Automatic Speech Recognition for Sanskrit

ANOOP C. S.
Dept. of Electronics and Communication Engineering
Govt. Engineering College, Sreekrishnapuram
Palakkad, Kerala, India
anoopcs@gecskp.ac.in

A. G. RAMAKRISHNAN
Dept. of Electrical Engineering
Indian Institute of Science
Bangalore, India
agr@iisc.ac.in

***Abstract*—This paper presents our work on building a speaker independent, large vocabulary continuous speech recognition system for Sanskrit using HMM Toolkit (HTK). To our knowledge, this is the maiden attempt on a Sanskrit automatic speech recognizer. A Sanskrit speech corpus with a vocabulary size of 8370 words is built. The corpus contains orthographic, phoneme and word level transcriptions of 1360 sentences. The speech data were collected from 3 sources: All India Radio website, Indian Heritage Group under C-DAC and Vyoma Linguistic Labs Foundation. Mel Frequency Cepstral Coefficients together with 0th order coefficient and delta and acceleration parameters are used as features. Triphone HMMs, trained using HTK, are used as acoustic model. Bigram probabilities with back-off smoothing are used as language model. Both phoneme and word level recognizers were developed on the Sanskrit corpus. The system provides a word level accuracy of 89.64% and a sentence level correctness of 58.76% on the test set of 274 sentences. A graphical user interface for the speech recognizer is built using Java Swings.**

***Keywords*—Sanskrit ASR, Speech Recognition, ASR, Sanskrit, Sanskrit speech corpus, HTK, TIMIT, Hidden Markov Models, HMM, MFCC**

I. Introduction

Most of the research work on automatic speech recognition (ASR) is focused on English. Recently, a lot of effort has been put into developing ASR for Indian languages [1] [2] [3] [4]. Sanskrit is one of the oldest Indian languages, which has a very influential role in Indian heritage. The available Sanskrit literature encompasses a rich tradition of poetry and drama as well as scientific, technical, philosophical and religious texts. Sanskrit continues to be widely used as a ceremonial language in Hindu religious rituals and Buddhist practice in the form of hymns and chants. Sanskrit is also used extensively in the Carnatic and Hindustani branches of classical music. Kirtanas, bhajans, stotras, and shlokas of Sanskrit are popular throughout India. Sanskrit is among the 14 original languages of the eighth schedule to the constitution. The state of Uttarakhand has ruled Sanskrit as its second official language. Several villages in India, like Mattur (Shimoga district, Karnataka), still uses Sanskrit in everyday communication. This work is an attempt to provide a helping hand to the revival of Sanskrit language by building an effective ASR. The development of such a system would help in converting the audio books available in Sanskrit to corresponding transcriptions. It can also be very useful to digitize ancient documents on palm leaf manuscripts simply by someone reading it. These efforts can help in the rejuvenation of Sanskrit.

The performance of a large vocabulary continuous speech recognition system depends largely on the quality of the phoneme level recognizer. As a result, various methods have been attempted to improve the phoneme level recognizer using a variety of features, fusion of feature sets, improved statistical models, improvements in pronunciation, acoustic and language models, etc.

Over the past 2 decades, most of the speech recognition research was based on hidden Markov models (HMM) [5] [6] [7] [8]. They used Gaussian mixture models (GMM) to represent the HMM states. The acoustic feature vectors were made up mainly of Mel frequency cepstral coefficients (MFCCs) or perceptual linear predictive coefficients (PLPs) [9]. The HMM Toolkit (HTK) [10] [11] [12] [13] is a portable toolkit developed for building and manipulating hidden Markov models.

But in the last few years, the speech recognition research has shifted its focus to deep neural networks (DNNs) [14] [15] for training HMMs. With the improvements in machine learning algorithms and computational capabilities, a greater number of hidden layers and output layers can be accommodated in DNNs.

II. An overview of our approach

Speech signal is treated as an encoded version of a sequence of discrete symbols which carry some meaning. The function of our automatic speech recognizer is to map the speech data back to the meaningful sequence of symbols. A sequence of Sanskrit words $\mathbf{W} = w_1, w_2, \ldots w_K$ produces a sequence of acoustic vectors or observations $\mathbf{O} = \mathbf{o_1}, \mathbf{o_2}, \ldots \mathbf{o_T}$. We need to decode the most probable word sequence given the observations, i.e.

$$\widehat{\mathbf{W}} = \arg\max_{w} P(\mathbf{W}|\mathbf{O}) \tag{1}$$

Using Bayes' rule, we have

$$P(\mathbf{W}|\mathbf{O}) = \frac{P(\mathbf{O}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{O})} \tag{2}$$

The denominator term $P(\mathbf{O})$ is independent of the word sequence. Thus we have:

$$\widehat{\mathbf{W}} = \arg\max_{\mathbf{w}} P(\mathbf{O}|\mathbf{W})P(\mathbf{W}) \tag{3}$$

The first term $P(\mathbf{O}|\mathbf{W})$ represents the probability of the observation sequence given the word sequence. This is termed as *acoustic model*. The second term $P(\mathbf{W})$ represents the probability of the word sequence and is termed as *language model*. Language models are represented in terms of a finite state network (FSN).

For large vocabulary continuous speech recognition (LVCSR) systems, it is a common practice to build statistical models at the sub-word level (e.g. phonemes) and concatenate them to synthesize the word level models. This is accomplished with the help of a pronunciation dictionary, which contains the composition of each word in the vocabulary in terms of its sub-word units. In Sanskrit, just like the other Indian languages, the pronunciation is unique for each of the words. Suppose that the word sequence $\mathbf{W}$ is composed of pronunciation sequence $\mathbf{U} = \mathbf{u_1}, \mathbf{u_2}, \ldots \mathbf{u_K}$ where each $\mathbf{u_i}, i = 1 \ldots K$ is the pronunciation corresponding to word w_i. Each u_i can be expressed as a sequence of sub-word units like phonemes, i.e. $\mathbf{u_i} = u_i^{(1)}, u_i^{(2)}, \ldots, i = 1 \ldots K$, where $u_i^{(j)} \in \mathcal{L}$, the set of basic phoneme units in Sanskrit language. Then

$$P(\mathbf{O}|\mathbf{W}) = \sum_{\mathbf{U}} P(\mathbf{O}|\mathbf{U})P(\mathbf{U}|\mathbf{W}) \tag{4}$$

where

$$P(\mathbf{U}|\mathbf{W}) = \prod_{i=1}^{K} P(\mathbf{u_i}|w_i) \tag{5}$$

The summation in (4) can be simplified with max operation since we have a one-to-one correspondence between the word and its pronunciation. In this case $P(\mathbf{U}|\mathbf{W}) = 1$. So, the problem of finding $\mathbf{W}$ that maximizes $P(\mathbf{O}|\mathbf{W})$ is the same as the problem of finding $\mathbf{U}$ that maximizes $P(\mathbf{O}|\mathbf{U})$.

The language model term in (3), $P(\mathbf{W})$ can be given as:

$$\begin{aligned} P(\mathbf{W}) &= P(w_1, w_2, \ldots w_K) \\ &= P(w_1)P(w_2|w_1)\ldots P(w_K|w_1, w_2, \ldots w_{K-1}) \end{aligned} \tag{6}$$

It is almost impossible to estimate $P(w_i|w_1, w_2, \ldots w_{i-1})$ for all possible sequence lengths and for all the words in the language. So it is common to use N-gram models where we approximate $P(w_i|w_1, w_2, \ldots w_{i-1})$ as:

$$P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-N+1}, \ldots w_{i-1}) \tag{7}$$

N-grams are computed from a large text corpus using the relative frequency approach.

$$P(w_i|w_1, w_2, \ldots w_{i-1}) = \frac{count(w_{i-N+1}, \ldots, w_{i-1}, w_i)}{count(w_{i-N+1}, \ldots, w_{i-1})} \tag{8}$$

Due to the limited size of the training text corpus, for many possible word sequences $count(w_{i-N+1}, \ldots, w_{i-1}, w_i)$ may be zero. To overcome this issue, *smoothing* techniques [16] are used. We use bigrams with back-off smoothing.

Once the acoustic models and language models are ready, they are incorporated into the finite state network formed by the sequence of HMM models. Viterbi algorithm is used to determine the best path through the network. The decoded state sequence is translated to the corresponding word sequence.

III. Architecture of the ASR for Sanskrit

A block diagram of the automatic speech recognition (ASR) system for Sanskrit language [17] is shown in Fig. 1. The following sections describe the block diagram in detail.

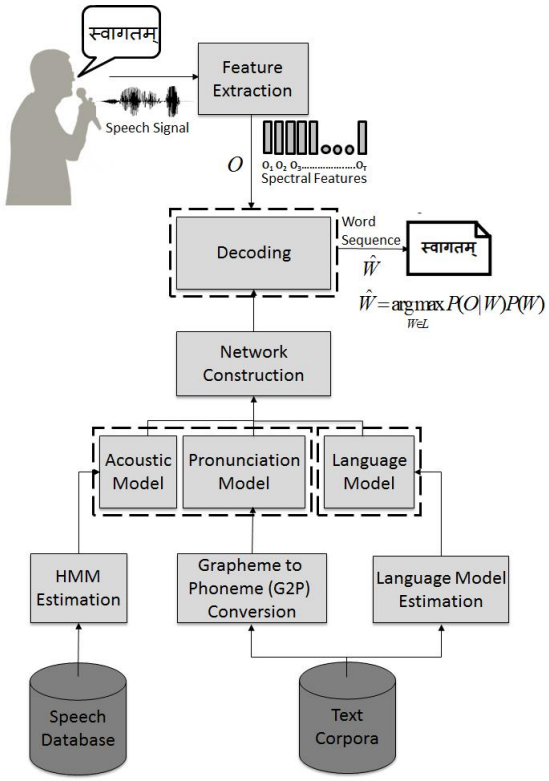

Figure 1. Block diagram of ASR

A. Feature extraction

The first step in the recognition process is to convert the continuous speech signal into a set of equally spaced discrete feature vectors, such that each of them represents the speech waveform for the duration covered by it. This process is called *feature extraction*. The duration is chosen such that the speech waveform can be treated stationary during that time. Fig. 2 shows the block diagram of the feature extraction process.

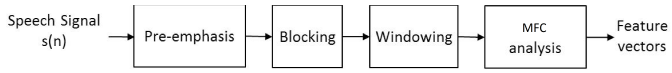

Figure 2. Block diagram of the feature extraction process

1) Pre-emphasis: The amplitude of voiced speech falls off roughly at the rate of -6 dB/ octave at higher frequencies. So the high frequency components have lower magnitude than the low frequency ones. To compensate for this, pre-emphasis is applied to the speech signal prior to the spectral analysis. We use the first order high pass filter for pre-emphasis. The transfer function of the filter is in (9).

$$H(z) = 1 - 0.97z^{-1} \tag{9}$$

2) Blocking: Blocking process divides the entire speech signal into overlapping segments called *frames*. Overlapping improves the correlation between the spectral estimates of successive frames. We use frame widths and frame shifts of 25 ms and 10 ms, respectively.

3) Windowing: Windowing tapers the signal to zero at the beginning and end of the frame, thus minimising the signal discontinuities at both extremities. Our Sanskrit ASR system uses Hamming window, which has the form

$$w(n) = 0.54 - 0.46\ cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \tag{10}$$

4) Filter bank analysis: Mel frequency cepstral coefficients (MFCC) are widely used as features for speech recognition tasks. The computation of MFCCs emulates the processing of speech signal by human ear. Cochlea in the inner ear resolves frequencies nonlinearly across the audio spectrum. This nonlinear frequency resolution is achieved by Mel scale filter banks. They use triangular filters in the frequency domain, equally spaced in Mel scale. Mel scale is defined as:

$$Mel(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \tag{11}$$

MFCC computation emulates the frequency resolving mechanism of the inner ear. It calculates the magnitude spectrum for each frame, thereby identifying the frequencies present in the frame. The magnitude coefficients are weighted by the corresponding triangular filter gain and the accumulated result is a representative of the spectral magnitude in that filter channel. 26 filter channels are used in our ASR.

Next the logarithm of the Mel-scale filter bank parameters is taken. This replicates the nonlinear loudness perception of the human ear. To have a perception of double the volume, it requires almost 8 times the energy. Logarithmic compression makes MFCCs a better representative of the incoming sound.

Finally, the MFCCs are obtained by applying discrete cosine transform (DCT) to the log filter bank parameters $f_j,\ j = 1 \ldots F$, where F is the number of filter bank channels.

$$c_i = \sqrt{\frac{2}{F}} \sum_{j=0}^{F-1} f_j \cos\left(\frac{\pi i}{F}\left(j + 0.5\right)\right),\ i = 0 \ldots F-1 \tag{12}$$

The use of overlapping filters makes the filter bank energies quite correlated with each other. Application of DCT decorrelates the energies. This allows the use of diagonal covariance matrices in the GMMs corresponding to each state of an HMM. Only 12 of the 26 DCT coefficients are retained and the rest are discarded.

A normalized energy term is appended to the final feature set. This is computed as the log of the signal energy.

$$E = log\left(\sum_{n=1}^{N} s_n^2\right) \tag{13}$$

where $s_n, n = 1 \ldots N$ denotes the speech samples. Energy is normalised by subtracting the maximum value of E in the utterance and adding 1 to it.

Dynamic variation of feature vectors over time is captured using delta and acceleration coefficients. Delta coefficients are calculated as:

$$d_t = \frac{\sum_{k=1}^{K} n(c_{t+k} - c_{t-k})}{2\sum_{k=1}^{K} k^2} \tag{14}$$

where d_t and c_t are the delta and static coefficients. Typically $K = 2$. Acceleration coefficients are calculated using the same formula applied to the computed delta coefficients.

We use MFCCs with 0th order cepstral coefficients and delta and acceleration parameters as features. Generally higher order cepstral coefficients are numerically very small. This causes a wide range of variances among the low and high cepstral coefficients. Liftering operation rescales cepstral coefficients so that they have similar magnitudes.

B. Pronunciation modelling

The set of words in the text corpus forms the vocabulary for the ASR. Unlike English, Sanskrit has a one to one correspondence between the orthography and pronunciation of words. So the pronunciation model for each of the word in the vocabulary can be constructed from the corresponding orthography. This process is known as grapheme to phoneme (G2P) conversion. Graphemes are the basic units in the written language. The G2P conversion scheme for our text corpora is explained in section IV-B.

C. Acoustic modelling

Word level acoustic models are built by concatenating the phoneme level models. Each of the phonemes is represented by a left-right HMM. The number of states N in each HMM is set to 5. Out of these, the entry and exit states of the HMM are non-emitting. They act as the

joining points for the phoneme HMMs to create word level HMMs.

A flat start model is used to initialize the phoneme level HMMs. Here we assign the global speech mean and variance to each of the Gaussian distribution in every phoneme HMM. The lack of phone boundary information in the transcriptions of our Sanskrit speech data, makes the flat start initialization our default choice. Embedded training is performed thereafter. Flat start scheme using HTK command `HCompV` makes all the states of all the models equally likely in the first iteration of embedded training and provides uniform segmentation of each of the training utterances. The hope is that enough phoneme HMMs align with their actual realizations so that phoneme models converge to the actual alignment in further iterations.

Once the initial models are created, embedded-training is performed on the entire training data. This performs a single Baum-Welch re-estimation of the whole set of phoneme-HMMs simultaneously. For each of the training sentences, the corresponding phoneme HMMs are concatenated to form a composite HMM. Now forward-backward algorithm is used to accumulate the statistics, such as state occupation counts, etc., for parameter estimation. When all the training data are processed, the accumulated statistics are used to re-estimate the HMM parameters. Single Gaussian phoneme models are later mixture incremented to raise the number of Gaussians to 16.

D. Language modelling

N-gram models are widely used as language models in LVCSR systems. They can be estimated from a sufficiently large text corpus using relative frequency approach. Our work uses bigrams with back-off smoothing [18].

E. Network construction

A network describes the sequence of words that can be recognized. LVCSR systems normally use word-loop. Word-loop has all the word HMMs placed in parallel, with a loop-back. This structure allows any word sequence to be recognized. Networks are represented by a list of nodes and arcs. Nodes represent the end of words and arcs represent the word transitions. Language models (N-grams) can also be incorporated into the finite state networks (FSN) by having some extra network nodes for loop-back transitions.

F. Decoding

Once the FSN is ready, Viterbi search is used to decode the sequence of observation vectors from the unknown speech. HTK uses a slightly modified version of Viterbi algorithm called *token passing algorithm* [10]. It uses logarithmic arithmetic to avoid numerical underflows during probability computations. The steps in the algorithm can be summarized as follows:

1) At time t, each state S_j in the HMM holds a token. The contents of the token are the partial log likelihood $\delta_j(t)$, current time, identity of the previous word and a pointer to a record of history information for the token. This token represents the best partial match between the model and the observation sequence $\mathbf{o_1}, \mathbf{o_2}, \ldots \mathbf{o_t}$, such that the model is in state S_j at time t.
2) At time $t+1$, pass a copy of the token in each state S_i to all the connecting states S_j and increment the value of the copy by $log(a_{ij}) + log\,(b_j(\mathbf{o_t}))$.
3) When the token reaches the end of a word and moves to the start of the next word, update the tokens by incrementing it with log bigram probabilities. Grammar scaling factor s and word insertion penalty p can be incorporated at this point. They modify the language model log probabilities as:

$$\overline{\tilde{P}(w)} = s\tilde{P}(w) + p \tag{15}$$

   Here $\tilde{P}(w)$ represents the log probability evaluated by the language model.
4) At every state, retain only the token with the highest probability.
5) When a token reaches the end of the utterance, the route it followed through the network is traced back with the help of history pointers.

IV. Development of a speech corpus in Sanskrit

For a reliable speech recognition system in any language, HMM models need to be trained with a large amount of speech data containing almost all the phonemes used in the language, in all valid phonetic contexts. So, the first step in building an automatic speech recognition system is to build a phonetically rich corpus that covers almost all possible phonetic contexts in that language. This section describes the Sanskrit speech/text corpus creation process.

A. Speech data collection

Sanskrit speech data and corresponding transcriptions were collected from 3 different sources: All India Radio (AIR) website [19], Indian Heritage Group (IHG), C-DAC [20] and Vyoma Linguistic Labs Foundation. All the three sources had speech data from male speakers, but only the AIR data included speech from female speakers. The collected data amounted to around 3 hour and 50 minutes of speech with 1581 utterances. This included data from 11 male speakers and 4 female speakers. The collected files were divided into different directories based on the news reader.

The audio data from AIR had 2 channels, with a sampling frequency of 16 kHz and was in MP3 format. The data from C-DAC were video files. Vyoma data had single channel, sampled at 44.1 kHz and were in wav format. The audio data was extracted from C-DAC videos using a software called *AoA audio extractor*. The entire data was converted into a uniform format with single channel,

16 bits per sample and 16 kHz sampling frequency. The transcriptions from AIR and C-DAC were in PDF format. They were first converted to JPG image format using an online PDF to JPG converter. The output image files were converted into UTF-8 format Sanskrit text using the Sanskrit OCR tool [21] developed by Dr. Oliver Hellwig of Department for Languages and Cultures of Southern Asia, Freie Universitat, Berlin, Germany. The recognition accuracy of the OCR was reasonably good. The errors in OCR data were corrected manually. The final transcriptions were saved in text format.

The entire speech data was segmented to individual utterances by a team of 4 members at MILE lab. The collected text files were segmented into individual sentences by detecting the end of sentence markers in Sanskrit namely, "।" (danda) and "॥ " (double danda). The audio files and the respective transcriptions were saved with the same file name, but with different extensions.

B. Grapheme to phoneme (G2P) conversion

The training set of sentences must cover most of the phonemes and phonemic contexts in the language. So, selection of sentences requires the phoneme statistics for the sentences. This is achieved using grapheme to phoneme (G2P) conversion. Given the UTF-8 sequence of graphemes, sentences can be mapped directly to the corresponding phoneme sequence. We used the Harvard-Kyoto scheme for transliterating Sanskrit in ASCII.

A G2P converter for Sanskrit was developed in Java. The basic idea was to get the characters in UTF-8 format, in sequence and convert them to the corresponding pronunciation models. Special cases of halant / virama (◌्), nukta (◌़), anusvara (◌ं) and visarga (◌ः) are handled differently.

1) If a *halant* occurs in the unicode, it implies that the preceding phoneme is a pure consonant. So the final schwa sound must not appear in the pronunciation.
2) If *nukta* appears after a character, the pronunciation of the character is changed. If *nukta* follows the characters क, ख, ज, ड, ढ, and फ their pronunciations change to /q/,/x/,/z/,/r/,/r/ and /f/, respectively.
3) *Anusvara* is generally pronounced as the nasal sound of the varga (consonant group based on place of articulation) of the following consonant. Examples are गंगा (gaGga), अंचल (aJcala), अंड (aNDa), अंत (anta), अंबर (ambara).
4) If *anusvara* is followed by a consonant which is not belonging to any varga, it is pronounced as nasal /m/. eg: अंश(amza).
5) Visarga usually appears at the end of the word and must be pronounced in combination with a vowel, same as the one preceding it. eg: प्रवाचकः (pravAcakaha), गुरुः (guruhu), स्थितिः (sthitihi), असुरेः (asurehe).

C. Selection of training and test sets

For phonetically rich sentence selection, phoneme statistics of the collected data is required. For this, MLF files are created with monophone and triphone labels. First a complete list of words appearing in the collected data is created. The pronunciation for these words are created using G2P program. A pronunciation dictionary is prepared with the sorted list of word pronunciations. Master label files (MLF) with phoneme and word level transcriptions are created. Silence labels are added to the beginning and end of each sentence. Unlike TIMIT [22] corpus, the segment boundaries for phonemes and words are not marked. They contain only the phoneme/word sequence and not the segment boundaries. Using the HTK command `HLEd` a master label file (MLF) with triphone labels is created.

Collected speech contained a total of 134054 phoneme instances from 46 phoneme classes. Sentence-wise distribution of these phonemes as well as some other statistics like number of words, number of UTF-8 code points, etc. are generated. Training set of sentences are selected based on the following conditions.

1) Sentence should have number of words between 3 and 20
2) Select a sentence, if it has the maximal occurrence of any of the phonemes.
3) Select sentences having minority phonemes. Minority phonemes are the ones having less than 150 occurrences in the entire data.
4) Select a sentence, if it has the maximal occurrence of any of the triphones.

Out of the remaining sentences all the sentences satisfying, condition 1 are inducted into the test set. There were 1086 sentences in training set and 274 sentences in test set.

D. Associated data creation

Since this corpus is mainly intended for the development of a speech recognition application, the orthographic, phone and word level transcriptions are included for each of the utterance. Master label files are created with phoneme and word level transcriptions. The pronunciation corresponding to the list of vocabulary words is captured in a dictionary file. A back-off bigram language model is generated using the collected Sanskrit corpus. Also a finite state network of words is built using the learnt language model. All the data creation steps are performed using Java.

V. Results and Discussion

For building an ASR system in Sanskrit, we first built a speech corpus of 1360 sentences, 8370 words and 46 phoneme classes. The audio corpus had single channel data with 16 kHz sampling frequency and 16 bits per sample. The phoneme and word level transcriptions for these audio files were prepared. A pronunciation dictionary containing

the collected words and their respective pronunciations is created using the developed G2P converter. Out of the 1360 sentences, 1086 sentences were included in the training set based on the criteria described in section IV-C. Statistics of the collected data are summarized in Table I.

Table I
Statistics of collected Sanskrit speech data

Statistics	Training set	Test set
Male utterances	867	220
Female utterances	219	54
Total utterances	1086	274
Duration - male data	02:01[a]	00:20
Duration - female data	00:34	00:06
Duration - Total	02:35	00:26

[a]Duration in hh:mm (hours: minutes) format.

Performance of the developed LVCSR system for Sanskrit is summarized in Table II. The percentage of correctness and accuracy are calculated as: correctness $C = \frac{H}{N} \times 100\%$ and accuracy $A = \frac{H-I}{N} \times 100\%$, where H, I and N indicates the number of hits, number of insertions and total number of phonemes/words/sentences in the test data. 11267 out of the 15703 phonemes, 1897 out of the 2037 words and 161 out of the 274 sentences in the test set were decoded correctly.

Table II
Performance of the developed LVCSR system for Sanskrit on test set

Acoustic model	H	I	N	A(%)	C(%)
Phoneme	11267	1484	15703	62.3	71.8
Word	1897	71	2037	89.6	93.1
Sentence	161	—	274	—	58.8

VI. Conclusions

A large vocabulary continuous speech recognition (LVCSR) system has been built for Sanskrit language. This is the maiden attempt on a Sanskrit ASR, to our knowledge. A graphical user interface (GUI) for the speech recognizer has also been built using Java Swings. The developed system provides a phoneme level accuracy of 62.3% and a word level accuracy of 89.6% on the test set. 161 out of the 274 sentences in the test set were decoded correctly, yielding a sentence level correctness of 58.8%. A Sanskrit speech corpus with orthographic, word and phoneme level transcriptions has also been built. This corpus has 1360 sentences and 8370 words covering 46 phonemes.

We plan to extend this work to develop a very large vocabulary continuous speech recognizer (VLVCSR) using deep learning framework.

Acknowledgment

I would like to thank Dr. P. Ramanujan, Ex-Associate Director (IHLC), Centre for Development of Advanced Computing and Mr. Venkatasubramanian P., Managing Director, Vyoma Linguistic Labs Foundation for making me available a large volume of transcribed audio data, in different domains, for Sanskrit.

References

[1] Kumar M., Rajput N. and Verma A., "A large-vocabulary continuous speech recognition system for Hindi," IBM Journal of Research and Development, vol. 48, Issue. 5.6, pp.703–715, 2004.

[2] Samudravijaya K., Chourasia K. V. and Chandwani M., "Phonetically rich Hindi sentence corpus for creation of speech database," Proc. of Int. Symp. on Speech Technology and Processing Systems and Oriental COCOSDA-2005, Indonesia, pp.132–137, 2005.

[3] Rao P. V. S., Samudravijaya K. and Agrawal S. S., "Hindi speech database," Proceedings of international conference on spoken language processing ICSLP00, pp. 456–459, 2000.

[4] Mitra P., Banerjee P., Garg G. and Basu A., "Application of triphone clustering in acoustic modeling for continuous speech recognition in Bengali", 19th International Conference on Pattern Recognition, Tampa, FL, pp. 1-4, 2008.

[5] Rabiner L. R., "A tutorial on hidden Markov models and selected applications in speech recognition", Proceedings of IEEE, volume 77, pp. 257-286, 1989.

[6] Rabiner L. R., "Speech recognition in machines", The MIT Encyclopedia of the Cognitive Sciences, MIT Press, Cambridge, USA, 1999.

[7] Lee K. F. and Hon H. W., "Speaker-independent phone recognition using hidden Markov models", IEEE Transactions on Acoustics, Speech and Signal Processing, volume 37, pp. 1641-1648, 1989.

[8] Lachiri Z., Gabzili H. and Ellouze N., "Experimental study of the HMMs effect on the word recognition performance", First International Symposium on Control, Communications and Signal Processing, pp. 615-618, 2004.

[9] Hermansky H., "Perceptual linear predictive analysis", J. Acoust. Soc. Am., Vol. 87, No. 4, pp. 1738-1752, 1990.

[10] Young S., Evermann G., Kershaw D., Moore G., Odell J., Ollason D., Valtchev V. and Woodland P., "The HTK book Version 3.2", Cambridge University Engineering Department, 2002.

[11] Valtchev V., Woodland P. C., Odell J. J. and Young S. J., "Large vocabulary continuous speech recognition using HTK", IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-94, Adelaide, SA, vol.2, pp. II/125-II/128, 1994.

[12] Furstoss M. J., Wax D. A., Larsen N. A. and Kepuska V. Z., "Development of a large vocabulary continuous speech recognition system for rich transcription evaluation using HTK", in Proc. IEEE Automatic Speech Recognition and Understanding Workshop, Waikoloa, Hawaii, 2008.

[13] Woodland P. C. and Cole D. R., "Optimising hidden Markov models using discriminative output distributions", Proceedings ICASSP-91, International Conference on Acoustics, Speech, and Signal Processing, Toronto, Ont., vol.1, pp. 545-548, 1991.

[14] Mohamed A., Dahl G. and Hinton G., "Acoustic Modelling using Deep Belief Networks", IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 1, pp. 14-22, 2012.

[15] Deng L., Hinton G. and Kingsbury B., "New types of deep neural network learning for speech recognition and related applications: an overview", IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, pp. 8599-8603, 2013.

[16] Chen S. F. and Goodman J., "An empirical study of smoothing techniques for language modelling", Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[17] Anoop. C. S.,"Automatic recognition of contemporary speech in Sanskrit", M. E. Thesis, Indian Institute of Science, 2015, unpublished.

[18] Katz S. M. "Estimation of probabilities form sparse data for the language model component of a speech recognizer", IEEE Trans., vol. ASSP-35, no. 3, pp. 400-401, 1987.

[19] All India Radio, http://newsonair.nic.in/language-bulletins-archive.asp, 2015.
[20] C-DAC Technology for Analysis of Rare Knowledge Systems for Harmonious Youth Development (TARKSYA), Indian Heritage Group (IHG), http://bhoomi.csa.iisc.ernet.in:8080/ihg/tarkshya/index.jsp, 2015.
[21] Dr. Hellwig O., Sanskrit OCR, http://www.geschkult.fuberlin.de/e/indologie/mit-arbeiter/drittmittel/hellwig/, 2015.
[22] Fisher W. M., Fiscus J. G., Pallett D. S., Dahlgren N. L., Garofolo J. S. and Lamel L. F., "DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM", 1993.