Investigation of Different G2P Schemes for Speech Recognition in Sanskrit

Anoop C. S.(✉)[1][0000−0003−2850−9696] and
A. G. Ramakrishnan[1][0000−0002−3646−1955]

Indian Institute of Science, Bengaluru, Karnataka - 560012, India
{anoopcs,agr}@iisc.ac.in

Abstract In this work, we explore the impact of different grapheme to phoneme (G2P) conversion schemes for the task of automatic speech recognition (ASR) in Sanskrit. The performance of four different G2P conversion schemes is evaluated on the ASR task in Sanskrit using a speech corpus of around 15.5 hours duration. We also benchmark the traditional and neural network based Kaldi ASR systems on our corpus using these G2P schemes. Modified Sanskrit library phonetic (SLP1-M) encoding scheme performs the best in all Kaldi models except for the recent end-to-end (E2E) models trained with flat-start LF-MMI objective. We achieve the best results with factorized time-delay neural networks (TDNN-F) trained on lattice-free maximum mutual information (LF-MMI) objective when SLP1-M is employed. In this case, SLP1 achieves a word error rate (WER) of 8.4% on the test set with a relative improvement of 7.7% over SLP1. The best E2E models have a WER of 13.3% with the basic SLP1 scheme. The use of G2P schemes employing schwa deletion (as in Hindi, which uses the same Devanagari script as Sanskrit) degrades the performance of GMM-HMM models considerably.

Keywords: G2P · ASR · Sanskrit · LF-MMI · E2E · Kaldi.

1 Introduction and motivation for the study

Sanskrit is one of the oldest languages known to the human race. It is one of the 22 scheduled languages listed in the eighth schedule to the Indian constitution and has official status in the states of Himachal Pradesh and Uttarakhand. It is the only scheduled language in India with less than 1 million native speakers. Though not used in active communication currently, Sanskrit has a lasting impact on the languages of South Asia, Southeast Asia, and East Asia, especially in their formal and learned vocabularies. Many of the modern-day Indo-Aryan languages directly borrow the grammar and vocabulary from Sanskrit. In addition, it encompasses a vast body of literature in various areas spanning from mathematics, astronomy, science, linguistics, mythology, history, and mysticisms.

Though Sanskrit studies bear importance due to historical and cultural reasons, very little focus has been put on developing necessary computational tools to digitize the vast body of literature available in the language. Specifically, building a large vocabulary automatic speech recognition (ASR) system can expedite the digitization of a large volume of literature available across various written forms in Sanskrit. However, a major challenge in building such a system for any Indian language is the lack of readily available training data. Furthermore, an ASR system for Sanskrit poses additional challenges like the high rate of inflection, free word order, highly intonated speech, and large word lengths due to sandhi. These issues make speech recognition in Sanskrit both unique and challenging compared to other spoken languages.

Despite the above-mentioned issues, there have been a few attempts to build ASR systems for Sanskrit in recent times. [3] employ Gaussian mixture model (GMM) - hidden Markov model (HMM) based approaches using HMM toolkit (HTK) [28]. Recently [1] has attempted Sanskrit speech recognition using Kaldi [23] time-delay neural network (TDNN) models [19]. End-to-end speech recognition with connectionist temporal classification (CTC) objective [10] has been performed on Sanskrit with limited amount of data using spectrogram augmentation [18] techniques in [4]. [2] employs domain adaptation approaches to learn the acoustic models for Sanskrit from the annotated data in Hindi.

Pronunciation modeling is one of the major components of all traditional ASR systems. Sanskrit possesses alphabets that are largely designed to avoid multiple pronunciations for the same alphabet. Thus, there is a one-to-one correspondence between the alphabets and the corresponding sound units in most cases. However, there are a few instances where the context modifies the pronunciation of the alphabet. The presence of *chandra bindu* (◌ँ), *anusvāra* (◌ं), *visarga*(◌ः), *nukta*(◌़) or *virama*(◌्) in the alphabet alters the pronunciation.

The pronunciation rules for handling them are described in [3]. Sanskrit is normally written in Devanagari script, just like Hindi. Hence the pronunciation variations due to *virama* are handled in most grapheme to phoneme (G2P) converters for Devanagari, as they also occur in Hindi. However, most of the G2P schemes consider *anusvara* and *visarga* as individual phonemes without considering their context. They also appear quite frequently in Sanskrit text. How the performance of the ASR for Sanskrit will be impacted when these contextual pronunciation rules are incorporated into the G2P converter is unclear. Also, being a more popular Indian language, Hindi has many off-the-shelf G2P conversion tools [5, 17]. One might be tempted to use the same tools for Sanskrit as they both share the same orthography. However, Hindi G2P conversion tools are designed to handle schwa deletion, a phenomenon where the inherent vowel in the orthography is not pronounced in some contexts. The impact of using such G2P tools for speech recognition in Sanskrit has not been studied so far. In this work, we address the above issues and investigate the performance of four different G2P schemes on Sanskrit ASR. We evaluate their performance on multiple Kaldi-based ASR systems on a Sanskrit dataset of 15.5 hours duration that we have collected.

The remaining part of the paper is organized as follows: Section 2 introduces the Sanskrit dataset used in this work. Section 3 gives a theoretical overview of different speech recognition systems in Kaldi, which we explore for the evaluation of G2Ps. Section 4 describes the setup used for our experiments. This section also details the four G2P schemes and the Kaldi systems used. Results of our experiments are discussed in section 5 followed by conclusions in section 6.

2 Sanskrit speech dataset collected for the study

The Sanskrit dataset we use in this work has 15.5 hours of speech data consisting of 7900 utterances from 41 speakers. The speech data is mainly from 3 domains: (a) news recordings, (b) recordings of short stories, and (c) video lectures. Speech data is in raw wav file format with 16 kHz sampling frequency and 16 bits per sample. There are 41 speakers: 21 male and 20 female. Each audio file contains recordings from a single speaker. The corpus contains around 15100 unique words. The dataset is split into train and test subsets with approximately 12 and 3.5 hours of data, respectively, with no overlap between speakers in the subsets.

The details of the dataset are given in Table 1.

Table 1. Details of the Sanskrit speech dataset collected by us from the recordings of news, short stories, and lectures, for our ASR experiments. The ”Words” column gives the number of unique words in the subsets.

	Details		**Speakers**			**Duration**		
Data	**Utterances**	**Words**	**Total**	**Male**	**Female**	**Total**	**Male**	**Female**
Train	6067	11537	16	8	8	11:54:25	5:22:16	6:32:09
Test	1813	5145	25	13	12	3:28:52	1:44:12	1:44:39

3 Summary of different Kaldi-based ASR systems

ASR can be treated as a sequence detection problem which maps a sequence of T-length input acoustic features $O = \{\mathbf{o}_t \in \mathcal{R}^m, t = 1, 2, \ldots, T\}$ to an L-length word sequence $W = \{\mathbf{w}_l \in V, l = 1, 2, \ldots, L\}$. $\mathbf{o}_t$ denote the m-dimensional acoustic feature vector at time t, $\mathbf{w}_l$ denote word at position l and V denote the vocabulary in that language. The ASR estimates the most probable word sequence $\hat{W}$ given O.

For ASR systems employing HMMs [24],

$$\begin{aligned}\hat{W} &= \arg\max_W P(W|O) && (1)\\ &= \arg\max_W \sum_S P(O|S,W)P(S|W)P(W) && (2)\\ &\approx \arg\max_W \sum_S P(O|S)P(S|W)P(W) && (3)\end{aligned}$$

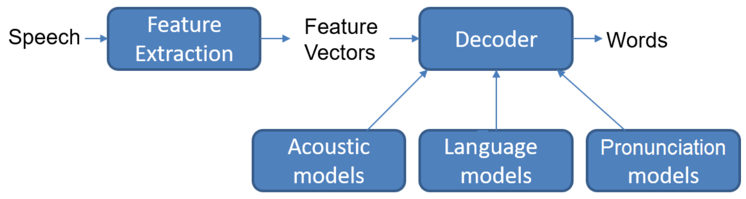

Figure 1. Block diagram of an ASR.

where S denote the underlying HMM state sequence $S = \{s_t \in \mathcal{S}, t = 1, 2, \ldots, T\}$, $\mathcal{S}$ being the set of all possible HMM states. The factors $P(O|S)$, $P(S|W)$, and $P(W)$ in (3) are called the acoustic models, pronunciation models, and language models (LM), respectively [26]. A block diagram of an ASR depicting each of these components are shown in Figure 1.

The acoustic models can be further factorised as:

$$P(O|S) = \prod_{t=1}^{T} P(\mathbf{o}_t|\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{t-1}, S) \approx \prod_{t=1}^{T} P(\mathbf{o}_t|s_t) \tag{4}$$

The inner term $P(\mathbf{o}_t|s_t)$ represents the probability of observing the feature vector $\mathbf{o}_t$ given that the HMM state is $\mathbf{s}_t$ at time t. Traditional GMM-HMM based systems make use of GMM to model this probability. With the arrival of deep neural networks (DNN) for acoustic modeling in speech [12] GMMs made way for multi-layer perceptron (MLP) classifiers which computed the frame-wise posterior $P(s_t|\mathbf{o}_t)$. Frame-wise likelihood $P(\mathbf{o}_t|s_t)$ is then approximated as $\frac{P(s_t|\mathbf{o}_t)}{P(s_t)}$ using the MLP output. However, to obtain the HMM state (senone) labels needed to train MLP, they still depended on the alignments from a GMM-HMM system. These alignments are also used to compute the priors on the senones.

The pronunciation model can be factorized as:

$$P(S|W) = \prod_{t=1}^{T} P(s_t|s_1, s_2, \ldots, s_{t-1}, W) \approx \prod_{t=1}^{T} P(s_t|s_{t-1}, W) \tag{5}$$

HMM state transition probabilities model this term. HMMs are usually prepared at the sub-word level (phonemes/triphones).

n-gram models are used for language modeling.

$$P(W) \approx \prod_{l=1}^{L} P(w_l|w_{l-n+1}, \ldots, w_{l-1}) \tag{6}$$

Decoding typically employs weighted finite state transducers (WFST) [16]. A decoding graph, normally referred to as HCLG, is created as:

$$S \equiv HCLG = min(det(H \ o \ C \ o \ L \ o \ G)) \tag{7}$$

where *o*, *min* and *det* are the FST operations; composition, minimisation and determinization and S is the search graph. Here, H, C, L, and G denote HMM structure, context-dependency of phonemes, lexicon, and grammar. In this graph, each arc has input label as an identifier for the HMM state and output label as a word or ϵ (indicating no label). The weight typically represents the negative log probability of the state. Decoding finds the best path through this graph. Usually, the search space is huge, and beam pruning [13] is employed instead of an exact search, where all the paths not within the beam are discarded.

The assumption in HMM that the observations are independent of past/future phone states, given the current state does not hold in practice. Due to the incorrectness in the model maximum likelihood may not be an optimal training criteria. A lattice-free version of maximum mutual information (MMI) objective [6] for training neural network acoustic models was proposed in [22]. MMI is a sequence-level discriminative criteria given by:

$$\mathcal{F}_{MMI}(\theta) = \sum_{u=1}^{U} log \frac{P_\theta(O_u|W_u)P(W_u)}{\sum_W P_\theta(O_u|W)P(W)} \tag{8}$$

Here U and θ represent the total number of utterances and the model's parameters, respectively. The objective tries to maximize the probability of the reference transcription while minimizing the probability of all other transcriptions. Both numerator and denominator state sequences are represented as HCLG FSTs. The denominator forward-backward computation is parallelised on a graphical processing unit (GPU). They replace word-level LM with a 4-gram phone LM for efficiency. They also use 2-state skip HMMs for phones, where the skip connections allowed for sub-sampling of frames at the DNN output.

Regular LF-MMI still depends on the alignments from GMM-HMM models. The output of the network is tied triphone/biphone states. The tying is done using a decision-tree built from GMM-HMM alignments. Flat-start LF-MMI [11] removes this dependency and makes the training completely end-to-end (E2E). It trains the DNN in a single stage without going through the usual GMM-HMM training and tree-building pipeline. Unlike regular LF-MMI, there is no prior alignment information in the numerator graph here. HMM transitions and state priors are fixed and uniform as in the regular case. Context dependency modeling is achieved by using full left-biphones, where separate HMMs are assumed for each possible pair of phonemes. The phoneme language models for the denominator graph are estimated using the training transcriptions and pronunciation dictionary.

4 Setup for the ASR experiments

In this work, we investigate the performance of different G2P conversion schemes in the ASR task in Sanskrit. We also benchmark different Kaldi-based systems on our Sanskrit dataset described in section 2. We explore the GMM-HMM, MLP, and factored time-delay neural network (TDNN-F) [21] architectures. MLP architectures are trained with frame-level cross-entropy objective. Training of TDNN-F models is explored with both regular and flat-start LF-MMI objectives. All the experiments are performed using Kaldi toolkit [23].

4.1 Extraction of the MFCC-based features

In our experiments, we use Mel-frequency cepstral coefficients (MFCC) as features. They are extracted from frames with a window size of 25 ms and frameshifts of 10 ms. For the training of the GMM-HMM and MLP models, low-resolution MFCC features with 13 coefficients are used. Splicing with left and right contexts of 9 gives a 247-dimensional vector at the input of the MLP. High-resolution features with 40 coefficients are used for training the TDNN-F models with the LF-MMI objective. They also use 100-dimensional i-vectors [7] computed from chunks of 150 consecutive frames. In the E2E scheme, only high-resolution MFCC features are used as features. With three consecutive MFCC vectors, the input dimensions to the regular and E2E TDNN-F networks are 220 and 120, respectively.

4.2 The different pronunciation models studied

A pronunciation lexicon maps the words to the corresponding phoneme sequence with the help of G2P converters. We investigate four G2P schemes for Sanskrit pronunciation modeling and evaluate their relative performance in speech recognition. They are:

1. Epitran [17]: Epitran is a massive multilingual G2P system supporting 61 languages, including Hindi. Though this system supports the Devanagari script, it is primarily designed to handle the schwa deletions in Hindi. Performance evaluation of the Sanskrit ASR using this scheme can give an idea about the negative impact of using Hindi G2P converters for Sanskrit.
2. Indian language speech sound label set (ILSL12) [14]: ILSL12 provides a standard set of labels (in Roman script) for speech sounds commonly used in Indian languages. Similar sounds in different languages are given a single label. It is commonly used in multilingual speech recognition systems in Indian languages.
3. Sanskrit library phonetic (SLP1) basic encoding scheme [25]: SLP1 can represent phonetic segments, phonetic features, and punctuation in addition to the basic Devanagari characters. SLP1 also describes how to encode classical and Vedic Sanskrit.
4. SLP1 modified with specific pronunciation rules for *anusvāra* and *visarga* (SLP1-M): In SLP1-M, we incorporate the pronunciation modification rules in Sanskrit to the basic SLP1 scheme hoping for a better ASR performance. A few examples for indicating the differences in the encoding schemes are shown in Table 2. In row 1, Epitran removes the final schwa, whereas the other three schemes retain it. Also note that the *anusvāra* is followed by a

Table 2. G2P schemes evaluated for speech recognition in Sanskrit.

Sl. No.	**Deavanagari**	**Epitran**	**ILSL12**	**SLP1**	**SLP1-M**
1	शंकर	ʃ ə ŋ k ə r	sh a **q** k a r a	S a **M** k a r a	S a **N** k a r a
2	गुरुः	g u r u ə **h**	g u r u **hq**	g u r u **H**	g u r u **h u**

velar consonant, क(ka) and hence it is modified to ”N” denoting the velar resonant ङ(nga) in SLP1-M. In row 2, SLP1-M modifies the final *visarga* (◌ः), by adding a voiced echo of the previous vowel. ILSL12 and SLP1 use separate phonemes for the *visarga*. Epitran maps it to ”h”, which denotes the ह (h) sound.

4.3 Training of the language models

We use bigram language models trained with a large text corpus using IRSTLM toolkit [8]. The transcriptions of the speech data are extended with the text data from Sanskrit data dump [27] in wiki and from several websites. The collected text is filtered such that the sentences contain only Devanagari Unicode. There were 244652 sentences with 413828 distinct words in the final text corpus used for language modeling.

4.4 Training of the different acoustic models

GMM-HMM models: The development of GMM-HMM systems involves stepwise refinement of models. At each step, the complexity of the model is increased to refine the alignments of the training data with the model and passed to next step. Initially, context-independent monophone HMMs are built. There are 218 position-dependent phones (i.e., phones marked with their word-internal positions - beginning/ending/standalone/internal). An equal-alignment scheme is used for initial alignments, and the parameters are learned using maximum likelihood estimation.

Next, context-dependent phone (triphone) models are constructed using the left and right contexts of phones. To avoid the explosion in the number of HMM states due to the triphone context, a decision tree is trained to cluster the states that are acoustically similar. This way, the same acoustic model is shared across multiple HMM states. The application of the following transforms further refines these models:

- Linear discriminant analysis (LDA) + maximum likelihood linear transform (MLLT) [9]: Here MFCC frames are spliced with left and right contexts of 3 each, and LDA is applied to reduce the dimension to 40. Now the diagonalizing transform MLLT is estimated over multiple iterations.
- Speaker adaptive training (SAT): This trains a feature-based maximum likelihood linear regression (fMLLR) transform to normalize the features such that they better fit the speaker.

Subspace Gaussian mixture models (SGMM) [20] can compactly represent a large collection of GMMs. Here model parameters are derived from a set of state-specific parameters, and from a set of globally-shared parameters which can capture phonetic and speaker variations.

Hybrid DNN-HMM models with MLP architecture: We use a multi-layer perceptron (MLP) with seven hidden layers and 1024 hidden nodes per layer. The output layer has 2576 nodes corresponding to the senones in the final GMM-HMM model. We randomly select 10% of the training data as the validation set. The network is trained with frame-level cross-entropy (CE) objective.

TDNN-F models with LF-MMI objective: We use TDNN-F models for training with both regular and flat-start LF-MMI objectives. They have similar structure as TDNN, but are trained with the constraint that one of the two factors of each weight matrix should be semi-orthogonal. TDNN-F blocks have a bottleneck architecture with a linear layer followed by an affine layer. The linear layer transforms the hidden-layer dimension to a lower bottleneck dimension, and the affine layer transforms it back to the original dimension. In regular LF-MMI training, the hidden and bottleneck dimensions are 768 and 96, respectively. In flat-start LF-MMI (E2E) scheme, we use hidden and bottleneck dimensions of 1024 and 128, respectively. Linear and affine layers are followed by ReLU activation and batch normalization. They also have a kind of residual connection where the output of the current block is added to the down-scaled (by a factor of 0.66) output of the previous block. There are 12 such blocks. The final output layer has dimensions of 1424 and 108, respectively, for the regular and flat-start LF-MMI networks. Speed and volume perturbations [15] are used in regular LF-MMI training. In the E2E scheme, all the training utterances are modified to be around 30 distinct lengths. Utterances with the same lengths are placed in the same mini-batch during training. Speed perturbation is used to modify the length of each utterance to the nearest of the distinct lengths.

5 Experimental Results

We use word error rate (WER) as the metric to assess the performance of the Kaldi-based systems. WER is defined as:

$$WER = (I + D + S)/N \tag{9}$$

where I, D, S, and N denote the number of insertions, deletions, substitutions, and total words in the test set. $N = 14135$ in our case.

The results of speech recognition experiments for different G2P conversion schemes with the conventional GMM-HMM models are listed in Table 3. The well-curated SLP1-M scheme provides the best results across all the GMM-HMM models. The performance of ILSL12 and SLP1 are almost similar as they are just direct mappings between the orthography and the phonology, except for the

Table 3. The performance (WER in %) of GMM-HMM models on the test set using different G2P schemes for Sanskrit.

No.	Model	Epitran	ILSL12	SLP1	SLP1-M
M1	Mono	27.6	24.4	23.9	**23.0**
M2	Tri	21.1	18.2	18.1	**17.3**
M3	Tri+LDA+MLLT	22.6	20.8	20.0	**18.7**
M4	Tri+LDA+MLLT+SAT	23.3	20.7	20.6	**19.8**
M5	SGMM	17.0	15.7	15.8	**15.4**

differences in handling the pitch accents of Vedic Sanskrit. There is considerable deterioration in the performance of Epitran over the other G2P converters. The minimal degradation ranges from around 1.2% in SGMM models to around 3.2% in monophone HMM models. Though Epitran supports Devanagari graphemes, it is primarily designed to support Hindi, where the schwa deletion is prominent. However, schwa deletion is not present in Sanskrit, and hence its performance is slightly worse than the other G2P conversion schemes. Among the GMM-HMM models, SGMM provides the best results across all the G2P converters. They provide an improvement of at least 1.9% over the best triphone models. All the triphone models are better than the monophone models. The application of LDA+MLLT and SAT does not improve the results. We have assumed that each utterance in the training set belongs to a different speaker to have the effect of more number of speakers in the i-vector training for TDNN-F models. This could be the reason for the poor performance of SAT.

Table 4. The performance (WER in %) of different neural network models on the test set using different G2P schemes for Sanskrit.

No.	Model	Objective function	Model size	Epitran	ILSL12	SLP1	SLP1-M
M6	MLP	CE	9.0 M*	22.6	22.3	21.9	**20.0**
M7	TDNN-F	Regular LF-MMI	6.8 M	9.4	9.4	9.1	**8.4**
M8	E2E	Flat-start LF-MMI	4.9 M	14.9	14.6	**13.3**	14.7

$^*M$ stands for million.

The results of neural network models are listed in Table 4. SLP1-M gives the best results for the MLP and TDNN-F models. The best results for E2E models are achieved using the basic SLP1 scheme. E2E models employ full left-biphones for context dependency modeling. Retaining separate phonemes for *visarga* (◌ः) and *anusvāra* (◌ं) seems to help the basic SLP1 scheme in better modeling the biphone HMMs having *visarga*/*anusvara* as one of the phonemes. Surprisingly the Epitran G2P scheme is not as bad as in the case of GMM-HMM models, when it comes to the neural network models. They provide comparable results with ILSL12. TDNN-F architectures (rows 2 and 3 of Table 4) have better performances and smaller model sizes compared to the normal MLP architectures. TDNN-F model trained with regular LF-MMI objective gives the

Table 5. Performance (WER in %) of the SLP1-M scheme when a separate schwa phoneme is used for the vowel inherent in the consonant characters.

G2P scheme	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
SLP1-M	23.0	17.3	18.7	19.8	15.4	20.0	8.4	14.7
SLP1-M+ schwa phoneme	23.1	18.3	20.6	20.7	15.8	21.4	8.7	14.2

best performance on the corpus with the SLP1-M G2P scheme. However, they depend on GMM-HMM models for the initial alignments and tree building for tying states. On the other hand, the E2E scheme eliminates such dependencies on GMM-HMM models and makes the LF-MMI training flat-start. Also, the i-vector extraction process is not required in the E2E scheme. Thus they can be trained easily. They provide around 5.3-8.6% improvement over the MLP with all the G2P schemes on the Sanskrit dataset. However, when compared to the regular LF-MMI training, their performance is worse by at least 4.2%.

We further extend the SLP1-M with a separate phoneme label for schwa, the vowel inherent in the consonant characters; i.e., we use a different label ”ə” instead of mapping them to ”a”, the label for the vowel अ. The results of this are shown in Table 5. The first row of this Table is replicated from Tables 3 and 4. WER degrades for all the models in this scheme except for the E2E models.

6 Conclusions

This work explores the significance of G2P conversion schemes on the speech recognition task in Sanskrit. We evaluate the performance of four different G2P conversion schemes, viz. Epitran, ILSL12, SLP1, and SLP1-M (SLP1 modified to include some contextual pronunciation rules) using traditional and neural network-based Kaldi models. SLP1-M performs the best among all the models except E2E. For E2E models, basic SLP1 performs the best. SLP1-M brings about some improvement in MLP and TDNN-F models. The relative improvements in WER over SLP1 are 8.7% and 7.7% for MLP and the state-of-the-art TDNN-F models trained with LF-MMI objective, respectively. Epitran scheme, which employs schwa deletion, deteriorates the performance of GMM-HMM models. TDNN-F models trained with LF-MMI objective perform the best among all the Kaldi models. They provide a WER of 8.4% on the Sanskrit test set with SLP1-M. E2E models with flat-start LF-MMI objective achieve a WER of 13.3% with the basic SLP1. However, the WER performance of E2E models is worse by at least 4.2% than the TDNN-F models trained with the regular LF-MMI objective.

Acknowledgments

We thank Science and Engineering Research Board, Government of India for partially funding this research through the IMPRINT2 project, IMP/2018/000504.

References

1. Adiga, D., Kumar, R., A., K., Jyothi, P., Ramakrishnan, G., Goyal, P.: Automatic speech recognition in Sanskrit: A new speech corpus and modelling insights. In: 59th Annual Meeting of the Association for Computational Linguistics, ACLFindings (2021). pp. 5039–5050. https://doi.org/10.18653/v1/2021.findings-acl.447
2. Anoop, C.S., Prathosh, A.P., Ramakrishnan, A.G.: Unsupervised domain adaptation schemes for building ASR in low-resource languages. In: Proceedings of Workshop on Automatic Speech Recognition and Understanding, ASRU (2021)
3. Anoop, C.S., Ramakrishnan, A.G.: Automatic speech recognition for Sanskrit. 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies, ICICICT **1**, pp. 1146–1151 (2019). https://doi.org/10.1109/ICICICT46008.2019.8993283
4. Anoop, C.S., Ramakrishnan, A.G.: CTC-based end-to-end ASR for the low resource Sanskrit language with spectrogram augmentation. 27th National Conference on Communications, NCC (2021). pp. 1–6. https://doi.org/10.1109/NCC52529.2021.9530162
5. Arora, A., Gessler, L., Schneider, N.: Supervised grapheme-to-phoneme conversion of orthographic schwas in Hindi and Punjabi. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7791–7795. Association for Computational Linguistics, (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.696
6. Bahl, L.R., Brown, P.F., de Souza, P.V., Mercer, R.L.: Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In:ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. pp. 49–52 (1986)
7. Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P.: Front-end factor analysis for speaker verification. IEEE Transactions on Audio, Speech, and Language Processing **19**, 788–798 (2011)
8. Federico, M., Bertoldi, N., Cettolo, M.: IRSTLM: an open source toolkit for handling large scale language models. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH (2008)
9. Gales, M.: Semi-tied covariance matrices for hidden Markov models. IEEE Transactions on Speech and Audio Processing **7**(3), 272–281 (1999). https://doi.org/10.1109/89.759034
10. Graves, A., Fernández, S., Gomez, F.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: In Proceedings of the International Conference on Machine Learning, ICML. pp. 369–376 (2006)
11. Hadian, H., Sameti, H., Povey, D., Khudanpur, S.: End-to-end speech recognition using lattice-free MMI. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. pp. 12–16 (2018). https://doi.org/10.21437/Interspeech.2018-1423
12. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior,A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597
13. Hugo Van, H., Filip Van, A.: An adaptive-beam pruning technique for continuous speech recognition. In: Proceeding of Fourth International Conference on Spoken Language Processing, ICSLP, IEEE. **4**, pp. 2083–2086 (1996)

14. Samudravijaya, K., Murthy H.A.: Indian language speech sound label set (ILSL12). In: Indian Language TTS Consortium & ASR Consortium, (2012) `https://www.iitm.ac.in/donlab/tts/downloads/cls/cls\_v2.1.6.pdf`
15. Ko, T., Peddinti, V., Povey, D., Khudanpur, S.: Audio augmentation for speech recognition. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. January, pp. 3586–3589 (2015)
16. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Computer, Speech and Language. **16**(1), 69–88 (2002). https://doi.org/10.1006/csla.2001.0184
17. Mortensen, D., Dalmia, S., Littell, P.: Epitran: Precision G2P for many languages. In: 11th International Conference on Language Resources and Evaluation, LREC 2018. pp. 2710–2714 (2019)
18. Park, D., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E., Le, Q.: Specaugment: A simple data augmentation method for automatic speech recognition. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. September, pp. 2613–2617 (2019)
19. Peddinti, V., Povey, D., Khudanpur, S.: A time delay neural network architecture for efficient modeling of long temporal contexts. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. 2015-January, pp. 3214–3218 (2015)
20. Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., Rose, R., Schwarz, P.,Thomas, S.: The subspace Gaussian mixture model - a structured model for speech recognition. Computer Speech and Language **25**(2), 404–439 (2011). https://doi.org/10.1016/j.csl.2010.06.003
21. Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohamadi, M., Khudanpur,S.: Semi-orthogonal low-rank matrix factorization for deep neural networks. In:Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. 2018-September, pp. 3743–3747 (2018). https://doi.org/10.21437/Interspeech.2018-1417
22. Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang,Y., Khudanpur, S.: Purely sequence-trained neural networks for ASR based on lattice-free MMI. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. pp. 2751–2755 (2016). https://doi.org/10.21437/Interspeech.2016-595
23. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely,K.: The Kaldi speech recognition toolkit. IEEE Workshop on Automatic Speech Recognition and Understanding (Dec 2011)
24. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2), 257–286 (1989). https://doi.org/10.1109/5.18626
25. Scharf, P.M., Hyman, M.D.: Linguistic issues in encoding Sanskrit. In: The Sanskrit Library (2012)
26. Watanabe, S., Hori, T., Kim, S., Hershey, J., Hayashi, T.: Hybrid CTC/attention architecture for end-to-end speech recognition. IEEE Journal on Selected Topics in Signal Processing. **11**(8), 1240–1253 (2017). https://doi.org/10.1109/JSTSP.2017.2763455
27. Wikimedia: Wiki Sanskrit data dump, `https://dumps.wikimedia.org/sawiki/`
28. Young, S.J., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., Woodland, P.: The HTK Book Version 3.4. Cambridge University Press (2006).