple occurrences of mixed Bengali-English words (which was unique to Bengali-English and did not show up in Hindi-English). To avoid any ambiguities with evaluating such words, we transliterated these mixed words entirely into Bengali.

Incomplete audio: Since the lecture audios are spliced into individual segments, sometimes a few of the segments have incomplete audio either at the start or at the end of the utterances. For the blind test audio files, which went through a careful manual annotation, such words were only transcribed if the word was mostly clear to the annotator. Else, it was omitted from the transcription.

Merged English words: Since the code-switching arises mostly due to technical content in the audio, there are instances of English words being merged together in the transcriptions without any word boundary markers. For example, words denoting websites and function calls generally have two or more words used together without spaces. For the blind test transcriptions, we segregated such merged English words into their constituents to avoid ambiguous occurrences of both merged and individual words arising together.

3.8 My Major Duties and Technical Contributions

My major contributions focus on organizational work (attending meetings, presenting updates, brainstorming certain aspects of the challenge) and technical work.

3.8.1 Dataset Setup

After the subtask 2 dataset was aligned and segmented by my colleagues, I inspected the transcriptions of the text (both for Hindi-English and Bengali-English) and discovered a number of artefacts that are detailed in the technical paper. A significant amount of time was spent in identifying and writing about the nature of these dataset characteristics.

The training and validation set data continued to have these artefacts, since an important part of the challenge is being able to learn despite these artefacts. However, the test data needed to be

aggressively cleaned. I achieved this with the help of a colleague and my advisor over 3-4 weeks as follows:

1. First, the extent of noise was identified manually and a decision was made to enlist the help of native speakers to correct the transcriptions for each segment of data in case they were misspelled/misaligned etc.

2. Then, the transcriptions were manually traversed to correct spelling mistakes. Additionally, a hitherto unseen characteristic of this dataset was that many punctuations were enunciated in the audio data, since the audio was a technical lecture (URLs, spoken code snippets, etc.). This data's transcriptions were first normalized into actual punctuation symbols (since various transcriptions transcribed it both as the actual punctuation symbol and the *name* of the symbol in English/Hindi/Bengali). During this process, a punctuation mapping was also created that maps English/Hindi/Bengali words (names of symbols) to their respective symbols. During evaluation, this mapping was applied to the submitted hypothesis so that, for example, a prediction '`comma`' for the ground truth ',' is not penalized.

3. Another prevalent error was that English words were written both in the Latin script and in the native script. Although most of these errors were corrected by native speakers, some still remained. In order to convert all English words to the Latin script, every native script word was transliterated to English and checked against an English dictionary to see if it is an actual English word. This mapping was verified and corrected (to remove words that are both English and native script words) by us. We thus had a transliteration mapping. We reported two types of WERs in the final leaderboard, with/without this mapping applied to the hypothesis. The normal WER checked the additional ability of systems to make predictions in the correct script, while the transliterated WER avoided penalizing this discrepancy.

3.8.2 Setting up the Evaluation pipeline

My colleagues primarily set up the evaluation and submission pipeline, which involved handling any incoming email (a submission) to a designated email address, computing WERs for the relevant subtask automatically, and sending a reply containing the computed WER.

Along with my colleagues, I discussed various error modes and appropriate responses for any incoming email. I also resolved a few nasty bugs and supplied the evaluation pipeline for Subtask 2.

3.8.3 Setting up the Baseline Recipe

I set up the Kaldi baseline recipe for subtask 2 and released its code with a detailed README on the challenge GitHub repository. The exact details are available in the technical section. My motivation for building the recipe was as follows:

1. The basic recipe code was derived from other Kaldi recipes (all Kaldi recipes share the same basic code) and adapted to work with our Subtask 2 data.

2. A naive lexicon did not achieve reasonable performance and resulted in a system with very high WERs. As a result, the majority of the innovation was in building a good lexicon. Improving the lexicon drastically brought down the WERs!

3. The description of the lexicon is as follows: Two different lexicons are used, each for Hindi-English and Bengali-English language pair. For each lexicon, the pronunciations are generated as follows for the entire vocabulary in the respective training set. If the word is in the Devanagari/Bengali script, we consider the respective pronunciation as the word's character sequence. This is because both languages have phonetic orthographies. To obtain pronunciations for English words, we use an open-source g2p package (`https://github.com/Kyubyong/g2p`). This

package provides pronunciations for numericals, retrieves pronunciations from CMUDict dictionary [66] for words that appear in its vocabulary and predict new pronunciations for words that do not. We also obtain pronunciations for the punctuations by mapping to their corresponding English words.

3.8.4 Paper Writing

For the technical paper, I wrote the sections pertaining to what I had worked on during the course of the challenge. We submitted the paper to Interspeech 2021 and uploaded a preprint to arXiV (`https://arxiv.org/abs/2104.00235`).

3.9 Collaborators

Researchers who collaborated on this project:

Anuj Diwan[1], Rakesh Vaideeswaran[2], Sanket Shah[3], Ankita Singh[1], Srinivasa Raghavan[4], Shreya Khare[5], Vinit Unni[1], Saurabh Vyas[4], Akash Rajpuria[4], Chiranjeevi Yarra[6], Ashish Mittal[5], Prasanta Kumar Ghosh[2], Preethi Jyothi[1], Kalika Bali[3], Vivek Seshadri[3], Sunayana Sitaram[3], Samarth Bharadwaj[5], Jai Nanavati[4], Raoul Nanavati[4], Karthik Sankaranarayanan[5], Tejaswi Seeram[7], Basil Abraham[7]

Affiliations:

[1] Computer Science and Engineering, Indian Institute of Technology (IIT), Bombay, India
[2] Electrical Engineering, Indian Institute of Science (IISc), Bangalore 560012, India
[3] Microsoft Research India, Hyderabad, India
[4] Navana Tech India Private Limited, Bangalore, India
[5] IBM Research India, Bangalore, India
[6] Language Technologies Research Center (LTRC), IIIT Hyderabad, 500032, India
[7] Microsoft Corporation India

Chapter 4

Bridging Scripts by Grounding in Speech

In Chapter 2, we explored how transfer learning from a high-to-low resource language via transliteration of the high-resource text to the low-resource script can improve the low-resource ASR system more than standard transfer learning approaches. In that work, we assumed the existence of a transliteration system between the two languages.

Note that in Chapter 2, we saw that the kind of transliteration that most benefits the downstream ASR task is one in which the two texts when written in the two scripts (source and target scripts) preserve some notions of sound i.e. their pronunciations in their respective languages have large correlation and acoustic consistency. This is indeed what one usually expects of a perfect transliteration system, as well. In this work, we explore the problem of learning a transliteration system from a language A to a language B in a weakly supervised fashion. Since we are interested in preserving the pronunciations or speech patterns, we use the audio domain as a grounding space for 'grounding' text in the two languages. Another motivation for exploring a transliteration system from a high-to-low resource language is that there are very few transliteration systems that go *from* a high-resource language *to* a low-resource language, as described in [69].

Formally, assuming we are given labelled monolingual speech data in the two languages, and that no other resource is available for the low-resource language, we devise a novel two-way NMT-like method that can produce a transliteration system from language A to language B. This is a weakly supervised system because we only have access to monolingual speech data; there is no parallel transliteration data available. The next few sections describe the methodology and preliminary results.

4.1 Technical Abstract

We present Bridge-H2L, a method of converting labeled ASR data from a high resource language such as English to a target low resource language with limited labeled data. Bridge-H2L uses a novel two-way neural machine translation model to convert high-resource phonemes to low resource graphemes and back. For training this model we do not require direct labeled data, and instead design two indirect supervision strategies. First, we use the high-resource ASR model to serve as a bridge for generating high-quality pseudo parallel data. Second, we use a round-trip reconstruction loss to select low-error self-supervised data. Our method does not make any assumption on the relatedness of the two languages or the presence of any transliteration libraries.

4.2 Related Work

- Transliteration-based multilingual ASR [5], [43].
- Transliteration-based augmentation for code-switched speech [70], [71].
- Using mismatched transcriptions to adapt ASR [72]

- Not transliteration, but adapting a high-resource ASR system to a low-resource language [17], [73].

4.3 Approach

Let h be the high resource language and let l be the low resource language. Let X_h and X_l denote the audio input sequences in the high and low resource ASR corpus. We define (X_h, Y_h) and (X_l, Y_l) as the high and low resource corpus respectively, where $Y_h \in \boldsymbol{\mathcal{Y}}_h$ and $Y_l \in \boldsymbol{\mathcal{Y}}_l$ such that $\boldsymbol{\mathcal{Y}}_h$ and $\boldsymbol{\mathcal{Y}}_h$ denote the sets of all grapheme sequences in the high and low resource scripts respectively. We also let $X_h \in \boldsymbol{\mathcal{X}}, X_l \in \boldsymbol{\mathcal{X}}$ such that $\boldsymbol{\mathcal{X}}$ denotes the set of all audio sequences. For each corpus, we also define (X_*^{trn}, Y_*^{trn}), (X_*^{dev}, Y_*^{dev}) and (X_*^{tst}, Y_*^{tst}) as the training, development and evaluation sets respectively. We assume the availability of a grapheme to phoneme conversion system $\boldsymbol{\mathcal{Y}}_h \mapsto \boldsymbol{\mathcal{P}}_h$ for the high resource language, where $\boldsymbol{\mathcal{P}}_h$ is the set of all phoneme sequences possible in h.

4.3.1 Round-trip Transliteration

The $\boldsymbol{\mathcal{Y}}_h \mapsto \boldsymbol{\mathcal{P}}_h$ system defined above is used to obtain phoneme sequences P_h for each transcription in Y_h. Consider an ASR system $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h]$ that takes as input a speech utterance $X \in \boldsymbol{\mathcal{X}}$ and outputs a phoneme sequence $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h](X)$ which is in $\boldsymbol{\mathcal{P}}_h$. This ASR system thus transcribes any speech utterance to a sequence of phonemes in the high-resource language h. This system is thus trained with (X_h^{trn}, P_h^{trn}) as the training set and (X_h^{dev}, P_h^{dev}) as the development set.

The low resource audio data $X_l \in \boldsymbol{\mathcal{X}}$ is passed through the trained ASR system $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h]$ to obtain phonemic transcriptions of the audio $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h](X_l)$ which are in $\boldsymbol{\mathcal{Y}}_h$. Using this, we have therefore, for the low-resource data, in addition to speech samples X_l and transcriptions Y_l, a set of machine-generated synthetic phoneme sequences in the high resource language, $\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h(X_l^{trn})$.

Using this, we can train a preliminary transliteration system from high-resource phonemes to low-resource graphemes, $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l]$ that trained on the pair $([\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h](X_l^{trn}), Y_l^{trn})$. As we recall, the inputs $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h](X_l^{trn}) \in P_h$ are synthetic high-resource transcriptions obtained by passing low-resource audio X_l through $[\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h]$, and $Y_l \in \boldsymbol{\mathcal{Y}}_l$ are the corresponding ground truth sequences in the low resource script.

Although this gives a preliminary transliteration system, this suffers from a huge language bias on the output side (since all the training examples it has seen are real low-resource sentences). This bias is undesirable since, although the system is trained on data that contains synthetic high-resource phonemes paired with real low-resource graphemes, we want to use it on data with real high-resource phonemes (P_h) to generate 'synthetic' transliterated low-resource graphemes. In practice, simply using this system and using beam search decoding generates low-resource sentence outputs that aren't very phonetically consistent with the input and instead resemble real low-resource sentences/words.

Therefore, we devise a round-trip transliteration idea to combat this. In addition to training the forward transliteration system $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l]$, we train a backward transliteration system $[\boldsymbol{\mathcal{Y}}_l \mapsto \boldsymbol{\mathcal{P}}_h]$ using the pair $(Y_l^{trn}, \boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{P}}_h(X_l^{trn}))$. This is basically the $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l]$ system with input/output swapped.

Given a high-resource training transcription phoneme sequence $P_h^{trn_i}$ from the corpus P_h^{trn}, we decode it using a Diverse Beam Search on $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l]$ to get n diverse outputs, where each output is a low-resource grapheme sequence. Some of these sequences will be phonetically inconsistent as explained above, but the hope is that due to diversity, some of these will be more phonetically consistent than outputs obtained using simple beam search decoding. Denote these outputs as $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l](P_h^{trn_i})_1$ through $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l](P_h^{trn_i})_n$.

Each output $[\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l](P_h^{trn_i})_j$ is transliterated back into high-resource phoneme sequences using $[\boldsymbol{\mathcal{Y}}_l \mapsto \boldsymbol{\mathcal{P}}_h]$'s 1-best Beam Search output. These outputs are thus $[\boldsymbol{\mathcal{Y}}_l \mapsto \boldsymbol{\mathcal{P}}_h]([\boldsymbol{\mathcal{P}}_h \mapsto \boldsymbol{\mathcal{Y}}_l](P_h^{trn_i})_j)$. Now, the hope is that through this round trip transliteration, those indices j for which the reconstructed

	Train	Dev	Test
Hindi	40.2	4.97	5.02
Telugu	30.0	2.55	2.44

TABLE 4.1: Size of the various datasets in hours.

phoneme sequence $[\mathcal{Y}_l \mapsto \mathcal{P}_h]([\mathcal{P}_h \mapsto \mathcal{Y}_l](P_h^{trn_i})_j)$ ($P_h^{trn_i}$ passed through $\mathcal{P}_h \mapsto \mathcal{Y}_l$and then $\mathcal{Y}_l \mapsto \mathcal{P}_h$) is *close* to the original phoneme sequence $P_h^{trn_i}$ are *good* reconstructions and therefore the corresponding transliteration pair $(P_h^{trn_i}, [\mathcal{P}_h \mapsto \mathcal{Y}_l](P_h^{trn_i})_j)$ is expected to be a *good* (i.e. phonetically consistent) training pair. This training pair being real on the input side and synthetic on the output side (generated using $\mathcal{P}_h \mapsto \mathcal{Y}_l$) is expected to complement the original 'biased' training data that was synthetic on the input side (generated using $\mathcal{X} \mapsto \mathcal{P}_h$) and real on the output side. Here *close* is yet to be defined: we define the *closeness/goodness* of a transliteration pair $(P_h^{trn_i}, [\mathcal{P}_h \mapsto \mathcal{Y}_l](P_h^{trn_i})_j)$ as the negative phoneme error rate (PER) between $P_h^{trn_i}$ and $[\mathcal{Y}_l \mapsto \mathcal{P}_h]([\mathcal{P}_h \mapsto \mathcal{Y}_l](P_h^{trn_i})_j)$; smaller the PER, the better.

We select a subset S^{trn} of such transliteration pairs based on a manually selected threshold on the goodness of the pairs. This selected subset, in combination with $(\mathcal{X} \mapsto \mathcal{P}_h(X_l^{trn}), Y_l^{trn})$ pairs forms our training dataset for further training the forward transliteration system $\mathcal{P}_h \mapsto \mathcal{Y}_l$and the backward transliteration system $\mathcal{Y}_l \mapsto \mathcal{P}_h$. This transliteration system $\mathcal{P}_h \mapsto \mathcal{Y}_l$can thus be used as a drop-in replacement for an off-the-shelf transliteration system, with some caveats as described in the next section.

4.3.2 Using the trained transliteration system

The transliteration systems were trained on pairs of *sentences* (since that is the only type of supervision available) and not *words*. This is unlike most transliteration systems that map a word to a word. Our systems thus do not fully preserve word boundaries i.e. assuming the input sequence P consists of n words $P_1, \ldots, P_n$ and the output sequence Y consists of m words $Y_1, \ldots, Y_m$. it is neither guaranteed that $m = n$ nor is it guaranteed that Y_j is the transliteration of P_j, where $j \leq min(m, n)$. This sort of transliteration is undesirable (and experiments in the next section show that this kind of transliterated pretraining is not very helpful). Therefore, inference is performed as follows:

1. An input sentence containing k words is cut up into individual words $w_1, \ldots, w_k$. Then, each word w_j is passed to $[\mathcal{P}_h \mapsto \mathcal{Y}_l]$ separately. Now, a simple way to get a transliteration is pick the 1-best output from $[\mathcal{P}_h \mapsto \mathcal{Y}_l]$ and use that as the word's transliteration.

2. To get better performance, we do something slightly more complex: for an input word w_j, first perform Diverse Beam Search in $\mathcal{P}_h \mapsto \mathcal{Y}_l$and for each diverse output, we pass it to $\mathcal{Y}_l \mapsto \mathcal{P}_h$and find the 1-best Beam Search output. Then, we pick the output from $\mathcal{P}_h \mapsto \mathcal{Y}_l$that has the highest goodness when passed through $\mathcal{Y}_l \mapsto \mathcal{P}_h$.

3. Finally, we join each word's transliteration together using a space character to get the sentence's transliteration.

4.4 Experiments

4.4.1 Datasets

We evaluate our approach using English as the high-resource language and Hindi, Telugu as two low-resource languages. We use the 100-hour Librispeech English dataset [21], the Microsoft Speech Corpus (Indian Languages) dataset [22] for Telugu and the Hindi ASR Challenge dataset [23] for Hindi. Table 4.1 presents detailed statistics.

4.4.2 Experimental Setup

All experiments are run using the Fairseq [74] toolkit. We describe the setup for each of the systems described in Section 4.3.

1. $\mathcal{Y}_h \mapsto \mathcal{P}_h$: We use the open-source g2p English package `https://github.com/Kyubyong/g2p` for the $\mathcal{Y}_h \mapsto \mathcal{P}_h$system.

2. $\mathcal{X} \mapsto \mathcal{P}_h$: We use the XLSR architecture [14] for the $\mathcal{X} \mapsto \mathcal{P}_h$model. Starting from the pretrained XLSR model estimated using unsupervised pretraining trained on multilingual datasets for 53 languages, we finetune it on the (X_h, P_h) corpus as described in Section 4.3 using the default hyperparameters described in [14].

3. $\mathcal{P}_h \mapsto \mathcal{Y}_l$: We use the `transformer-tiny` architecture with 2 encoder layers, 2 decoder layers, a hidden size of 64 for both the encoder and the decoder and a learning rate of 0.005 for $\mathcal{P}_h \mapsto \mathcal{Y}_l$. For Diverse Beam Search, we use the inbuilt decoding scripts provided by Fairseq with a diversity hyperparameter of 10.0.

4. $\mathcal{Y}_l \mapsto \mathcal{P}_h$: We use the `transformer-tiny` architecture with 2 encoder layers, 2 decoder layers, a hidden size of 64 for both the encoder and the decoder and a learning rate of 0.005 for $\mathcal{Y}_l \mapsto \mathcal{P}_h$.

5. Downstream ASR system: All downstream ASR systems are trained using the wav2vec2.0 architecture [19]. All models start from the pretrained wav2vec2.0 model estimated using unsupervised pretraining trained on the full Librispeech dataset.

4.4.3 Experimental Results

Here we present some preliminary results obtained as part of this work. We present results on a 10-hr random training subset of the Hindi and Telugu datasets. We compare the approach with two baselines.

- **NoPre**: The wav2vec2.0 model is directly finetuned until convergence on the downstream ASR corpus (X_l, Y_l) using the default hyperparameters (except the number of updates; the number of updates are set such that the model reaches convergence) for 10-hr finetuning described in [19].

- **SelfTrain**: This baseline is inspired by self-training approaches for ASR. Its most direct influence is the approach described in [5]. To setup this baseline, we first take the trained low-resource ASR model from the **NoPre** baseline and pass real English audio X_h from the Librispeech baseline into it. The generated synthetic outputs **NoPre**(X_h) (English audio transcribed to low-resource text using **NoPre**) are considered 'transliterations' of the ground truth English text Y_h. This 'transliterated' data $(X_h, \textbf{NoPre}(X_h))$ is used to further train the **NoPre** checkpoint. Finally, the resulting system is finetuned on the downstream ASR corpus (X_l, Y_l) until convergence. In experiments, we also ablate the choice of further training from **NoPre** (as in the original self-training paradigm) (and we call this approach **SelfTrain**) vs. using these transliterations to train wav2vec2.0 from its self-supervised checkpoint, rather than the **NoPre** checkpoint (we call this **SelfTrain****NoPre**)

For all Hindi ASR systems, we report results with/without decoding with a 4-gram KenLM [34] trained on the training transcriptions. For Telugu, we only report results without LM. We also ablate choices made in the approach. We ablate parts of our approach by observing what happens when we remove the final inference-time Diverse Beam Search (when generating transliterations. We call this ablation **Our****DBS**) and when we finetune from the **NoPre** checkpoint rather than the self-supervised wav2vec2.0 checkpoint (we call this ablation **NoPre**+**Our**). Table 4.2 presents results for Hindi and Table 4.3 presents results for Telugu.

Method	Without LM		With LM	
	Dev	Test	Dev	Test
NoPre	26.476	25.874	16.142	15.723
SelfTrain	25.170	24.435	16.529	15.952
SelfTrain\NoPre	27.491	26.926	16.476	16.061
Our	26.500	25.803	16.235	15.859
Our\DBS	26.704	26.032	-	-
NoPre+Our	25.078	24.218	16.391	15.783

TABLE 4.2: WERs on 10-hr Hindi training subset and various approaches

Method	Without LM	
	Dev	Test
NoPre	36.844	35.751
SelfTrain	35.878	34.067
SelfTrain\NoPre	37.902	36.713
Our	37.04	36.31
Our\DBS	37.257	35.921
NoPre+Our	-	-

TABLE 4.3: WERs on 10-hr Telugu training subset and various approaches

4.5 Discussion

For Hindi, from Table 4.2, one can first observe that, comparing the best results for each method i.e. the 'with LM' results, one observes that **NoPre** outperforms the other approaches i.e. none of the fancier approaches manage to beat **NoPre**. This indicates that given an LM, **NoPre** probably benefits enough from the pretrained wav2vec2.0 model that no further pretraining is necessary. However, in Chapter 2, we did see improvements with some kind of pretraining despite this. This means that currently our approach is, unfortunately, not a great approach.

However, if we are to compare the 'without LM' WERs for Hindi, we can see some interesting trends emerge. Firstly, **NoPre+Our** performs the best out of all the methods. Additionally, there is a large WER gap between **Our** and **NoPre+Our** (and indeed, between **SelfTrain** and **SelfTrain\NoPre**); this indicates that starting the pretraining from a checkpoint that has already seen downstream data (**NoPre**) is much better than starting without it! Further, removing Diverse Beam Search worsens results, which ablates that decision for Hindi. However, the trend is somewhat different for Telugu. In Table 4.3, we can observe that removing Diverse Beam Search improves results, so this architectural decision may not be relevant to the final performance.

Comparing approaches fairly (i.e. not starting with the **NoPre** checkpoint) i.e. comparing **SelfTrain\NoPre** with **Our**, we observe that **Our** does much better, for both languages. This is also borne out by the quality of transliterations that we shall present in the next section.

4.6 Qualitative Analysis of Transliterations

Figure 4.1 shows two examples of transliteration from. As a transliteration system, our model qualitatively easily beats the **SelfTrain** baseline and seems to be of high quality (although these high quality transliterations do not seem to significantly improve downstream task performance). This approach may thus be more useful for other tasks that can benefit from high quality transliterations, rather than Speech Recognition.

Orig: HIS VISIT OF THE DAY BEFORE IT IS TO BE REMEMBERED HAD BEEN VERY SHORT AND VERY LITTLE EXPLICATIVE HE FOUND TREVILLE IN A JOYFUL MOOD
SelfTrain: हिज विजर अफ दडे ब फोर एरिस्टीवीए मिंबर्ड हद बिन वेरी शोर्ट अनवेयरि लिरल एक्सप्लकेथ ही फैनस्रवेय अनई जॉई फोर्मेड
Ours: है विसत अवि दिया ड़े बिफोर इत इस तुर् बी रिमेंबर है बिन वहरी शोर्ट अन्द वहरी लिएतल इक्सप्लिस्टक्ट ही फान्द त्रेविल इन अया जोफल मुद

Orig: DON'T MOTHER ME SAID THE JOLLY WIDOW WITH A KINDLING EYE
SelfTrain: दोनट मदर मी स्षेद दि जाली विदोंसु विद यह केनिलिंग आ
Ours: दोन मदर में से दिया जाली विदों विद अया केनिलिंग आया

FIGURE 4.1: Two examples of transliteration from English to Hindi, with **SelfTrain** and **Our** system

4.7 Conclusion

In this chapter, we presented some preliminary work on bridging scripts via speech to build a weakly-supervised transliteration system. Qualitative results indicate that our transliteration system is superior to a baseline transliteration and is of high quality; however the benefits of this approach for a downstream ASR task are fairly minimal. In future work, we plan to try out this system for other tasks as well as explore pretraining from accented English speech.

4.8 Collaborators

Researchers who collaborated on this project:

Tarang Jain[*,1], **Anuj Diwan**[*,1], Shreya Khare[2], Ashish Mittal[1,2], Tejas Indulal Dhamecha[2], Sunita Sarawagi[1], Preethi Jyothi[1], Samarth Bharadwaj[2]

where $*$ contributed equally.

Affiliations:

[1]Indian Institute of Technology Bombay ; [2]IBM Research

Chapter 5

Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages

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.

5.1 Introduction

End-to-end (E2E) automatic speech recognition (ASR) systems are becoming an increasingly popular choice for ASR modeling [75]. 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 example that illustrates the potential utility of RNR. Suppose our reference text is “call the bus" and we train an E2E model with the output vocabulary $\Re$ that uses a single character ζ to represent both “c" and “k" in the original English alphabet. If the E2E model is able to accurately predict ζ, 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 E2E 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 in each language to produce an appropriate reduction, train an E2E ASR system with the reduced vocabulary and finally reconstruct the 1-best ASR prediction using an FST-based reconstruction module and a recurrent neural network (RNN) based module.

5.2 Related Work

Our line of work is closely related to error correction models for ASR that explicitly correct errors made by the ASR system [76]. Several such efforts for error correction of ASR systems have been carried out in prior work; [77] presents a review. Past work has looked at reordering ASR hypotheses using machine translation-inspired techniques [78], [79], leveraging contextual information using recurrent models and rescoring confusion networks generated by the ASR system [80]–[85]. Our proposed framework is different in that it is not strictly a postprocessing technique and requires the ASR system in itself to be modified 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 [86]–[88].

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 [89]. Attempts at redefining the phone set in a data-driven manner have also been made for monolingual ASR systems [90]. Recent work has looked at the influence of merging phones for code-switched speech recognition [91] and building language-agnostic multilingual systems by mapping graphemes in Indian languages to a single script [92].

5.3 RNR: Reduce and Reconstruct

RNR proceeds through the following steps:

1. We 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 $\Re$. For a given language, its original alphabet will be the Unicode block of the script used by the language. For example, the original alphabet of Gujarati is the Unicode codepoint range `U+0A80` to `U+0AFF` and similarly the Unicode codepoint range `U+0C00` to `U+0C7F` for Telugu.

2. The transcriptions in the speech corpus used for training are then modified to come from $\Re$ i.e. each character in each transcription is converted to the character in the reduced alphabet $\Re$ that it maps to. Punctuation is not modified. The speech input is also not modified. An E2E ASR model is then trained on these resulting transcriptions. The output predictions from this model will therefore be from $\Re$. We can call this the *reduced ASR system*.

3. A reconstruction module is trained to reconstruct transcriptions in the original alphabet from the reduced alphabet. Note that a given character in the reduced alphabet has multiple possible reconstructions in the original alphabet and therefore it is a non-trivial task that requires contextual information and language modelling to figure out the correct reconstruction. We implemented two reconstruction modules, an FST-based reconstruction model and an encoder-decoder network-based reconstruction model.

4. We therefore have an ASR system that decodes a transcription in the reduced alphabet $\mathfrak{R}$ given a speech signal as input and a reconstruction module that converts transcriptions from the reduced alphabet to the original alphabet. We provide the 1-best beam search decoded output from the ASR system as input to the reconstruction module and use the output of the reconstruction module as the final hypothesis.

5. Using n-best decoded transcriptions ($n = 100$) from the ASR model, for reconstruction, resulted in exactly the same performance as just using the 1-best, presumably due to all of the n transcriptions being fairly similar. Thus we only use the 1-best for all our experiments. We also considered making part of the decoding lattice available during reconstruction (i.e. using the reconstruction module somewhat doubly as a reconstructor as well as a language model rescorer) but this proved difficult to implement given time constraints and will be explored in future work.

5.3.1 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] [2]

We expect such a reduction function to work well for our architecture pipeline because of the following intuition:

- If the reduction is based on merging graphemes that are acoustically similar/confusable but linguistically non-confusable, the reduced ASR system is expected to be able to predict the (reduced) transcriptions with a significantly better WER, because it does not have to disambiguate the acoustically similar graphemes any longer.

- Because the graphemes that are merged are linguistically non-confusable, the reconstruction module that converts from the reduced to the original alphabet is expected to be able to disambiguate them. Although it becomes harder for the reconstruction module to correctly reconstruct as the reduction becomes more and more aggressive, we hope that since we are performing disambiguation using a text-to-text reconstruction model i.e. at the resolution of graphemes, rather than performing it at the audio level i.e. at the frame level, we can recover the original sequence by being able to model long-range contextual dependencies better.

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. For a given place of articulation, we merge the graphemes corresponding to all four plosives into a single grapheme in $\mathfrak{R}$. Overall, the graphemes corresponding to the above-mentioned twenty plosives are reduced to five graphemes in $\mathfrak{R}$. There are five graphemes corresponding to nasal sounds in Gujarati and Telugu, which we reduce down to a single grapheme.

[1] Reduction mappings could be data-driven and learned automatically from speech, which we leave for future work.

[2] 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.

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 the reduced alphabet. 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]

5.3.2 Reconstruction Modules

FST-based Reconstruction

Our first reconstruction model is implemented using a cascade of FSTs. This structure is somewhat similar to the decoder in [93], 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 shortestpath algorithm on the composed FST:

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

The FSTs S, E, L and G (described below) are weighted using negative log probability costs, thus making shortestpath appropriate. The output labels of O will correspond to the best reconstructed word sequence in the original alphabet.

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 introduce 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 the standard pronunciation lexicon FST (that maps sequences of phonemes to sequences of words) [94]. 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 E2E 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`.

[3] The exact reduction functions for these reductions can be found at the following Google Drive link: `https://drive.google.com/drive/folders/18uGZFweTVST1amjeUaBMZ_mtTn3GHEL4?usp=sharing`

***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 Ngram language model is trained on the training set transcriptions of the full speech data.[4]

Encoder-decoder reconstruction model

The sequence-to-sequence learning paradigm [95] using RNNs can be naturally applied to the reconstruction problem, where the input sequence is a reduced grapheme sequence and the output sequence is a sequence of words in the original alphabet. This is similar to a Neural Machine Translation (NMT) task where the source 'language' comes from the reduced alphabet $\Re$ and the target language is the original language.

The FST-based reconstruction model is restricted to work with an Ngram model, while RNN-based models have the ability to model long range dependencies beyond fixed contexts. Additionally, RNN-based models that predict words as output can benefit from using pretrained word embeddings, like FastText [96] and GloVE [97] embeddings, that are available for a large number of languages.

Our encoder network is a multi-layer bidirectional long short term memory (LSTM) network and the decoder is a multi-layer unidirectional LSTM network equipped with a standard additive attention mechanism [95]. As with standard encoder-decoder models, this reconstruction network is trained to minimize a cross-entropy loss. We experiment with two different tokenization schemes: 1. G2W: Input is a sequence of graphemes and the output is a sequence of words. 2. B2B: Both the input and output sequences are tokenized into subword units using the Byte-Pair Encoding (BPE) algorithm [31].

An advantage of a reconstruction module is that it only requires textual data to be trained. This is particularly of use with low-resource languages that may not have large quantities of labeled speech, but have access to larger text-only corpora. We note here that because the reconstruction task is supposed to reconstruct reduced transcriptions predicted by the ASR system into the original alphabet, the reconstruction model actually needs reduced grapheme sequences predicted by an ASR system as its input. This information is clearly unavailable in text-only corpora since there is no corresponding speech available.

Instead, we reduce each sentence of the large corpus using the chosen reduction and use this reduced grapheme sequence as a proxy for decoded ASR hypotheses. Indeed, one can imagine that these reduced sequences do not follow the same probability distribution as the decoded ASR hypotheses and this may not be an optimal choice. Therefore one can also consider training on the large corpus to be a pretraining step, and thereafter one might consider generating ASR hypotheses for all the training instances of the speech corpus and using the ASR predictions as training data to finetune the pretrained reconstruction model. We investigate these possibilities further in Section 5.4.6. Note that [86] took a different approach by generating speech from the text-only corpus using Text-to-Speech (TTS) to get speech inputs for textual data. We do not use that approach here.

5.4 Experiments and Results

5.4.1 Public Results

All results are publicly available at `https://docs.google.com/spreadsheets/d/e/2PACX-1vRPrhSrAr08DJykSTuOW4M79TyubEHCbwCdJU4QdFCo4Sv1mlNzCohr8ChMz6d4lAocRXKdDXVMHp4k/pubhtml`. All the hyperparameter tuning results as well as exploratory experimental results are available here.

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

5.4.2 Dataset Details

We use the Microsoft Speech Corpus (Indian Languages) dataset [98] [22]. The Gujarati and Telugu speech corpora comprise 39.1 hours and 31.3 hours of training speech, respectively. Table 5.1 provides detailed statistics relevant to our train/dev/test splits. The OOV rates for Gujarati and Telugu on the test set are 5.2% and 12.0%, respectively.

TABLE 5.1: Detailed statistics of Gujarati and Telugu datasets

Language	Vocab	Set	# utts	# tokens
Gujarati	38774	train	22307	273085
		dev	500	6181
		test	3075	36335
Telugu	38034	train	22241	194555
		dev	500	4988
		test	3040	28413

5.4.3 Experimental Setup

ASR systems for both languages were built using the ESPNet Toolkit [29]. We used 80-dimensional log-mel acoustic features with pitch information. Our baseline E2E model is a hybrid LSTM-based CTC-attention model [30]. All our ASR systems use BPE tokenization [31], 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 decoder 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 were trained on an Nvidia GeForce GTX 1080 Ti GPU.

All the reconstruction FSTs were implemented using the OpenFST toolkit [99]. The G FST represents a Kneser-Ney smoothed 4-gram LM that was trained using the SRILM toolkit [67]. 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.

The encoder-decoder reconstruction models were implemented using the Fairseq toolkit [74]. For the G2W tokenization, we use a total vocabulary of $120K$ words in Gujarati and $150K$ words in Telugu. The word embeddings were initialized with pretrained FastText embeddings [96]. For the B2B tokenization, we use a BPE vocabulary of 25000 for both languages. We used 300-dimensional embedding layers at the input and output. We used 3, 512-unit encoder layers and 3, 256-unit decoder layers. We used a dropout rate of 0.2. During inference, we used a beam decoder with a beam width of 30.

5.4.4 ASR Experiments

Table 5.2 lists WERs for both Gujarati and Telugu on the development and test sets using different reduction functions and trained on two different training set 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.

TABLE 5.2: 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

Identity refers to the baseline system with an unaltered grapheme set. ρ_1 is the reduction function discussed in Section 5.3. 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 WERs are comparable to previously published results [100] 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.

All hyperparameters were obtained after careful hyperparameter finetuning. The `guj-tune` and `tel-tune` sheets available at the link in Section 5.4.1 contain detailed results for the hyperparameter tuning. This is a good quick reference to see the effect of varying different hyperparameters like embedding sizes, number of layers, dev set sizes, decoding beam widths, decoding penalties, etc.

TABLE 5.3: 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.

5.4.5 FST-based Reconstruction Experiments

Table 5.3 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 Edit Distance FST and we observe large reductions in WER for both the ρ_1 system and the identity

system. 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.

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. Table 5.4 provides details about this hyperparameter tuning.

TABLE 5.4: Tuning d and λ using dev WERs from the FST-based reconstruction model for the Full duration.

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
1	5	identity	40.0	40.7	43.3	45.5
		ρ_1	39.2	39.0	40.8	43.7
2	2.5	identity	42.7	40.3	43.0	44.7
		ρ_1	43.7	39.7	42.4	44.2
	5	identity	38.4	38.7	41.3	43.4
		ρ_1	37.9	37.1	39.1	41.9
	10	identity	39.2	40.1	41.9	44.1
		ρ_1	38.5	39.1	39.6	42.4
3	5	identity	37.9	37.8	40.6	42.5
		ρ_1	**37.8**	**36.5**	**38.5**	**41.2**

5.4.6 Encoder-Decoder Reconstruction Experiments

The encoder-decoder reconstruction model takes 1-best predictions from the ASR systems in Table 5.2 as input and recovers words in the original alphabet. We use two different datasets in these reconstruction experiments along with the two tokenizations discussed in Section 5.3. The corpus *large* refers to two open-source Wikipedia datasets [101] [102] for Gujarati and Telugu, consisting of 4.2 million tokens with a vocabulary of size 280K for Gujarati, and 12.1 million tokens with a vocabulary of size 950K for Telugu. *asr* refers to the additional use of decoded hypotheses from the ASR model for the ASR training data to train the reconstruction model. Data from the *asr* corpus is used to further finetune the model trained using the *large* corpus.

We observed that training the reconstruction model on only the ASR training transcripts performed badly in comparison to the use of *large*. Only using the ASR training transcripts obtained much worse and highly variable WERs for both Gujarati (approximately ranging from 45 to 55) and Telugu (approximately ranging from 60 to 80), indicating that the model was overfitting on the training set. Decreasing the architecture size did not help either. These negative results inspired us to try pretraining on *large* followed by finetuning on *asr*.

Table 5.5 shows the overall performance of our RNR paradigm using the RNN-based reconstruction model on both Gujarati and Telugu for the Full training duration. We did not run the 10-hr durations due to time constraints. For both Gujarati and Telugu, we see significant improvements in ASR performance using the ρ_1 reduction mapping (shown in the last row) compared to the baseline ASR system without any reconstruction (shown in the first row).

The B2B tokenization scheme is more effective than G2W. This is likely because of the large number of rare tokens available in the *large* text corpus; G2W will have parameters corresponding to each of these words which may not be reliably estimated while B2B is able to share parameters across these rare tokens due to its subword vocabulary.

However, note that overall, only marginal improvements are obtained; the FST reconstruction is much more effective than the NN reconstruction experiment. We believe that this phonemenon occurs due to how we modelled the NN as a black-box machine translation task between two 'languages'. On the other hand, in the FST reconstruction, we introduced additional supervision by incorporating information about the reduction mapping. In future work, we will exploring imposing these additional constraints on the NN reconstruction model.

TABLE 5.5: WERs from the encoder decoder-based reconstruction for Gujarati and Telugu

Tokens	Corpus	Reduction	WER (Guj)		WER (Tel)	
			Dev	**Test**	**Dev**	**Test**
Baseline			41.5	43.2	44.1	46.8
G2W	large	identity	41.6	43.3	44.2	47.3
		ρ_1	40.8	43.3	42.9	46.6
	asr	identity	41.8	43.3	45.8	48.7
		ρ_1	40.6	42.8	42.5	46.4
B2B	large	identity	41.5	43.2	45.2	46.8
		ρ_1	41.8	44.0	43.1	46.6
	asr	identity	42.0	43.5	46.7	49.1
		ρ_1	**39.9**	**42.6**	**41.3**	**45.2**

An interesting result to observe is that the B2B tokenization is not very effective for the identity reduction experiments. This is likely because for the identity reduction, it is essentially being trained to output exactly the same tokens as the input and hence gets massively biased towards the identity function. The ASR finetuning does not suffice to improve it from this bias. Additionally for the ρ_1 reduction, the ASR finetuning massively benefits the B2B experiments as compared to smaller gains for the G2W experiments.

All hyperparameters were obtained after careful hyperparameter finetuning. The `guj-nn` and `tel-nn` sheets available at the link in Section 5.4.1 contain detailed results for the hyperparameter tuning.

TABLE 5.6: 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 5.7: 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

5.4.7 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 5.6 lists WERs for both Gujarati and Telugu using the best FST reconstruction system ($d = 3, \lambda = 5$) for both training durations. NN reconstruction experiments are not reported here since they performed worse than the FST reconstruction, but these experiments are available at the link in Section 5.4.1.

As expected, 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 the ρ_1 mapping performs significantly better than the baseline and the identity mapping for both languages, yielding up to 7% relative reduction in test WER. 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 5.3 as well.

5.4.8 Using conformers

To demonstrate that RNR is effective across different E2E architectures, we also train a conformer-based model [103] for Gujarati 10-hours shown in Table 5.7. These models were also built using the ESPNet Toolkit [29]. We use 2 encoder layers and 1 decoder layer, each with 350 units and 4 attention heads. The encoder output has 256 units. 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.

5.4.9 Discussion and Qualitative Analysis

5.4.10 Choice of reduction function

One might wonder about the impact of the reduction function on ASR performance. In Table 5.2, we show how a randomly chosen ρ_1-rand can be detrimental to the ASR system. Additionally, we explore employing 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 rather than reducing quadruplets of plosives (as in ρ_1) 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 results in reduced

alphabets with no underlying acoustically confusable sounds is more beneficial with RNR than other reduction functions.

5.4.11 Effect of reduction function on correcting ASR errors

Consider the pre-reconstruction ASR system (from Section 5.4.4) 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. To do this, we first align the identity hypothesis text to the reference text. Call this alignment a_1. We similarly align the reduced hypothesis text to the reduced reference text. Call this alignment a_2. Let X be the number of substitution errors made by the identity system i.e., the number of character substitutions $a \rightarrow b$ (where $a \neq b$) in the identity system alignment a_1. Out of these substitutions, we count the number of substitutions Y for which the corresponding character alignment in a_2 is correctly predicted. This percentage is calculated as $\frac{Y}{X} \times 100\%$. 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 a sizable number of substitution errors incurred by the identity system.

5.4.12 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 5.8. 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 5.8: 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

5.4.13 Ambiguity in reconstructions

We further examine why Telugu is benefiting from ρ_1 slightly more than Gujarati despite having higher OOV rates. In the Telugu vocabulary, roughly $35K$ reduced word forms deterministically map to a single word in the original alphabet, while this number is less than $30K$ for Gujarati. Thus, there is less ambiguity in Telugu reconstruction overall. Also, all the plosives together make up 35.5% and 33.4% of the total number of graphemes in Gujarati and Telugu, respectively. Smaller number of grapheme merges in Telugu could be another reason for Telugu's improved performance with RNR.

5.4.14 Illustrative Examples

Figure 5.1 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.

5.5 Future Work

Future work will consider ideas such as:

FIGURE 5.1: An illustrative example each from Gujarati and Telugu using FST reconstruction, $\mathbf{d} = 3, \breve{} = 5$. *R* and *I* are the reference and identity transcriptions.

- Automatically learning a data-driven reduction mapping from the speech data. This mapping can be explicitly incentivized to correct for errors made by the ASR system.
- Combining the two-module approach into a single-module using multitask learning. An intermediate bottleneck layer can be trained to learn the reduced text and the output layer can be trained to learn the actual text.
- Make use of the ASR system decoding lattice during reconstruction, rather than just the 1-best hypothesis.

5.6 Conclusions

This work proposes an effective reduce-and-reconstruct technique that can be used with E2E ASR systems for low-resource languages. We first reduce the output vocabulary by meaningfully merging graphemes and then build an E2E ASR system. The output from this system is then passed through a reconstruction module which is trained to recover the original grapheme sequence from the reduced sequence making advantage of larger text-only corpora. 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.

5.7 Work done during the BTP

Part of this work was done during an RnD Project during Spring 2020. Here, I describe the additional work that was done during the BTP:

1. Guided by peer reviews, rewriting and improving sections of the paper to make them clearer
2. Adding 10-hr low-resource experiments and demonstrating how the approach is more effective in this low-resource scenario
3. Adding conformer experiments, demonstrating that the approach works even on an improved NN architecture
4. Adding new qualitative and ablation experiments, such as the effect of the choice of reduction function, perplexity decrease due to reduction, ASR error correction analysis, etc.

5.8 Collaborators

Researchers who collaborated on this project:

Anuj Diwan, Preethi Jyothi
Affiliation:
Indian Institute of Technology Bombay

Chapter 6

Other Work

6.1 Phoneme-based Multilingual ASR

Prior work [37] has explored improving phoneme recognition by using a multilingual system that has a shared phone layer and language-specific output phoneme layers. This approach shows improvements over a private phoneme model and a shared phoneme model. We conducted some preliminary work on extending these ideas along several directions. These results are available at:

1. I re-implemented the Allosaurus paper [37] by extending the publicly released code [1] to replicate the published results.

2. Discovered a number of bugs in the aforementioned repository and submitted 2 Pull Requests (#7, #8) to fix these bugs.

3. The dataset sizes of the languages used in the paper varied from 1148k utterances (English) to 8k utterances (Russian). To improve performance on an imbalanced set of languages, we tried using temperature-based sampling [104] [105].

   Given N languages, let $P_D(i)$ be the probability with which we sample an utterance from language i. If D^i_{train} is the training set for language i, then for temperature-based sampling,

   $$P_D(i) = \frac{q_i^{1/\tau}}{\sum_{k=1}^{n} q_k^{1/\tau}} \text{ where } q_i = \frac{\left|D^i_{\text{train}}\right|}{\sum_{k=1}^{n} \left|D^k_{\text{train}}\right|}$$

   Here τ controls the temperature. If $\tau = 1$, then we perform proportional sampling i.e. languages with larger datasets are sampled more, in proportion to their dataset size. If $\tau = \infty$, then we perform uniform sampling i.e. all languages are sampled uniformly, irrespective of their dataset sizes. An intermediate value works best.

4. The paper uses a shared phone layer and per-language output phoneme layers. Inspired by this idea, we attempted to develop an ASR system with a shared phoneme layer and language-specific output grapheme layers. This approach is well-suited for languages with simple phoneme-grapheme mappings i.e. phonemic orthographies, like Indian languages.

   Because phonemes and graphemes are nearly interchangeable for Indian languages, another line of research that we explored was using a shared grapheme (rather than phoneme) layer in a script shared across all Indian languages. If this script was the International Phonetic Alphabet (IPA), it would be a shared phoneme layer. However, it is easier and more natural to use Extended Devanagiri. We created mappings from Devanagiri to per-language scripts in order to implement this idea.

[1] `https://github.com/xinjli/allosaurus`

5. The paper compares the shared layer model and the private layer model. We developed some ideas to combine the two i.e. develop a hybrid model that has both shared layers and private layers, where one can automatically learn per-language or even per-utterance weights to choose between these layers depending upon how much sharing is necessary for good performance. For example, high-resource languages may prefer the private layers while low-resource languages may prefer the shared layers.

6.2 Phoneme-based Transliteration for Hindi

We developed our own transliteration technique to remedy the errors made by the off-the-shelf `indic-trans` [20] transliteration system. Our code is publicly available at `https://github.com/ajd12342/phoneme-translit`. Preliminary experiments indicated that using this system instead of `indic-trans` does not result in significant downstream improvements. Nevertheless, we describe it here.

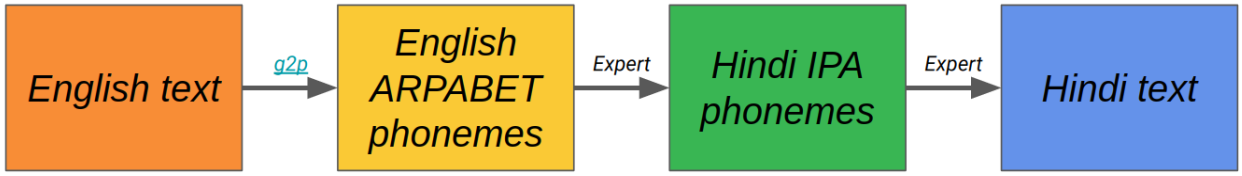

FIGURE 6.1: Phoneme-based transliteration

Figure 6.1 depicts our transliteration technique.

First, the input English text is converted to English ARPABET phonemes using an off-the-shelf grapheme-to-phoneme converter, g2p [106]. The ARPABET is an alphabet used to represent English phonemes; this alphabet was used to represent phonemes in the well-known CMUdict system [107].

Second, the English ARPABET phonemes are converted to Hindi IPA phonemes using an expert mapping. In this expert mapping, some vowels have multiple mappings; in this case, any one mapping is chosen uniformly at random. For stops, we use the aspirated phone when it appears in the word-initial position, otherwise the unaspirated phone.

Finally, the Hindi IPA phonemes are converted to Hindi text using an expert mapping and a rule-based algorithm as follows:

- Convert all consonant phonemes to consonant graphemes using the mapping. For each consonant grapheme, if it is followed by nothing (end-of-word) or by an unresolved phoneme(so a vowel), keep as-is. If it is followed by a consonant, insert a *halantya*, the half-letter modifier.
- Convert all vowel phonemes to vowels graphemes as follows: If it is preceded by nothing or by an unresolved phoneme(so a vowel), use the mapping's vowel as the conversion. If it is preceded by a consonant, use the mapping's diacritic.
- Convert the following type of substring : *nasal consonant* followed by *halantya* to a single character, the *anuswaar*.

Chapter 7

Conclusion

Developing less data-intensive techniques for low-resource languages is an important, socially impactful research problem that demands new approaches. We worked on four projects for improving speech recognition for such languages. To compensate for the limited data, we explored novel ways of using data from high-resource languages, using linguistic attributes of different low-resource languages, and using multilinguality and phonemic ideas to share knowledge across languages. We also publicized this line of research and encouraged participation via an Interspeech Special Session. Having tested our approaches on several Indian languages, we observed significant improvements over baseline systems and gained insights into the inner workings of these algorithms.

Appendix A

Paper Notes

In this chapter, we present detailed rough notes that have been made on papers referred to for some of our aforementioned research work as well some unrelated papers that were read when the project topic was being finalized.

A.1 Papers related to 'Low Resource ASR: The surprising effectiveness of High Resource Transliteration'

A.1.1 wav2vec: Unsupervised Pre-training for Speech Recognition [108]

1. Unsupervised pretraining via learning representations of raw audio. Trained on unlabeled audio data.
2. System components:
   (a) Architecture: CNN that converts audio to a general representation to be fed to an ASR system.
   (b) Objective: Contrastive loss that simply distinguishes a true future audio sample from negatives i.e. contrastive loss for next time step prediction task. Negatives are sampled from the same sequence using empirical evidence that sampling from other places gives inferior results.
   (c) Encoder network: Audio signal to latent space. Heavy hyperparameter optimization for its hyperparameters.
   (d) Context network: Combines multiple steps to get contextualized representations. The final receptive field is 210 ms.
   (e) Causal convolution: Convolution using only prev timesteps. Trick: Use dilated convolutions in layer i+1 to account for strides in layer i.

A.1.2 vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations [109]

1. Idea 1: Discrete units via autoencoding sometimes coupled with an autoregressive model. Idea 2: Learn continuous speech representations by predicting context.
2. Here they combine the two: Learn discrete representations not by reconstruction the input like in autoencoder, but context prediction task.
3. Pipeline for vq-wav2vec: X(raw audio) -> Z (dense representation) -> $\hat{Z}$ (quantized representation) -> C(context representation). Z -> $\hat{Z}$ is a single-timestep transformation. Others are aggregations.
4. Acoustic models are built by quantizing the audio followed by BERT. These representations are then fed into the acoustic model.

5. To choose discrete: Gumbel softmax as well as online k-means like VQ-VAE.

6. VAE: Probabilistic model for learning a graphical model for a data distribution using a simple graphical model $Z \longrightarrow X$. Process is: Draw z from $p(z)$, then x_i from $p(x|z)$. Prior on Z is a Gaussian. We have a likelihood model $p(x|z)$ that characterizes the generation process. This is a generative network (decoder). For a black-and-white image, perhaps a Bernoulli distribution. So now we want to learn the 'true' posterior distribution of z. $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$. Straightforward but intractable due to computation of $p(x)$. So instead, learn an inference network $q(z|x)$ that approximates $p(z|x)$. How to learn? Minimize KL divergence between $p(z|x)$ and $q(z|x)$. Loss function turns out to be autoencoder negative log likelihood of the pipeline to reconstruct the input data well, plus a regularizer that pushes the latent variable approximate posterior $q(z|x)$ to be close to the prior $p(z)$.

7. VQ-VAE: Vector quantization: There are k vectors in dictionary (these are learned as centres of k clusters as we can imagine) and we find the one with the closest distance (arg min). This makes it non-differentiable so we copy gradients from z_q to z_e. A fully-differentiable way would use Gumbel softmax to soften arg min(opp of arg max) with softmax.

A.1.3 wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations [19]

1. Phoneme recognition is harder than word recognition due to lack of dictionary. Self-supervised learning (first learn general representation on unlabeled data followed by finetuning on labeled data). Great for NLP, active for CV.

2. Encode via CNN, then a mask like Masked Language Modelling, then feed to Transformer to get a contextualized representation trained via contrastive loss, similar to wav2vec.

3. CNN-encoded latent representations are first learned into discrete linguistic units via Gumbel softmax to quantize them before feeding into the contrastive task with Transformer. This is more effective than non-quantized. This is somewhat like converting speech into discrete 'speech' units, much like phonemes.

4. Gumbel distribution : $e^{-(x+e^{-x})}$. Gumbel distribution can be used to sample from discrete distribution (when you need to sample it for purposes of optimization function). You have an arg max at the end through which gradients cannot backprop. So instead of arg max, use a soft approximation: softmax with temperature! A good reference is at `https://anotherdatum.com/gumbel-gan.html`.

5. System information:

   (a) Model: Feature encoder (X -> Z), transformer (Z -> C), quantization module (Z -> Q) to represent targets in self-supervised objective.

   (b) Feature encoder: Several CNN blocks containing temporal convolution followed by GELU.

   (c) Transformers: Output of previous is fed to transformer. Positional embedding is using a convolutional layer.

   (d) Straight-through estimator: In case of a threshold function t, suppose you want to estimate incoming gradient for $f(x)$ using outgoing gradient of $t(f(x))$. This cannot be done as the gradient for t is 0 everywhere. Then, you simply set incoming grad to outgoing grad (as if the threshold was an identity function). This works well apparently.

(e) Quantization module: l_{g_v} are learned probabilities for a given z. These probabilities are used to sample from the discrete distribution (done fully differentiably using the standard Gumbel softmax trick). $-\log(\log(u))$ is the inverse-CDF way of sampling from Gumbel distribution.

6. So this is concisely, a way to get quantized representations, but the representations have a 'product' structure: you have G codebooks each with V entries that contribute part of the final vector. This is like a discrete building up of a feature vector (like each codebook corresponds to a categorical feature in some way). Everything after that is just a fully differentiable way to do this sampling.

7. Training: Mask a proportion of feature encoded latent timesteps before feeding to context network. Objective is to identify the correct quantized latent representation from a set of distractors for each masked timestep using contrastive loss. Recall that wav2vec did the same but 1) not quantized and 2) for k steps into future. The current approach is more similar to MLM like in BERT. Finally, you get some contextualized representations. You then finetune on labelled data.

8. In Masking, replace the masked vectors with a 'trained feature vector' shared between all timesteps. This is different from BERT's masked tokens, which are replaced by a special [MASK] token. Also here since these are audio frames, there is a masking span (a range of frames are masked).

9. $L = L_m$(contrastive task) $+ \alpha \times$ [diversity-loss] $+ \beta \times$ [L2 on feature encoder]

10. Contrastive loss: Distractors are sampled from other masked timesteps of same utterance. It is simply a softmax over cosine similarity logits, and we want to maximize the value of softmax for the true example. Diversity loss: Make codebook relevant. For a batch, get average distribution (just avg), and maximize its entropy.

11. Finetuning: Add a single linear projection on top. Loss is now a CTC loss, and they apply SpecAugment, an augmentation method for ASR.

A.1.4 Unsupervised Cross-lingual Representation Learning for Speech Recognition [14]

1. 'Unsupervised pretraining transfers well across languages' [110] freezes all pretrained representations. Also, there, only English was used to pretrain. Here, multiple languages are used. Can also finetune on multiple languages at the same time.

2. 100 distractors in XLSR and wav2vec2.0, as opposed to 10 distractors in wav2vec.

3. Multilingual batches are formed by sampling speech samples from multinomial distribution $(\frac{n_l}{7}N)^(\alpha)$. n_l is no of hours for language l, and α will control importance to high-resource.

A.1.5 An embarrassingly simple approach to zero-shot learning [111]

1. Features -> Attributes -> Classes are a two-linear-layer network, where the last transformation is not learned but provided by prior knowledge. The paper shows bounds on generalization error of such approaches.

2. Transfer learning: Extract knowledge from source tasks applicable to future tasks. Domain adaptation vs. transfer learning: In transfer learning, input distribution (distribution of images for eg. i.e. train-test sets) is same while tasks i.e. labelling functions are different(classification, segmentation). Domain adaptation assumes task is same but distributions of dataset are different.

A.1.6 Towards Zero-shot Learning for Automatic Phonemic Transcription [112]

1. 0 training data languages! Difficulty is phoneme inventories differ between training and target languages, so unseen phonemes are hard. They apply zero-shot learning here.
2. Explored in: Unsupervised speech processing that learn universal representations. These don't produce phonemes directly, however, so the same problems remain.
3. Universal Phonemic Model (UPM). Decompose phoneme into attributes (vowel, open, etc.) and predict over these attributes. Then, unseen phonemes can be inferred using these known attributes.
4. Architecture:
   (a) X(frames) -> H(acoustic space) (standard acoustic model)
   (b) H -> P(attribute space): Gives info about attributes
   (c) P -> phonemes using signature matrix S describing attributes vs phonemes.
5. X-SAMPA is the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA). Articulatory attributes for eg. manner and placement of articulation are well studied in articulatory phonetics.
6. The X-SAMPA format is a computer readable(ASCII) repr for IPA. As a result, each IPA segment (base, diacritic, etc.) is written out in different parts of the X-SAMPA format one after the other. Thus, a pre-known $P_{\text{base}} -> 2^{A_{\text{phone}}}$ mapping is used and all others are broken up into segments greedily suffix-wise.
7. Digression: In zero-shot translation, target language pair not in training dataset.
8. They use the ESZSL from Romera-Paredes and Torr.
9. Here they simply multiply with S, since all attributes contribute to the phoneme, rather than in the Allosaurus paper [37] where a single phone contributes to the phoneme.
10. Uniform sampling: Randomly choose corpus, then randomly choose batch.
11. For comparability, they use a shared phoneme multilingual baseline with the same LSTM architecture.
12. Substitution error rate shows UPM is better at resolving confusion. Ablation on number of training languages. A bit qualitative but useful nonetheless.

A.1.7 Attention Is All You Need [113]

1. Dispense with RNNs and CNNs entirely, use just attention. Superior in quality, more parallelizable, less time to train.
2. BLEU score: A score proposed by Papineni et al. 2001 for machine translation. Position-independent n-gram comparisons.
3. Batch norm: BatchMean-shift and BatchVariance-scale a layer's activations (batch=minibatch) and allow SGD to recover original mean and variance using trainable mean, variance(bias, gain) parameters. Can't use easily in RNN- need different μ, σ for diff timesteps. Also online learning.
   Alternatives: weight norm, layer norm.

- Weight norm: Reparametrize weight as $w = \frac{g*v}{||v||}$
- Layer norm: Normalize across features/hidden units, for each example. There are trainable bias, gain parameters(same for each example).

4. Encoder: Pretty clear from paper.
5. Decoder: Masked self-attention, with shifted outputs, so that output position predictions depend only on previous output positions.
6. Scaled dot-product attention: $Q, K = d_k, V = d_v$. $Q = n_q * d_k, K = n_k * d_k, V = n_k * d_v$. First find dot product of queries with keys: QK^T. Divide by $\sqrt{d_k}$ (scaled by dimension). So we have $n_q * n_k$. Optional masking to remove some of these n_k positions. Softmax along $\leq n_k$, to get weights on values.(If masked, remaining weights are 0). Now multiply this $n_q * n_k$ weight matrix with V, the value matrix to get final outputs of dimension $n_q * d_v$, as desired. Scaling is to avoid dot product from getting too large in magnitude, causing softmax problems. This is similar to length normalization.
7. So here the compatibility function is a dot product i.e. a matmult. In additive attention, it is a feed-forward NN.
8. Multihead attention: Project the d_{model} k, v, q into d_k, d_k, d_v dimensions h times using (learnable) linear projections, apply attention, concatenate to get $h * d_v$ dimensional output and project again onto d_{model}. Jointly attend to different (learned) representations at different positions.
9. Output and input embedding tying is interesting. Output embedding is shown to be better than input embedding!

A.2 Papers related to 'Bridging Scripts by Grounding in Speech'

A.2.1 Low-Resource Machine Transliteration Using Recurrent Neural Networks [114]

This is a good paper that learns transliteration using RNNs given a small parallel pronunciation dictionary, geared towards low-resource langs. The related work section explains the Transliteration task itself well and provides a great literature review as well.

A.2.2 Efficient Neural Machine Translation for Low-Resource Languages via Exploiting Related Languages [115]

Although an NMT paper, this paper uses an important concept we've been talking about: Using a common Devanagiri script for all Indo-Aryan languages! They use this common script for all langs and then train NMT systems (Langs are still differentiated using <GU>,<HI> tags, but the script used is always Devanagiri.)

A.2.3 Transliteration for Cross-Lingual Morphological Inflection [69]

A technique very similar to our transliteration paper, for Morphological Inflection rather than ASR! Here, they explore explicitly placing the transfer (source) and test (target) language into a common space using two ideas: 1) Transliterating source to target lang 2) Converting both to a common alphabet (IPA). Very relevant work that can be directly transferred to an ASR setting. Also, we now have proof that the idea works for a) a different task b) Non-Indian languages. As a follow-up, we can explore usage of this idea to more tasks and languages and demonstrate it is a very useful drop-in replacement in ANY pretraining step!

A.2.4 Can Multilingual Language Models Transfer to an Unseen Dialect? A Case Study on North African Arabizi [116]

Not immediately related to our main line of work, but a nice paper about transferring Multiling. LMs (they use mBERT) to a non-standard dialect of Arabic spoken in Algeria, written in Latin script with lots of code-mixing with French. I like this paper because Narabazi has parallels with Hinglish (non-standard dialect of Hindi written in Latin script with lots of code-mixing with English).

A.2.5 Transliteration Based Data Augmentation for Training Multilingual ASR Acoustic Models in Low Resource[5]

1. Given N languages with labelled speech data, train a standard multilingual ASR system with language-specific output layers as follows:
   (a) Train using aggregated CTC loss, where aggregate is the sum of CTC losses at each language output layer. (*My comment:* For an utt in lang A, only that language's layer is used for guiding the loss? Since other layers cannot be supervised by anything?)
   (b) Train using soft-forgetting ('twin loss'). Train using 'guided' loss (all past papers by same authors)
2. This training scheme is automatically assumed by the authors to make it so that now when you supply an utterance in language A, output of language A layer is transcription of utterance (reasonable) and outputs of language B,C,D layers are transliterations of utterance. (*My comment:* Why? Nothing explicitly does this! It only hopes that this happens due to the common acoustic representation and the absence of LM. Is this enough? This will easily fail when one of the languages are lower resource than the others, since the 'common acoustic representation' isn't actually 'common'?
3. Thereafter this network is used to do a data augmentation and then this same network is further trained with the augmented data (as far as I could understand). (*My comment:* Using output from network to train same network? This is potentially harmful (not using ground truths)).
4. The following experiments are done: Given 4 training langs, comparing monolingual systems vs. normal multilingual system vs. multilingual system built using this approach. Improvements are marginal; filtering out augmented data using their metrics makes the improvements significant enough. They use LSTMs. Further, they use 4 layers LSTM for the monolingual system and 6 layers for all other systems ('since more data is available'), which is pretty unfair to the monolingual systems.
5. Use unlabelled data/ data from other languages by passing it through the previously trained multiling network and use the output as training data.

A.3 Papers related to 'Reduce and Reconstruct: Improving Low-resource End-to-end ASR'

A.3.1 A spelling correction model for end-to-end speech recognition [86]

1. The LM inside an end-to-end model is trained only on transcriptions and so it is bad on rare words. Indeed people have used external LMs to augment E2E framework(top-n rescoring using external LM), but this paper explicitly corrects for the error made by the E2E model by following it with a spelling correction model.

2. The paper uses text-only data to generate audio signals using TTS (this is compared to back-translation in machine translation). Then it runs the LAS speech recognizer on this to get predicted text-ground truth pairs. Then the SC model is trained on this. Outperforms simple LM rescoring and direct LAS model training on synthetic data.

3. Baseline: LAS enc-dec with attention using a stack of convolutional and LSTM layers. The output is wordpiece units.

4. The SC task is simpler than (unsupervised) LM because it simply needs to correct the LAS errors: so it can harness the LM capacity(or assume that the LAS model already has quite some LM capacity).

5. Training set for SC: We need parallel text consisting of LAS hypotheses and ground truths. For each sentence y_i, TTS utterance u_i is produced and N hypotheses $H_{i1}...H_{iN}$. All hypotheses are used(random sampling of one hypothesis).

6. SC model: Attention based enc-dec seq2seq(so a transformer).

    - Embedding: Wordpieces.
    - Encoder: Stack of bidirectional LSTMs with dropout. Attention is multi-headed additive attention(using the key, value as the enc output, query as the hidden cell corr to the LSTM after target output).
    - Decoder: Target is sent through a forward LSTM(used for the attention), then a stack of forward LSTMs, finally a softmax for probabilities. Attention is fed to all LSTM layers and the softmax.
    - General: Residual connections are applied, per-gate layer normalization(Layer norm: Normalize across features/hidden units, for each example. There are trainable bias, gain parameters(same for each example)) applied within LSTM cells.
    - Loss: MLE for prob of ground-truth output given input.

7. Inference: Use N hypotheses from LAS and M hypotheses for each from SC to get $N * M$ hypothesis. Use an LM to rescore and find $A* = \underset{\mathbf{A}}{\text{argmax}} \lambda_{LAS} * p_i + \lambda_{SC} * q_{ij} + \lambda_{LM} * r_{ij}$

8. External LM is the best; (LAS->SC-MTR) $>$ (LAS->SC) $>$ (LAS-TTS) $>$ (LAS) . Adding LM always improves it.

9. Attention weights are monotonic; whenever error occurs attention starts focusing on surrounding context.

10. Oracle WER for n-best lists : The lowest WER out of all the hypotheses in the n-best list is taken to represent the WER of the list. This is of course a lower bound that is achievable only if an 'oracle' picks the best alternative every time.

11. Oracle WER is significantly decreased when using expanded n-best list. Thus, LM rescoring on expanded list is motivated.

12. Since TTS != real data, TTS decoding was found to be have improved much more than real audio decoding when the SC model was applied. Thus, SC model seems to be correcting mostly TTS-like LAS errors and not exactly the LAS errors as we had hoped for. So adding noise to TTS is something the authors tried.

A.3.2 Correction of Automatic Speech Recognition with Transformer Sequence-to-sequence Model [87]

1. 'Translates' corrupted ASR to the correct language. Outperforms 6-gram LM rescoring and nearly matches performance of Transformer-XL LM.
2. Regularisation is key to make this network work:
   (a) Data augmentation using perturbed outputs of many ASR models trained on K-fold partitions.
   (b) Pretrain corrector Enc and Dec with BERT weights.
3. Baseline: Jasper DR. This is a conv E2E model with CTC loss.(Interestingly, output of each block is added into the input of all following blocks like DenseNet). Just like in Jasper paper, LM Used for rescoring are a 6-gram KenLM and TransformerXL. KenLM is shallow fusion(sum of log probabilities), TransformerXL additionally rescores KenLM.
4. Correction model is transformer with dropout.

A.3.3 Automatic Spelling Correction with Transformer for CTC-based End-to-End Speech Recognition [88]

1. This paper uses an NMT Transformer architecture by using ASR hypotheses as input on the source language and ground truth transcriptions on the target language, Mandarin.
2. Doesn't use any extra text only dataset. Instead, the training set transcriptions are used, but they try using threshold-based data expansion method for CTC-based acoustic model with greedy search. They also call it n-best data expansion.

A.3.4 Phone merger specification for multilingual ASR: the Motorola polyphone network [89]

This work is a multilingual ASR system that merges phones from across languages. Specifies merger of mergers of phonetically distinct, but phonologically non-contrastive phones. It tries to develop mergers using language-independent phone features.

A.3.5 Structured redefinition of sound units by merging and splitting for improved speech recognition [90]

This work attempts to redefine the phoneme set used to train ASR models by devising a merge-and-split algorithm, where the phoneme set is merged and split in a data-driven way so as to maximize the likelihood of the training data. This new transformation is applied to the original phoneme set to produce a new one that can be used for training. It uses a Monte-Carlo distance metric to measure closeness of phones.

A.3.6 Phone Merging For Code-Switched Speech Recognition [91]

This paper considers merging phonemes of two languages for improving code-switched speech recognition. It tries to do this by both manually merging similar phonemes as well as using a data-driven approach. In the data-driven approach, they propose context sensitivity to avoid spurious phone swaps, and also consider probabilistic merging (decoder decides whether to merge). They also show why although in their experiments manual merging is the best, it is not optimal.

A.3.7 Language-Agnostic Multilingual Modeling [92]

1. This paper attempts to improve upon a language-dependent multilingual model by using a language-agnostic multilingual model. It transliterates sentences in all 4 languages to a single writing system (Latin) using a pair language model trained on pairs like (Hindi word) : (Latin word) . Uses the Latin graphemes to train the multilingual model (so only 44 targets as opposed to 454 targets if we don't merge Indian language graphemes into Latin). Didn't find out whether they are able to transliterate back into the Indian language (seems not).

2. So it doesn't seem to be merging similar sounding graphemes into one grapheme like us. Additionally it spends no effort reconstructing into the original grapheme set.

A.4 Papers related to 'Phoneme-based Multilingual ASR'

A.4.1 Universal Phone Recognition with a Multilingual Allophone System [37]

1. There is a fundamental difference between phones (language independent units of sounds) and phonemes (lexical constrasts in a particular language). Identically annotated phonemes in two languages can correspond to different phonetic realizations. However, using phoneme transcriptions as-is is what most models do, which is a simplification.

2. This paper models language independent phones and language dependent phonemes separately.

3. Language-independent phone recognition implies that one can build a universal phone recognizer.

4. PHOIBLE: repository of cross-linguistic phonological inventory data. Allophone: One of a set of multiple possible spoken phones to produce a single phoneme. When only a single allophone works in a given context, they are called positional variants, but some occur in free variations.

5. The universal recognizer 'Allosaurus' can be customized into language-dependent recognizers.

6. Most multilingual systems (apart from few new ones like XLSR) depend on phonetically inspired acoustic models.

7. Shared phoneme models suffer from the oversimplification described before that doesn't consider the disconnect across langs.

8. Private phoneme models have a shared encoder but perform phoneme classification separately. This ignores cross-lingual phonetic associations and not applicable to new languages.

9. Allosaurus: Encoder produces distribution over universal phone set. Allophone layer(weights where 1 means allophone mapping and 0 means none) is used to do max pooling to transform it into phoneme logits for each language. Initialized with signature matrix but trainable, L2 penalty to stay near the signature matrix.

10. Max-pooling necessary due to definition of allophone layer. Alternatively we could consider the allophone layer to be a probability distribution, perhaps initialized with a uniform distribution for the known allophones and one could do average pooling instead. But this may have been a bad initialization since we only know the mappings, not the degree/confidence of mappings. So the max pooling approach sounds nice, actually. Also, it allows one to find a one-one phone-phoneme mapping for a given utterance, which would not have been possible for avg pooling.

11. This approach can be used with any architecture. Here they use a simple CTC loss. Phone recognition is for any language in the world, and hence phoneme recognition is doable to some degrees of accuracy using PHOIBLE.
12. The training set is built from phonetically diverse languages. Both read speech and spontaneous speech data is used for robustness.
13. Epitran is a tool to transliterate graphemes into IPA phonemes.

A.5 Other Papers

A.5.1 Unsupervised Learning of Spoken Language with Visual Context [117]

1. Older work: Used word boundary detection and object detection, embedded words and objects, and match the word to region. Now, eschew all this and directly compute similarity scores for entire audio and region.
2. Data collection: $\geq$ 8 words as given by Google ASR. For analysis, used Kaldi to train speech recognizer using WSJ recipe to force align.
3. Multimodal network : two branches, image one uses VGG-16, caption one uses CNNs. Inputs are fixed size: image is centre cropped, audio is truncated/zero padded to either 10 or 20 s. Two embeddings are generated.
4. Loss function is easy. 3 scores are used in each item of the summation. Good results.

A.5.2 Learning Word-Like Units from Joint Audio-Visual Analysis [118]

1. Discovery of word-like acoustic units in speech and associate with images. No standard ASR used.
2. $O(n)$ due to second modality as opposed to self-comparing $O(n^2)$ algorithms.
3. Multimodal models - joint models from different domains.
4. Most structure exactly same as Harwarth et al. 2016. Only difference: now being applied locally.
5. To find where to locally apply the networks, the image is divided into a grid, and all image crops whole boundaries lie on grid lines are used. To keep the number of such crops under control,
   (a) grid size is fixed to $10 * 10$
   (b) aspect ratio should be between 2 : 3 and 3 : 2 and
   (c) minimum bounding width and height are 30% of orig image. Thus, too small, and too narrow crops are avoided.
6. For audio, a 1-dimensional grid is used along time axis(in the spectrogram).
   (a) grid size is fixed to 10 pixels
   (b) segment length should be between 50 and 100 frames i.e. 0.5 to 1 s. When learning groundings, 10 s duration constraint as in training is not imposed.

7. A voice activity detector is used to discard groundings that contain silence. To alleviate the issue of overlapping segments at the top of the list, as you go down the list, the interval IOU of its segment against all segments is computed. If there's overlap above 0.1, it is discarded.
8. k-means clustering is also done to find affinities for analysis.
9. Usage of text transcriptions improves the recall as expected but not by much, as the paper claims.
10. Acoustic segments are associated with words by force-aligning Google recognition text to audio and finding which words overlap with the segment in time.
11. Tradeoff: As k increases, purity increases(more specialized clusters) but coverage decreases.
12. Different clusters capturing same label close neighbours: distances highly discriminative across word types. Also semantically similar clusters are close together, indicating the space captures semantic information.
13. To figure out dependence on the VGG network: the classes from ILSVRC2012 were derived from WordNet synsets. Acoustic cluster labels are looked up in WordNet and all its synonyms(from WordNet) are compared to every ILSVRC2012 synset using a path similarity measure in WordNet. Hyponym : type-of Hypernym.

A.5.3 Learning Hierarchical Discrete Linguistic Units from Visually-Grounded Speech [119]

1. ZeroSpeech 2019: Detect subword unit embeddings and generate speech from these embeddings.
2. There exist speech synthesizers that work on subword units or other kinds of embeddings, which is cool.
3. Using Gumbel-softmax to use discrete variables directly in neural networks: interesting idea!

A.5.4 Optimization methods

1. Momentum: Moving average of gradients. Use this for the update. Akin to rolling a ball down a hill with air resistance($\gamma < 1$). Momentum has less 'air resistance'($\gamma > 0$) compared to SGD.
2. Nesterov accelerated gradient: Momentum may go too fast due to large gradient. Calculate gradient wrt $\theta - \gamma_{t-1}$ i.e. at the approximate future location of our parameters. Improved RNN performance.
3. Now we have updates that understand the topology better. Now tweak parameter-wise.
4. Adagrad:Small updates for frequently occurring features, large updates for infrequently occurring features. Divide LR for each feature such that if past sum of gradients was small, have larger updates.
5. Adadelta: Adagrad has a monotonically decreasing LR. To fix that, use a decaying average of squared gradients rather than full sum of squared gradients. Instead of a learning rate, a decaying RMS of past parameter updates is used! This is to ensure units are the same($g_t / \text{RMS}[g]_t$ = unitless $\times \eta$ = unitless, so replace η with $\text{RMS}[\Delta_\theta]_{t-1}$)
6. RMSProp: Adadelta but only the first fix.

7. Adam: Adaptive Moment Estimation. Adam stores exp dec avg of past squared gradients and past gradients. Adam is like a heavy ball with friction: it prefers a flat minima. Hence m_t is approx mean and v_t is approx variance of gradients. Bias-correct these(due to initial zero initalization) then update according to RMSProp(replace gradient with avg of gradients).
8. AdaMax: Instead of squared gradients(L_2), use norm L_p. L_∞ works well, hence this theory. $\max(\beta_2 * \text{old average}, \text{new value})$ as opposed to a weighted sum.
9. Nadam: Adam + NAG.
10. AMSGrad: Sometimes adaptive LR methods are beaten by SGD with momentum. Use max of prev and new avg rather than just new avg.

Bibliography

[1] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The State and Fate of Linguistic Diversity and Inclusion in the NLP World," *arXiv preprint arXiv:2004.09095*, 2020.

[2] S. Khare, A. Mittal, A. Diwan, S. Sarawagi, P. Jyothi, and S. Bharadwaj, "Low Resource ASR: The Surprising Effectiveness of High Resource Transliteration," in *Proc. Interspeech 2021*, 2021, pp. 1529–1533. DOI: 10.21437/Interspeech.2021-2062.

[3] 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, and K. Sankaranarayanan, "MUCS 2021: Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages," in *Proc. Interspeech 2021*, 2021, pp. 2446–2450. DOI: 10.21437/Interspeech.2021-1339.

[4] A. Diwan and P. Jyothi, "Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages," in *Proc. Interspeech 2021*, 2021, pp. 3445–3449. DOI: 10.21437/Interspeech.2021-644.

[5] S. Thomas, K. Audhkhasi, and B. Kingsbury, "Transliteration Based Data Augmentation for Training Multilingual ASR Acoustic Models in Low Resource Settings," in *Proc. Interspeech 2020*, 2020, pp. 4736–4740. DOI: 10.21437/Interspeech.2020-2593. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-2593.

[6] Y. Tsvetkov, "Linguistic knowledge in data-driven natural language processing," Ph.D. dissertation, 2016.

[7] T Schultz and A Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1-2, pp. 31–51, 2001.

[8] A Ghoshal, P Swietojanski, and S Renals, "Multilingual training of deep neural networks," in *ICASSP 2013*, IEEE, 2013, pp. 7319–7323.

[9] 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, pp. 7304–7308.

[10] S Tong, P. N. Garner, and H Bourlard, "Multilingual training and cross-lingual adaptation on CTC-based acoustic model," *arXiv:1711.10025*, 2017.

[11] X Li, S Dalmia, J Li, M Lee, P Littell, J Yao, A Anastasopoulos, D. R. Mortensen, G Neubig, A. W. Black, *et al.*, "Universal phone recognition with a multilingual allophone system," in *ICASSP*, 2020, pp. 8249–8253.

[12] J Kunze, L Kirsch, I Kurenkov, A Krug, J Johannsmeier, and S Stober, "Transfer learning for speech recognition on a budget," *arXiv preprint arXiv:1706.00290*, 2017.

[13] 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 *ICASSP 2018*, IEEE, 2018, pp. 4904–4908.

[14] A Conneau, A Baevski, R Collobert, A Mohamed, and M Auli, *Unsupervised Cross-lingual Representation Learning for Speech Recognition*, 2020. arXiv: 2006.13979 [cs.CL].

[15] A Datta, B Ramabhadran, J Emond, A Kannan, and B Roark, "Language-Agnostic Multilingual Modeling," in *ICASSP*, 2020, pp. 8239–8243.

[16] D Wang and T. F. Zheng, "Transfer learning for speech and language processing," in *APSIPA 2015*, IEEE, 2015, pp. 1225–1237.

[17] O Scharenborg, F Ciannella, S Palaskar, A Black, F Metze, L Ondel, and M Hasegawa-Johnson, "Building an asr system for a low-resource language through the adaptation of a high-resource language asr system: Preliminary results," *Proceedings of ICNLSSPo*, 2017.

[18] J. Emond, B. Ramabhadran, B. Roark, P. Moreno, and M. Ma, "Transliteration Based Approaches to Improve Code-Switched Speech Recognition Performance," in *SLT 2018*, 2018, pp. 448–455.

[19] A Baevski, H Zhou, A Mohamed, and M Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[20] I. A. Bhat, V Mujadia, A Tammewar, R. A. Bhat, and M Shrivastava, "IIIT-H System Submission for FIRE2014 Shared Task on Transliterated Search," in *FIRE*, Bangalore, India, 2015, pp. 48–53, ISBN: 978-1-4503-3755-7. DOI: 10.1145/2824864.2824872. [Online]. Available: http://doi.acm.org/10.1145/2824864.2824872.

[21] V Panayotov, G Chen, D Povey, and S Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *ICASSP 2015*, IEEE, 2015, pp. 5206–5210.

[22] 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," Aug. 2018, pp. 11–14. DOI: 10.21437/SLTU.2018-3.

[23] S. L. I. Madras, *Hindi ASR Challenge Dataset*, Jul. 2020. [Online]. Available: https://github.com/Speech-Lab-IITM/Hindi-ASR-Challenge.

[24] O Kjartansson, S Sarin, K Pipatsrisawat, M Jansche, and L Ha, "Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali," in *Proc. of 6th SLTU*, 2018, pp. 52–55.

[25] *Zeroth-korean*, https://www.openslr.org/40/.

[26] S. T. Abate, W. Menzel, and B. Tafila, "An amharic speech corpus for large vocabulary continuous speech recognition," in *INTERSPEECH-2005*, 2005.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, Hilton Waikoloa Village, Hawaii: IEEE Signal Processing Society, Dec. 2011.

[28] S. Virpioja, P. Smit, S.-A. Grönroos, and M. Kurimo, "Morfessor 2.0: Python implementation and extensions for morfessor baseline," en, D4 Julkaistu kehittämis- tai tutkimusraportti tai -selvitys, 2013, p. 38. [Online]. Available: http://urn.fi/URN:ISBN:978-952-60-5501-5.

[29] 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. DOI: 10.21437/Interspeech.2018-1456. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1456.

[30] 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.

[31] 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. DOI: 10.18653/v1/p16-1162. [Online]. Available: http://dx.doi.org/10.18653/v1/P16-1162.

[32] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A. N. Gomez, Ł Kaiser, and I Polosukhin, "Attention Is All You Need," *NeurIPS*, 2017.

[33] J Cho, M. K. Baskar, R Li, M Wiesner, S. H. Mallidi, N Yalta, M Karafiat, S Watanabe, and T Hori, "Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling," in *SLT 2018*, IEEE, 2018, pp. 521–527.

[34] K Heafield, "KenLM: Faster and smaller language model queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland: Association for Computational Linguistics, Jul. 2011, pp. 187–197. [Online]. Available: `https://www.aclweb.org/anthology/W11-2123`.

[35] D. R. Mortensen, S. Dalmia, and P. Littell, "Epitran: Precision G2P for many languages," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. C. C. chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, Eds., Paris, France: European Language Resources Association (ELRA), 2018, ISBN: 979-10-95546-00-9.

[36] S. E. Eden, "Measuring phonological distance between languages" (ucl (university college london)," Ph.D. dissertation, 2018.

[37] X. Li, S. Dalmia, J. B. Li, M. R. Lee, P. Littell, J. Yao, A. Anastasopoulos, D. R. Mortensen, G. Neubig, A. Black, and F. Metze, "Universal phone recognition with a multilingual allophone system," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8249–8253, 2020.

[38] *Ministry of Human Resource Development, India:*, Read on to know more about Indian languages, Online, Last accessed on 03-04-21.

[39] J. Heitzman and R. L. Worden, *India: A country study*. Federal Research Division, 1995.

[40] *Office of the Registrar General & Census Commissioner India, Part A:*, Family-wise grouping of the 122 Scheduled and Non-Scheduled Languages–2001, Online, Last accessed on 03-04-21.

[41] *Office of the Registrar General & Census Commissioner India, Part A:*, Distribution of the 22 scheduled languages - India, States & Union Territories - 2001 Census, , Online, Last accessed on 03-04-21.

[42] H. B. Sailor and T. Hain, "Multilingual speech recognition using language-specific phoneme recognition as auxiliary task for Indian languages," *in Interspeech*, pp. 4756–4760, 2020.

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

[44] P. Auer, *Code-switching in conversation: Language, interaction and identity*. Routledge, 2013.

[45] Ö. Çetinoğlu, S. Schulz, and N. T. Vu, "Challenges of computational processing of code-switching," *arXiv preprint arXiv:1610.02213*, 2016.

[46] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *arXiv preprint arXiv:1904.00784*, 2019.

[47] Y.-C. Chen, J.-Y. Hsu, C.-K. Lee, and H.-y. Lee, "DARTS-ASR: Differentiable architecture search for multilingual speech recognition and adaptation," *arXiv preprint arXiv:2005.07029*, 2020.

[48] S. Tong, P. N. Garner, and H. Bourlard, "An investigation of multilingual ASR using end-to-end LF-MMI," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6061–6065.

[49] H. Lin, L. Deng, D. Yu, Y.-f. Gong, A. Acero, and C.-H. Lee, "A study on multilingual acoustic modeling for large vocabulary ASR," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 4333–4336.

[50] B. Li, Y. Zhang, T. Sainath, Y. Wu, and W. Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5621–5625.

[51] J. Cui, B. Kingsbury, B. Ramabhadran, A. Sethy, K. Audhkhasi, X. Cui, E. Kislal, L. Mangu, M. Nussbaum-Thom, M. Picheny, *et al.*, "Multilingual representations for low resource speech recognition and keyword search," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 259–266.

[52] V Pratap, A Sriram, P Tomasello, A Hannun, V Liptchinsky, G Synnaeve, and R Collobert, *Massively Multilingual ASR: 50 Languages, 1 Model, 1 Billion Parameters*, 2020. arXiv: 2007.03001 [eess.AS].

[53] M. Miiller, S. Stiiker, and A. Waibel, "Multilingual adaptation of RNN based ASR systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5219–5223.

[54] K. Manjunath, K. S. Rao, D. B. Jayagopi, and V Ramasubramanian, "Indian languages ASR: A multilingual phone recognition framework with IPA based common phone-set, predicted articulatory features and feature fusion.," in *Interspeech*, 2018, pp. 1016–1020.

[55] C. Liu, Q. Zhang, X. Zhang, K. Singh, Y. Saraf, and G. Zweig, "Multilingual graphemic hybrid ASR with massive data augmentation," *arXiv preprint arXiv:1909.06522*, 2019.

[56] A. Dey and P. Fung, "A Hindi-English code-switching corpus.," in *LREC*, 2014, pp. 2410–2413.

[57] A. Pandey, B. M. L. Srivastava, and S. V. Gangashetty, "Adapting monolingual resources for code-mixed Hindi-English speech recognition," in *IEEE International Conference on Asian Language Processing (IALP)*, 2017, pp. 218–221.

[58] S. Sivasankaran, B. M. L. Srivastava, S. Sitaram, K. Bali, and M. Choudhury, "Phone merging for code-switched speech recognition," in *Third Workshop on Computational Approaches to Linguistic Code-switching*, 2018.

[59] K. Taneja, S. Guha, P. Jyothi, and B. Abraham, "Exploiting monolingual speech corpora for code-mixed speech recognition," *in Interspeech*, pp. 2150–2154, 2019.

[60] S. Ganji, K. Dhawan, and R. Sinha, "IITG-HingCoS corpus: A Hinglish code-switching database for automatic speech recognition," *Speech Communication*, vol. 110, pp. 76–89, 2019.

[61] B. M. L. Srivastava, S. Sitaram, R. K. Mehta, K. D. Mohan, P. Matani, S. Satpal, K. Bali, R. Srikanth, and N. Nayak, "Interspeech 2018 low resource automatic speech recognition challenge for Indian languages.," in *SLTU*, 2018, pp. 11–14.

[62] *Indian Language TTS Consortium and ASR Consortium*, Indian Language Speech sound Label set (ILSL12), 2016, Online, Last accessed on 03-04-21.

[63] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[64] 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, pp. 2751–2755.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[66] R. L. Weide, "The CMU pronouncing dictionary," *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*, 1998.

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

[68] R. Gretter, "Euronews: A multilingual benchmark for ASR and LID," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[69] N. Murikinati, A. Anastasopoulos, and G. Neubig, "Transliteration for cross-lingual morphological inflection," in *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Online: Association for Computational Linguistics, Jul. 2020, pp. 189–197. DOI: 10.18653/v1/2020.sigmorphon-1.22. [Online]. Available: https://www.aclweb.org/anthology/2020.sigmorphon-1.22.

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

[71] M. Ma, B. Ramabhadran, J. Emond, A. Rosenberg, and F. Biadsy, "Comparison of data augmentation and adaptation strategies for code-switched automatic speech recognition," in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2019, pp. 6081–6085.

[72] C. Liu, P. Jyothi, H. Tang, V. Manohar, R. Sloan, T. Kekona, M. Hasegawa-Johnson, and S. Khudanpur, "Adapting asr for under-resourced languages using mismatched transcriptions," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 5840–5844.

[73] S. Tong, P. N. Garner, and H. Bourlard, "Cross-lingual adaptation of a CTC-based multilingual acoustic model," *Speech Communication*, vol. 104, pp. 39–46, 2018.

[74] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A Fast, Extensible Toolkit for Sequence Modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[75] 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.

[76] 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*, IEEE, vol. 1, 1996, pp. 427–430.

[77] R. Errattahi, A. E. Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," in *Procedia Computer Science*, 2018. DOI: 10.1016/j.procs.2018.03.005.

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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, ISSN: 15206149. DOI: 10.1109/ICASSP.2019.8683745. arXiv: 1902.07178.

[87] 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. DOI: 10.1109/ICASSP40776.2020.9053051.

[88] S. Zhang, M. Lei, and Z. Yan, *Automatic spelling correction with transformer for ctc-based end-to-end speech recognition*, 2019. arXiv: 1904.10045 [eess.AS].

[89] 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.

[90] 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.

[91] 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. DOI: 10.18653/v1/W18-3202. [Online]. Available: https://www.aclweb.org/anthology/W18-3202.

[92] 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. DOI: 10.1109/ICASSP40776.2020.9053443.

[93] 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. DOI: 10.1109/ASRU.2015.7404790.

[94] 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.

[95] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[96] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning Word Vectors for 157 Languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[97] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://www.aclweb.org/anthology/D14-1162.

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

[99] 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, Prague, Czech Republic: Springer-Verlag, 2007, 11–23, ISBN: 354076335X.

[100] 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.

[101] Gaurav, *Gujarati Wikipedia articles*, Dec. 2019. [Online]. Available: https://www.kaggle.com/disisbig/gujarati-wikipedia-articles/version/1.

[102] ——, *Telugu Wikipedia articles*, Dec. 2019. [Online]. Available: https://www.kaggle.com/disisbig/telugu-wikipedia-articles/version/1.

[103] 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. arXiv: 2005.08100 [eess.AS].

[104] R. Aharoni, M. Johnson, and O. Firat, "Massively multilingual neural machine translation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3874–3884. DOI: 10.18653/v1/N19-1388. [Online]. Available: https://www.aclweb.org/anthology/N19-1388.

[105] X. Wang, Y. Tsvetkov, and G. Neubig, "Balancing training for multilingual neural machine translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8526–8537. DOI: 10.18653/v1/2020.acl-main.754. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.754.

[106] K. Park and J. Kim, *G2pe*, https://github.com/Kyubyong/g2p, 2019.

[107] R. L. Weide, *Carnegie Mellon Pronouncing Dictionary*, 1998.

[108] S Schneider, A Baevski, R Collobert, and M Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[109] A. Baevski, S. Schneider, and M. Auli, *Vq-wav2vec: Self-supervised learning of discrete speech representations*, 2020. arXiv: 1910.05453 [cs.CL].

[110] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux, *Unsupervised pretraining transfers well across languages*, 2020. arXiv: 2002.02848 [eess.AS].

[111] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International Conference on Machine Learning*, 2015, pp. 2152–2161.

[112] X. Li, S. Dalmia, D. R. Mortensen, J. Li, A. W. Black, and F. Metze, *Towards zero-shot learning for automatic phonemic transcription*, 2020. arXiv: 2002.11781 [cs.CL].

[113] 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 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[114] N. T. Le, F. Sadat, L. Menard, and D. Dinh, "Low-resource machine transliteration using recurrent neural networks," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 18, no. 2, Jan. 2019, ISSN: 2375-4699. DOI: 10.1145/3265752. [Online]. Available: https://doi.org/10.1145/3265752.

[115] V. Goyal, S. Kumar, and D. M. Sharma, "Efficient neural machine translation for low-resource languages via exploiting related languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Online: Association for Computational Linguistics, Jul. 2020, pp. 162–168. DOI: 10.18653/v1/2020.acl-srw.22. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-srw.22.

[116] B. Muller, B. Sagot, and D. Seddah, *Can multilingual language models transfer to an unseen dialect? a case study on north african arabizi*, 2020. arXiv: 2005.00318 [cs.CL].

[117] D. Harwath, A. Torralba, and J. R. Glass, "Unsupervised Learning of Spoken Language with Visual Context," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, 1866–1874, ISBN: 9781510838819.

[118] D. Harwath and J. Glass, "Learning Word-Like Units from Joint Audio-Visual Analysis," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017. DOI: 10.18653/v1/p17-1047. [Online]. Available: http://dx.doi.org/10.18653/v1/P17-1047.

[119] D. Harwath, W.-N. Hsu, and J. Glass, "Learning Hierarchical Discrete Linguistic Units from Visually-Grounded Speech," *arXiv preprint arXiv:1911.09602*, 2019.