TELUGU AND TAMIL SPEECH RECOGNITION USING TRANSFER LEARNING TECHNIQUES

Sourya Kakarla (UNI: sk5057)

ABSTRACT

This project aims to develop an open-source baseline Automatic Speech Recognition (ASR) pipeline for the Dravidian languages Tamil and Telugu. We propose to use different speech datasets to train time-delayed neural networks. As the total size of usable Telugu speech data (45 hours) from these sources doesn't quite fit the requirement to achieve reasonable performance levels, we investigate the application of transfer learning from another Dravidian language, Tamil for which the data is relatively much more accessible. We present an end-to-end pipeline of taking a raw text corpus for the language model and generating a clean dataset along with a simple tool to verify its quality. Finally, a customized Kaldi recipe is developed by modifying the TEDLIUM one to build the language model and train a time-delayed neural network for the acoustic modeling using cleansed datasets and evaluate the performance of the models. In this evaluation, we also investigate the success of using transfer learning from Tamil and English to Telugu. We aim to contribute to the low-resource open-source ASR world and the general Speech Recognition and NLP ecosystem by publishing the tools and techniques used in this project as a modular library after sufficient testing and iterations.

***Index Terms*—** automatic speech recognition, low-resource languages, time-delayed neural networks, language model, Telugu, Tamil, open-source

1. INTRODUCTION

1.1. Motivation

There is a significant difference in the performance of ASR systems for high-resource languages like English when compared to the multitude of low-resource languages that are spoken all over the world. This is due to various geo-political and socio-cultural aspects of the global society in the 21st century. Having said that, extensive research on natural speech and language speech technologies is being done all over the world for a wide range of such underserved languages. We aspire to contribute this domain, more specifically the speech aspect by building a baseline ASR system for the Dravidian languages, Tamil and Telugu, spoken by 81 million and 93 million people respectively [1].

One of the key aspects of Speech technology is it being a powerful tool to access technology and consequently various social resources and opportunities for the illiterate and semi-illiterate. This is especially significant for a developing nation such as India with lower levels of literacy, especially in rural areas. The South Indian states of Andhra Pradesh and Telangana, for whom the predominant mother tongue is Telugu, have a literacy rates of 72.8% and 66.4% respectively [2]. This is even lower for underserved sections of the population, such as low income sections and rural populations.

Improving the state of the art implementations of ASR for a language like Telugu has immense potential of improving the quality of life for many illiterate and semi-literate. In the 21st century, the digital world has become a necessity to lead a wellrounded life. Inspite of the integration of digital technology and web services in many critical aspects of life in 21st century, there is a predominant bias towards using English in the defacto implementations of systems like Websites, Mobile Apps, IoT devices such as Google Home, Amazon Alexa etc.

To contribute towards bridging that gap, we aim to build open-source baseline ASR systems for Tamil and Telugu and investigate the effects of transfer learning on the performance of the models. Transfer learning has proven to be quite effective in improving performance of models for related and low-resource languages as reported in multiple studies [3, 4, 5]. We take inspiration from these works and aim to investigate using transfer

learning among the Dravidian family of languages and evaluate its performance.

2. STATE OF THE ART

Over the last few years, there has been a surge of Indic language speech recognition studies [6, 7, 8] backed by top organizations in academia and industry. Kaldi basline WERs are reported for Tamil and Telugu by Srivastava et al. [6] for training and test data of 40 hours and 5 hours respectively (Table 1). As more data is available to us, we hope to achieve atleast a better WER than the baseline reported here.

Language	GMM-HMM	DNN	TDNN
Tamil	33.55	25.47	19.45
Telugu	40.12	34.97	22.61

Table 1. Word Error Rates of GMM-HMM, DNN and TDNN (chain) models built using the training data, tested on the released test data [6]

We can see in Fig. 1 the Word Error Rates reported in a study [9] published in Dec, 2021. The models reported seem to be close to the top performing ones in the field. We aspire to achieve performance as close to them as possible given our resource constraints.

	gu	ta	te
Baseline (Srivastava et al. 2018)	19.8	19.4	22.6
Jilebi (Pulugundla et al. 2018)	14.0	13.9	14.7
Cogknit (Fathima et al. 2018)	17.7	16.0	17.1
CSALT-LEAP (Srivastava et al. 2018)	-	16.3	17.6
ISI-Billa (Billa 2018)	19.3	19.6	20.9
MTL-SOL (Sailor and Hain 2020)	18.4	16.3	18.6
Reed (Sen et al. 2021)	16.1	19.9	20.2
CNN + Context temporal features (Sen et al. 2020)	18.4	24.3	25.2
EkStep model*	19.5	22.1	21.9
M5:	**11.7**	**13.6**	**11.0**
M6:	12.3	15.1	12.4

Fig. 1. WER comparison of the best models (M5, M6) in [9] with the the top performers from the MSR 2018 [6] leaderboard as well as other recent state of the art methods.

3. METHODOLOGY

3.1. Datasets

3.1.1. Speech

During our investigation of publicly accessible datasets, we found that Tamil had a much larger speech dataset available when compared to Telugu. This led us to formulate our approach of developing a Telugu ASR model using Transfer Learning on top of layers extracted from the ASR model built for Tamil. The speech datasets used are as follows Table 2.

Lang	Dataset(s)	Duration	Total
Telugu	MSR Open Data [6]	45 h	53.71 h
	OpenSLR [10]	5.71 h	
	CMU INDIC DB [11]	3 h	
Tamil	Mozilla [12]	217.5 h	417.5 h
	IITM [7]	118 h	
	MSR Open Data [6]	45 h	
	OpenSLR [10]	7.5 h	

Table 2. Publicly available speech datasets of Tamil and Telugu Languages

3.1.2. Language Modeling

Though we had the option to use pre-built open-source language models, we took a raw corpus of Tamil and Telugu by IndicCorp [13] which was built by scraping thousands of web sources - primarily news, magazines and books, over a duration of several months. Their statistics are presented in Table 3.

Lang	# Articles	Sentences	Tokens
ta [14]	4.41M	31.5M	582M
te [15]	3.98M	47.9M	674M

Table 3. Tamil and Telugu text corpora used for language modeling

3.2. Github Repository

With the aim of contributing to the open-source community eventually, we created 3 Github repositories which are as follows:

- **speech_rec** [16]: This is the main repository for the project. The other two repositories are submodules of this repository. Most of the data preprocessing and handling modules are implemented in this repository.

- **kaldi_tamtel** [17]: In this repository, the Kaldi recipe for Tamil and Telugu customized from the Tedlium recipe is hosted. Logs [18] are maintained for most of the steps in entire pipeline of the recipe. This is a repository for the project. The other two repositories are submodules of this repository.

- **kaldi_tamtel_db** [19]: In this repository, the Kaldi data files which are input to Kaldi and are generated by various stages of the pipeline are hosted to keep track of changes. Although files which are too big for Github are not hosted as of now.

We have first started working on the Tamil part of the project. At the time of the submission, we haven't yet started working on the Telugu part explicitly. Therefore, in the following sections, most of the work will be focused on Tamil. Yet, it is easy to see that it can be extended to Telugu as well with minimal effort and complexity.

3.3. Preprocessing for language modeling

As the transcripts and corpora used for language modeling had a lot of undesirable entities like english words, punctuation, special characters etc., we devised a preprocessing pipeline [20] to remove these entities and to produce clean datasets. The salient features of the preprocessing steps are:

- **Repleace numbers with words**: Numbers written as numerals are converted to words using the `replace_num.sh` [21] script. This script uses the `indic-num2words` [21] module to get the words(s) for a given number in the given Indian language.

- **Remove URLs**: As the text contained urls, they had to be removed.

- **Remove parentheses**: Various kinds of parentheses and their contents are removed from the text.

- **Remove punctuation and special characters**: Punctuation and special characters except for periods are removed.

- **Remove parentheses**: Various kinds of parentheses and their contents are removed from the text. This script uses the `indic-num2words` module to get the words(s) for a given number in the given Indian language.

- **Remove non-language and whitespace character**: This step is kind of a fail safe. All characters except for the unicode range of the language and whitespace are replaced by a space. This step is kind of a fail safe as it is guaranteed to catch anything that's not caught in the above steps. It is to be ensured that the data is cleaned and adapted as much as possible before this step.

- **Convert periods into newlines**: As PocoLM uses newlines as the delimiter, we need to convert the periods into newlines.

- **Delete empty lines**: There is a possibility that some lines are empty after the above steps. These lines are deleted.

- **Fix whitespace**: Make sure there are no leading or trailing whitespace in each line. Make sure that there are no multiple whitespaces in each line.

- **Validate end result**: Each line of the corpus is checked whether it contains only tamil words that are separated by a single space without any leading or trailing whitespace. It is also verified that there are no empty lines in the processed corpus. Instead of verifying on the whole corpus, a random sample of 25k lines [22] is extracted out of the processed corpus and the validation is run using the `validate_text_file.sh` script [23].

3.4. Preprocessing Speech Dataset

The transcript texts of the audio are preprocessed similar to the steps outlined above for the most part. Therefore, skipping the details for brevity. As we have 4 different datasets, it's vital to have the audio files in standard format. The audio files are handled as follows:

- Convert all files to `wav` format using soxi

  - Sample Rate : 16000
  - Precision : 16-bit
  - Sample Encoding: 16-bit Signed Integer PCM

- Prepare data (input files to Kaldi) for each of the datasets in the required formats with the standard `dev, test, train` partition. This is done using the `create_data_files.py` module [24].
- Merge the data files for each of the datasets into a single unified dataset (with the partitions). Implemented in `merge_datasets.sh` script [25].
- **Create lexicon:** Significant effort was spent on using a unified parser [26] which would spell out phonemes in a uniform format for multiple Indian languages. phonemesMerge the data files for each of the datasets into a single unified dataset (with the partitions). We weren't able to get it working in time and we move on to simply splitting a word into its constituent unicode characters and treated that list as the pronunciation oh phonemes. This is implemented in `create_lexicon.py` module [27] and a snippet is shown in Fig. 2.

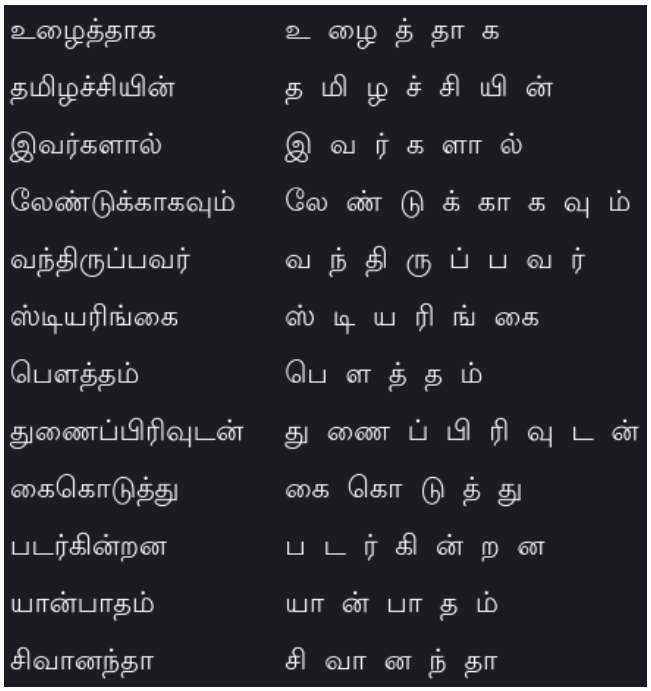

Fig. 2. Snippet of lexicon created by spiltting characters

3.5. Customizing Tedlium Recipe

We copy the Tedlium `s5_r3` folder into the `egs` folder of Kaldi and start making changes to build the recipe [17] for our project. The changes are as follows:

- **Adapting data path and initial stages**: The merged Kaldi input data from the above is placed in the `db` folder and relevant changes are made to the data download
- **Run utils/fix_data_dir.sh**: There seemed to have been a sorting issue in the data directory. This utility script is run to fix the issue.
- **Language model script**: `ted_train_lm.sh` is modified into `tamil_train_lm.sh` with it being pointed to the relevant data files to pick the data extracted after preprocessing.
- **Fix PocoLM script(s) encoding bug**: A few scripts like `prepare_int_data` were failing due to an encoding issue [28] in calling the method `subprocess.check_output`. This is fixed by adding the parameter `encoding='utf-8'` to the method in the PocoLM scripts.

3.6. Running the stages of the recipe for Tamil

3.7. Language Model Training

We were successful in building the language model (Stages 4 and 5) using the training text obtained after the earlier mentioned preprocessing. The following are the perplexity values for the language model:

Model	Log-Prob per Word	Perplexity
lm_4_prune_big [29]	-7.93	2785.2
lm_4_prune_small [30]	8.40	4472.5

Table 4. Log-Probability per Word and Perplexity calculated for the language models over 109506 words on `real_dev_set.txt`

-

3.8. Acousting Model Training

We were able to run all stages upto the stage 12 (of Tedlium recipe) successfully (eventually). During this

process, we were halted by a few errors in the earlier stages which we were able to fix with some debugging and some minor changes. At the time of writing this report, we are debugging an issue with `sort` in Stage 13.

4. RESULTS

Due to the above mentioned situation, we aren't able to report any comprehensive results apart from building the language model from scratch using a raw corpus. We will keep working on finishing the outlined goals and report more results in a later communication.

5. CONCLUSION

As mentioned above, we don't have any conclusions with any closure yet apart from an understanding of the Kaldi pipeline and the Tedlium recipe. In addition to that, we have designed and implemented necessary pre-processing techniques used on corpora to build language models.

6. PENDING/FUTURE WORK

- Finish acoustic modeling training and evaluation for Tamil.
- Run the same recipe adapted to Telugu
- Investigate using layers from Tamil/English neural networks for Telugu
- If possible, extend to other Dravidian languages like Kannada, Malayalam, etc.

7. REFERENCES

[1] David M. Eberheard, Gary F. Simons, and Charles D. Fennig, "Ethnologue: Languages of the world. twenty-second edition.. dallas, texas: Sil international," `http://www.ethnologue.com/22/country/IN/status/`, 2019.

[2] Ministry of Statistics National Statistical Office and Govt. of India Programme Implementation, "Key Indicators of Household Social Consumption on Education in India - Table (2.1): Literacy rate (in per cent) among persons of age 7 years and above for different States," `https://www.mospi.gov.in/documents/213904/301563//KI_Education_75th_Final1602590967945.pdf/4d0dcdc4-a8f0-0795-df06-be25f2b3a6f7`, 2019.

[3] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight, "Transfer learning for low-resource neural machine translation," *arXiv preprint arXiv:1604.02201*, 2016, `https://arxiv.org/abs/1604.02201`.

[4] Toan Q Nguyen and David Chiang, "Transfer learning across low-resource, related languages for neural machine translation," *arXiv preprint arXiv:1708.09803*, 2017, `https://arxiv.org/abs/1708.09803`.

[5] Bryan Li, Xinyue Wang, and Homayoon Beigi, "Cantonese automatic speech recognition using transfer learning from mandarin," 2019, `https://arxiv.org/abs/1911.09271`.

[6] Brij Mohan Lal Srivastava, Sunayana Sitaram, Rupesh Kumar Mehta, Krishna Doss Mohan, Pallavi Matani, Sandeepkumar Satpal, Kalika Bali, Radhakrishnan Srikanth, and Niranjan Nayak, "Interspeech 2018 Low Resource Automatic Speech Recognition Challenge for Indian Languages.," in *SLTU*, 2018, pp. 11–14, `https://msropendata.com/datasets/7230b4b1-912d-400e-be58-f84e0512985e`.

[7] IIT Madras Speech Lab, "Hindi-Tamil-English ASR Challenge," 2021, `https://sites.google.com/view/indian-language-asrchallenge/home`.

[8] Anuj Diwan, Rakesh Vaideeswaran, Sanket Shah, Ankita Singh, Srinivasa Raghavan K. M., Shreya Khare, Vinit Unni, Saurabh Vyas, Akash Rajpuria, Chiranjeevi Yarra, Ashish R. Mittal, Prasanta Kumar Ghosh, Preethi Jyothi, Kalika Bali, Vivek Seshadri, Sunayana Sitaram, Samarth Bharadwaj, Jai Nanavati, Raoul Nanavati, Karthik Sankaranarayanan, Tejaswi Seeram, and Basil Abraham, "Multilingual and code-switching ASR challenges for low resource indian languages," *CoRR*, vol.

abs/2104.00235, 2021, https://arxiv.org/abs/2104.00235.

[9] Tahir Javed, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra, "Towards building asr systems for the next billion users," *arXiv preprint arXiv:2111.03945*, 2021, https://arxiv.org/abs/2111.03945.

[10] Fei He, Shan-Hui Cathy Chu, Oddur Kjartansson, Clara Rivera, Anna Katanova, Alexander Gutkin, Isin Demirsahin, Cibu Johny, Martin Jansche, Supheakmungkol Sarin, and Knot Pipatsrisawat, "Open-source Multi-speaker Speech Corpora for Building Gujarati, Kannada, Malayalam, Marathi, Tamil and Telugu Speech Synthesis Systems," in *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, Marseille, France, May 2020, pp. 6494–6503, European Language Resources Association (ELRA), https://www.aclweb.org/anthology/2020.lrec-1.800.

[11] Language Technologies Institute at Carnegie Mellon University, "CMU INDIC speech synthesis databases," http://festvox.org/cmu_indic/index.html.

[12] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber, "Common Voice: A Massively-Multilingual Speech Corpus," *CoRR*, vol. abs/1912.06670, 2019, http://arxiv.org/abs/1912.06670.

[13] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar, "IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages," in *Findings of EMNLP*, 2020, https://indicnlp.ai4bharat.org/corpora/.

[14] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar, "IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages," in *Findings of EMNLP*, 2020, https://storage.googleapis.com/ai4bharat-public-indic-nlp-corpora/indiccorp/ta.tar.xz.

[15] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar, "IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages," in *Findings of EMNLP*, 2020, https://storage.googleapis.com/ai4bharat-public-indic-nlp-corpora/indiccorp/te.tar.xz.

[16] Sourya Kakarla, "Main Github Resository for Tamil and Telugu ASR Project," 2022, https://github.com/ma08/speech_rec.

[17] Sourya Kakarla, "Github resository for hosting the customized Kaldi recipe for Tamil and Telugu ASR Project," 2022, https://github.com/ma08/kaldi_tamtel/.

[18] Sourya Kakarla, "Kaldi logs of the Tamil and Telugu recipe," 2022, https://github.com/ma08/kaldi_tamtel/tree/main/logs.

[19] Sourya Kakarla, "Github resository for hosting small data files for the Tamil and Telugu ASR Project," 2022, https://github.com/ma08/kaldi_tamtel_db/.

[20] Sourya Kakarla, "clean_corpus.sh - A script to clean the raw text, speech_rec Github reporitory," 2022, https://github.com/ma08/speech_rec/blob/master/lang_model/preprocessing/clean_corpus.sh.

[21] Sourya Kakarla, "Shell script to replace numers with words, speech_rec repository, Github," 2022, https://github.com/ma08/kaldi_tamtel_db/.

[22] Sourya Kakarla, "stage5_out_rand 25klines.sh - A random sample of 25k lines form the processed corpus for testing, speech_rec Github reporitory," 2022, https://github.com/

ma08/speech_rec/blob/master/lang_model/preprocessing/stage5_out_rand25klines.txt.

[23] Sourya Kakarla, "validate_text_file.py - Validates whether a corpus file is clean, speech_rec Github repository," 2022, https://github.com/ma08/speech_rec/blob/master/lang_model/validate_text_file.py.

[24] Sourya Kakarla, "create_data_files.sh - Creates files needed by Kaldi from the speech dataset, speech_rec Github reporitory," 2022, https://github.com/ma08/speech_rec/blob/master/dataset_related/create_data_files.py.

[25] Sourya Kakarla, "merge_datasets.sh - Merge individual Kaldi input folders, kaldi_tamtel_db Github reporitory," 2022, https://github.com/ma08/kaldi_tamtel_db/blob/main/merge_datasets.sh.

[26] Arun Baby, Nishanthi NL, Anju Leela Thomas, and Hema A Murthy, "A unified parser for developing indian language text to speech synthesizers," in *International Conference on Text, Speech, and Dialogue*. Springer, 2016, pp. 514–521, https://www.iitm.ac.in/donlab/tts/unified.php.

[27] Sourya Kakarla, "create_lexicon.py - Create lexicon for Kaldi from text file, speech_rec Github reporitory," 2022, https://github.com/ma08/speech_rec/blob/master/dataset_related/create_lexicon.py.

[28] Sourya Kakarla, "Log file for encoding bug, kaldi_tamtel_db Github repository," 2022, https://github.com/ma08/kaldi_tamtel_db/blob/main/data/local/local_lm_failed_encoding/data/work/log/wordlist/prepare_int_data.log.

[29] Sourya Kakarla, "Log file for Stage 4 lm_4_prune_big stats, kaldi_tamtel Github repository," 2022, https://github.com/ma08/kaldi_tamtel/blob/main/logs/stage4_5_4.log.

[30] Sourya Kakarla, "Log file for Stage 4 lm_4_prune_small stats, kaldi_tamtel Github repository," 2022, https://github.com/ma08/kaldi_tamtel/blob/main/logs/stage4_5_5.log.