Low Resource ASR: The surprising effectiveness of High Resource Transliteration

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

[1] IBM Research

[2] IIT Bombay

[†] Equal contribution

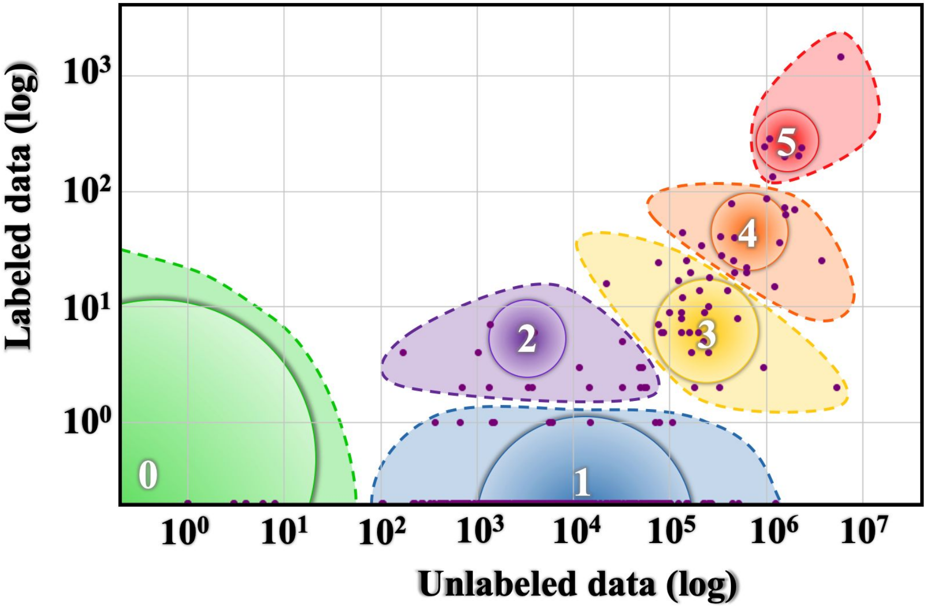
Slides by Anuj Diwan

Introduction

Motivations

Many advances in speech and NLP are powered by availability of data.

Only **high-resource** languages consistently benefit!



Reference:
P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The State and Fate of Linguistic Diversity and Inclusion in the NLP World," in ACL, 2020.

Motivations

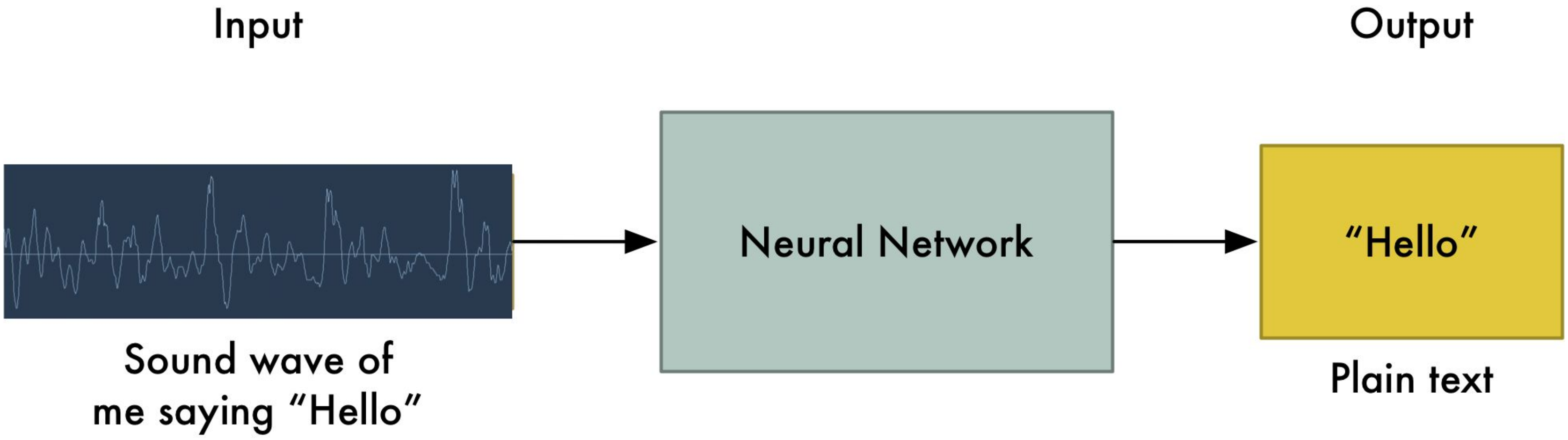
A vast majority of the 7000 languages of the world, including most **Indian** languages, fall in the **low-resource** category.

Techniques for **low-resource** languages need to be *less data-intensive* and often require interesting, radically new approaches.

Reference: https://www.pratidintime.com/80-tribal-languages-of-ne-facing-threat-of-extinction/

Automatic Speech Recognition

Convert an input speech signal to its corresponding transcript.



Reference:
https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a

Low-resource Speech Recognition

Developing **Automatic Speech Recognition (ASR)** techniques for **low-resource** languages.

In this paper, we explore a **Transliteration-based Transfer** approach for low-resource multilingual ASR.

Transliteration-based Transfer

Low Resource ASR: The surprising effectiveness of High Resource Transliteration

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

[1] IBM Research, [2] IIT Bombay

Accepted at Interspeech 2021

Introduction: Transfer Learning

- Using knowledge gained while solving one problem to solve a **different but related** problem.
- Use larger quantities of data from high-resource languages and **transfer** this knowledge to the low-resource language task.

Existing Approach

Pretrain using unlabelled+labelled speech from one (or more) ‘source’ high-resource languages

Learn a general ‘good’ representation of speech

Finetune all/part of model on labelled speech from ‘target’ low-resource language

Given the pretrained model, learn parameters for the specific target language

Existing Approach

What if source and target languages have disjoint grapheme spaces?

English: A B C D E F G H I J K L M ...

Hindi: क ख ग घ ङ च छ ज झ ञ ...

Existing Approach

What if source and target languages have disjoint grapheme spaces?

- Pretrain only the encoder of the encoder-decoder ASR architecture.
- Pretrain both the encoder and decoder. Before finetuning, replace output softmax layer with target language output softmax layer.

Sharing across languages is latent and not easily controllable!

Our Approach

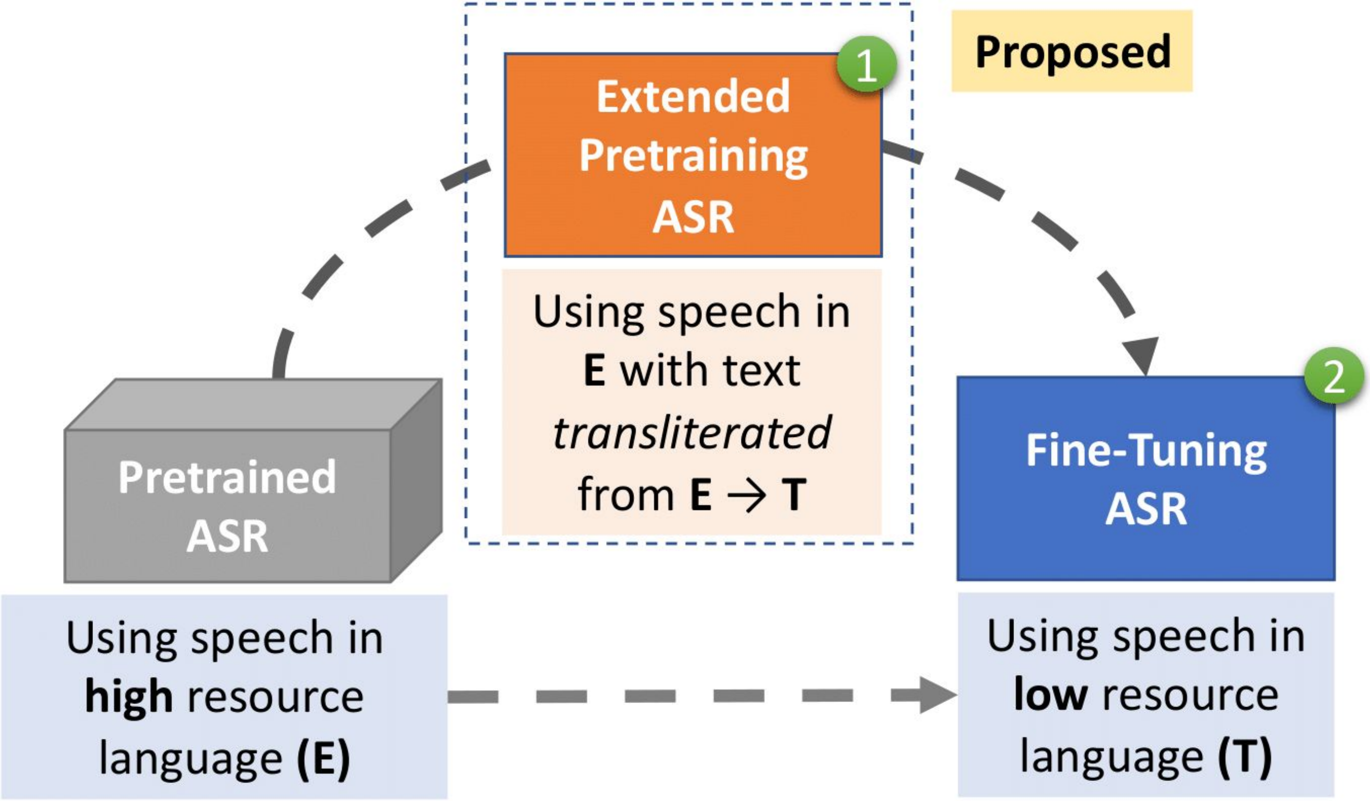
Encourage increased sharing across grapheme spaces.

1. **Transliterate** transcriptions in high-resource speech data.
    - *From* high-resource language
    - *To* low-resource language
2. **Pretrain** model on high-resource language using original audio and *transliterated* transcriptions.
3. **Finetune** model using limited data from low-resource language.

ground -> ग्राउंड
ground -> గ్రౌండ్

*We also call our approach **Eng2Tgt.***

Our Approach



Our Approach: Transliteration

en: ground without overbrimming *ipa*: gɹ'aʊnd wɪð,aʊt ,əʊvəbɹ'ɪmɪŋ
hi: ग्राउंड विदऔत ओवर्ब्रिमिंग *ipa*: gɾa:'ʊɳɖ w,ɪd'ɔ:t ,o:ʋrbɾɪm'ĩg
gu: ગ્રાઉન્ડ વિથાત ઓવર્બ્રીમિંગ *ipa*: gɾa:'ʊnɖ wɪtʰ'a:t ,o:ʋrbɾim'ĩg
bn: গ্রাউন্ড উইথাউত ওভারব্রিমিং *ipa*: gɾ'aunɖ,ɔ 'uitʰ,aut,o 'o:bʱaɾ,ɔbɾim,iŋ
te: గ్రూండ్ విథావుట్ ఒవెర్బ్రిమ్మిన్గ్ *ipa*: gr'u:nɖ ʋ'ita:ʋuʈ 'oʋerbr,imming
ko: 그라운드 위트하우트 오버르브리밍 *ipa*: kɯɾa̠undɯ ɥitʰɯɦa̠utʰɯ o̞bʌ̹ɾɯbɯɾimiŋ
am: ግሮዑንድ ውትሀዑት ኦቨርብርምምንግ *ipa*: gɨrounɨd witɨhout overɨbɨrɨmɨmɨnɨg

We use existing off-the-shelf transliteration systems.
For the 4 Indian languages: [indic-trans]
For Korean: [Microsoft Azure Transliterate API]
For Amharic: [Google Transliterate API]

Experiments

- **Source language:** English
- **6 languages**: Hindi, Telugu, Gujarati, Bengali, Korean, Amharic (more info: BTP Report)
- **2 ASR architectures:** Transformer [1] and wav2vec2.0 [2]
- **2 Training Durations:**
    - Full and 10-hr for Transformer expts
    - 10-hr and 1-hr for wav2vec2.0 expts

Note: 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.

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, .L Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in NeurIPS, 2017.

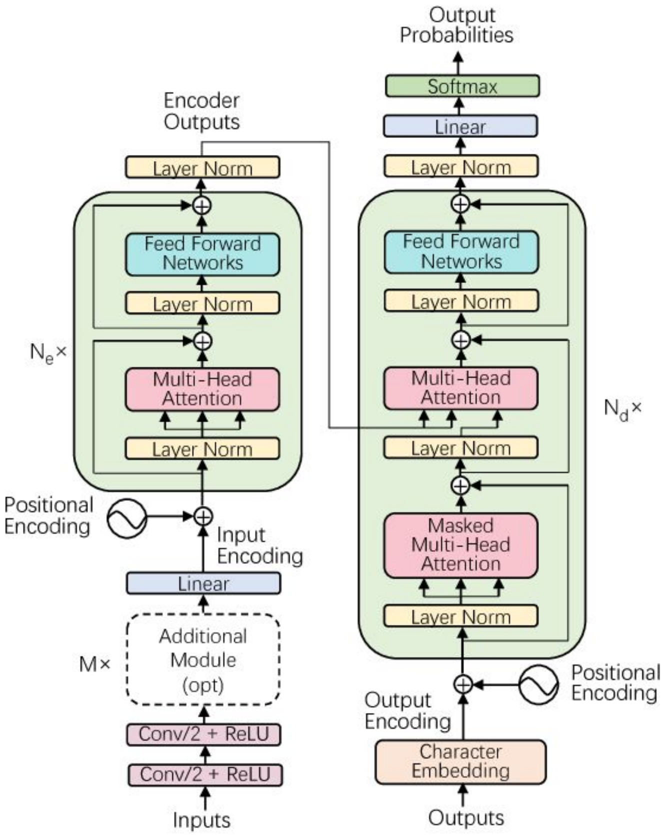
[2] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in NeurIPS, 2020.

Experiments: Baselines

1. **NoPre:** Train from scratch on low-resource data *without* pretraining.
2. **EngPre:** Pretrain using *untransliterated* text from English data, followed by finetuning on low-resource data.
3. **Tgt2Eng:** Based on [3].
    a. Pretrain using untransliterated text from English data
    b. Transliterate low-resource data transcriptions to English (Latin script) and finetune on this data.
    c. This model produces Latin script transcriptions. Thus, finally, transliterate back to low-resource language script.

[3] A. Datta, B. Ramabhadran, J. Emond, A. Kannan, and B. Roark, "Language-Agnostic Multilingual Modeling," in ICASSP, 2020.

Experimental Setup: Transformer



Transformer Architecture for Speech Recognition

We use the ESPNet toolkit to train hybrid CTC-attention Transformers

Major hyperparameters:
12 encoder layers with **2048** units
6 decoder layers with **2048** units
0.3 CTC, 0.7 Attention

More info in the paper

Reference:
L. Dong, S. Xu and B. Xu, "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition," in ICASSP, 2018.

Results: Transformer

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**

Word Error Rate (WER)
for different transliteration schemes
for the **Transformer** architecture

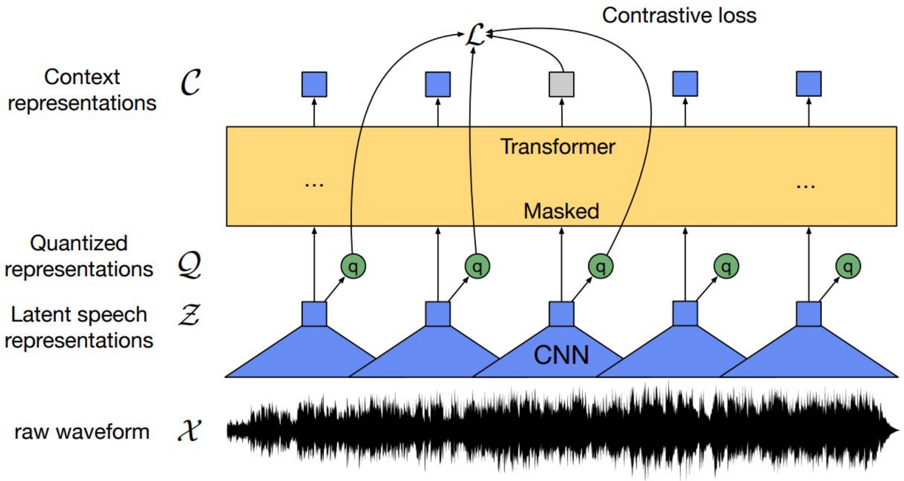
Results: Transformer

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**

- **NoPre** is *worse than* All three methods that use the English corpus
- Our approach is **better than** baselines in most cases
- Larger gains in low-resource **10-hr** setting
- Tgt2Eng is worse than all other methods. Likely due to lossy transliterations:

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

Experimental Setup: wav2vec2.0



Reference:
A Baevski, H Zhou, A Mohamed, and M Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in NeurIPS, 2020.

wav2vec2 Architecture for Speech Recognition

We use the fairseq toolkit

Model architecture and training schedules are according to the wav2vec2.0 paper

Before pretraining, all methods are **initialized** using the wav2vec2.0 model estimated using *unsupervised pretraining* on the complete Librispeech dataset

Thus, the **NoPre** baseline is replaced with the **SelfSup** baseline.

More info in the paper

Results: wav2vec2.0

	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

Word Error Rate (WER)
for different transliteration schemes
for the **wav2vec2.0** architecture.
For Korean, **Character Error Rate (CER)**
also reported in parentheses.

Note: We dropped Tgt2Eng since it fared badly in the Transformer expts

Results: wav2vec2.0

	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

- wav2vec **SelfSup** much better than Transformer **NoPre**
- Our method clearly outperforms EngPre in most settings on all languages
- Major exception is Amharic; we investigate this further

Our approach works even on a SOTA system like wav2vec that leverages powerful pretrained models!

Analysis and Discussions

Under what conditions is our approach most effective?

We propose that **two** properties should simultaneously hold:

- **High acoustic consistency** of the transliteration library
- **High phonological overlap** between the two languages

Analysis: Methodology

1. **Acoustic Consistency of Transliterations:**
   - Convert original English text to IPA (phones) using a g2p tool (epitran)
   - Convert transliterated text to IPA using native-language g2p tools
   - Compute PER between the two IPA sequences
2. **Phonological Similarity between Languages:**
   - Compute unigram distribution of phones in English and in low-resource language
   - Compute KL divergence between the two distributions

Analysis: Results

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

- For Hindi and Telugu, where KL dist is low *and* PER is low, we get consistent improvements in results
- Amharic has a large PER, which may explain its poor performance. However, more investigation is needed, since its KL dist is very low.

Analysis: Effect of Related Languages

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

WERs for Gujarati when pretrained using two approaches:
Hin2Tgt: Pretrain on 40 hrs of Hindi transliterated to Gujarati
Eng2Tgt40: Pretrain on 40 hrs of English transliterated to Gujarati

Pretraining on a related language helps!

Analysis: EngPre vs Eng2Tgt

Our analysis indicates that:

In **EngPre**, pretraining lets the model learn sound clusters, and then the fine-tuning phase is used to learn character labels for each such sound, in addition to learning new sounds which are missing in the English speech data.

In **Eng2Tgt**, the fine-tuning phase focuses more on the second aspect (learning new sounds) as the pretraining phase already attaches character labels to the sound clusters.

Future Work

- Extending this approach to **multilingual** ASR.
- Extending this approach to languages with **no transliteration** systems.

Thank you!