Appeared in proceedings of NCC, 2014

Better Phone Alignment for Confidence Measures in Voice Based Querying

Nikul Prajapati, Hitesh Tulsiani, Jigar Gada and Preeti Rao
Department of Electrical Engineering,
Indian Institute of Technology Bombay,
Mumbai 400076, India
{nicool, hitesh26, prao}@ee.iitb.ac.in

***Abstract*—Voice-based querying for information is a powerful technology that can hugely enhance the scope of information retrieval systems by enabling their remote access via the ubiquitous mobile phone. Information retrieval based on automatic speech recognition however is challenging due to the environment noise and speaker idiosyncrasies typical of real-world scenarios. Wherever possible, strong domain constraints on the language model are used to minimize the impact of signal degradation on recognizer performance. This can lead to grossly mismatched utterance and hypothesis occasionally, a situation that must be detected to protect the call from irrecoverable errors. We consider the revalidation of recognition hypotheses via confidence measures derived from forced alignment using an independent decoder. We show that our FST based aligner can accurately reject incorrect decoder hypotheses while being particularly robust to the phenomenon of incomplete utterances.**

***Keywords*—Automatic Speech Recognition, Confidence measures, Incomplete utterances, CMU-Sphinx, Open-FST.**

I. INTRODUCTION

The effective exploitation of the vast available digitized information today depends entirely on access technologies. While the web offers efficient search and retrieval with text-based querying, voice-based querying can lead to a manifold increase in reach. Mobile phone penetration is high and growing throughout the country, with a considerably larger presence than internet in rural areas. Speech-based systems involve automatic speech recognition (ASR) at the front-end of the information access system. A well-designed dialog can elicit spoken queries that are decoded by the ASR and presented in the form of text for the search. The retrieved information is converted to speech via a TTS system.

Our work is motivated by an on-going project on providing telephone speech based access to a database that provides up-to-date agricultural commodity prices in markets across Maharashtra. The interactive voice response (IVR) system is equipped with an ASR front-end and a back-end that downloads, on a daily basis, prices of agricultural commodities from a website (http://agmarknet.nic.in/) maintained by the Ministry of Agriculture, Government of India. A simple dialog elicits spoken queries from the user regarding the market and commodity of interest. The ASR is based on an MFCC-HMM framework built using the Sphinx toolkit [1]. The acoustic models and N-gram language model (LM) are trained on specially collected data obtained by field recordings of domain-specific utterances including realistic queries by 1500 native Marathi speakers [2]. The speech database comprises of the names of 32 districts, 279 markets and over 300 unique commodities. A phone-based recognition system is used in order to facilitate the easy inclusion of new commodity names in future. Some mandi and commodity names are multiword. Manual transcription at the word level has been carried out on the entire data making it valuable for research, development and testing of ASR systems.

Challenges in the ASR system design include the speaker diversity due to the distinct dialects of native Marathi speakers across the state as well as the presence of a variety of background acoustic noise in the received speech arising from the caller's environment. Training data that represents multiple dialects helps to reduce speaker dependence. As for the degradation due to noise, a LM that represents the restricted domain vocabulary can help improve the performance. Strong LM constraints however can give rise to incorrect decoder hypotheses when the acoustics are poorly matched to any grammatically valid word sequence such as occurs when SNR is poor or when an out-of-vocabulary word is uttered. Confidence measures are a way of rescoring the decoder hypotheses. Typically new knowledge sources or, at least, new estimates of the same acoustic and linguistic attributes are computed on one or more rank-ordered (N-best) decoder hypotheses. The purpose is to choose the best hypothesis from the available N-best list which comprises close, competing options using an independent method from that used to obtain the N-best list itself. The Confidence Measure (CM), as the name indicates, can also be used as a value indicating the match between the utterance and the decoder hypothesized state sequence. Based on the degree of match, the hypothesis is either accepted or the dialog system prompts the user for a confirmation. In the present work, we investigate the computation of a confidence measure on the decoder output word sequence using an independent aligner built by us.

In our previous work on confidence measures [3], we have shown the effectiveness of a combination of acoustic likelihood and phone duration features in the context of rejecting out-of-vocabulary words. Their computation needs accurate duration estimates and hence depends on phone alignment accuracy. Accurate phone alignment depends on several factors including the acoustic features extracted in the ASR front-end, the acoustic models, and the extent of match of the utterance acoustics with the models' training data.

In order to obtain a modular and flexible framework for experiments on rescoring, we develop an aligner based on FST where separate transducers are constructed for the language model, lexicon and phone context-dependency expansion. These are then combined into a single static network that serves to define the search space in the Viterbi engine. Acoustic models are not part of the network. Typically, such a "static network expansion" leads to a huge search space. Since we are confined to forced alignment with a word-level transcription, the search space is anyhow limited to incorporating simple word loop with beginning-interword-end silences or fillers, and allowing for multiple pronunciations. The "integrated" FST network is input to the Viterbi engine (tree-trellis search) along with the extracted acoustic features and previously trained acoustic models. Several options for experimentation with independently modified sub-modules are now possible. In the next section, we describe the design and implementation of our FST based aligner (affectionately named "Cyborg"). This is followed by a presentation of experiments that evaluate the phone alignment performance.

II. The Cyborg Aligner

Fig. 1 shows a block diagram of our FST based forced aligner. In the overall ASR system, the Cyborg aligner follows the main decoder. Trained acoustic models (AM), dictionary and the utterance transcription, as obtained from the decoder hypothesis, are used to obtain a phone level segmentation of the input audio signal. The individual modules are described next.

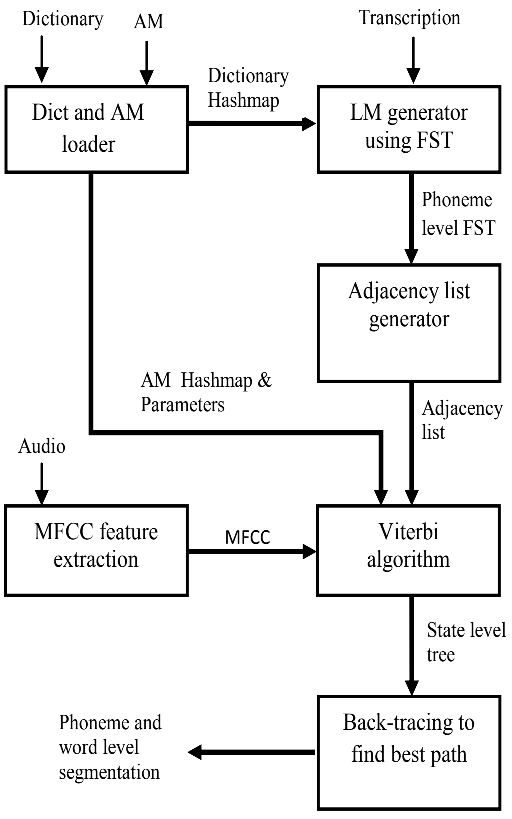

Fig. 1. Block Diagram of Cyborg forced aligner

A. *MFCC feature extraction*

This stage is responsible for processing the raw audio stream sampled at 8KHz sampling rate into a cepstral stream with a 13 MFCC coefficients per 10 ms frame. The CoMIRVA [4] library routine is used with data window duration of 25.625 ms. The feature vector is augmented with delta and double-delta coefficients implemented at the decoding stage.

B. *Dictionary and acoustic models loader*

Acoustic models required for phone alignment are

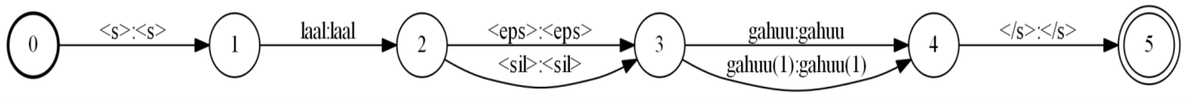

Fig. 2(a). Word Grammar of 'laal gahuu'

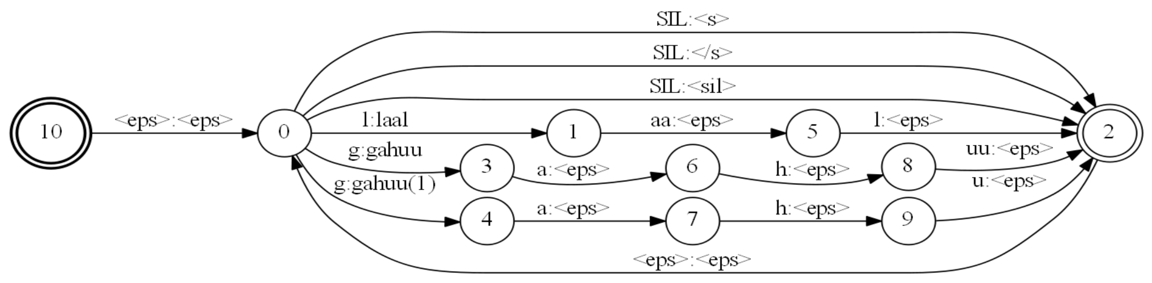

Fig. 2(b). Pronunciation lexicon of 'laal gahuu'

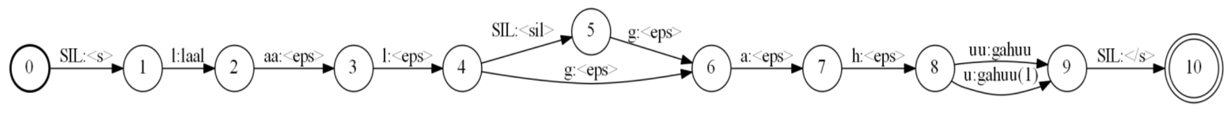

Fig. 2(c). Minimised Phoneme FST of 'laal gahuu'

trained using SphinxTrain [1]. The models are triphone (context dependent) HMMs (Hidden Markov Model) with GMMs for the observation vector distribution. The Gaussians are assumed to have diagonal covariance matrices. The Sphinx binary acoustic model files contain the means, variances, mixture_weights, transition_matrices are converted to Cyborg compatible binaries while model definition file (mdef) and the dictionary, which are in text format, are stored in hash maps for efficient access.

C. Generating FST to deal with alternate pronunciation and fillers

FST are automata in which each transition in addition to its usual input label is augmented with an output label from a possibly new alphabet. FST have been generated in Cyborg using Open FST library [5] for reducing the search space. Search space for a forced aligner is in terms of considering alternate pronunciations and fillers in between the words. By a pre-compilation of the search space, FST is expected to help improve decoding efficiency [6].

Composition, Determinization and Minimization is performed on Word Grammar file and Pronunciation Lexicon file, which are obtained from transcription and dictionary, to generate phoneme level FST. Fig. 2(a) shows Word Grammar with all the possibilities of transcription 'laal gahuu', taking in to account fillers (e.g: babble, horn, silence & other human speech disfluencies) and alternate pronunciation of each word present in 'laal gahuu' and Fig. 2(b) shows Pronunciation Lexicon which represents the mapping of phoneme sequence to words present in transcription considering alternate pronunciation and fillers for 'laal gahuu'. Minimized phoneme FST is shown in Fig. 2(c). In the present example of 'laal gahuu', for simplicity, we have only considered silence filler (SIL).

D. Adjacency list for FST

Adjacency list is an efficient data structure to store and access the neighboring vertices of the minimized FST. Adjacency list helps to obtain next possible

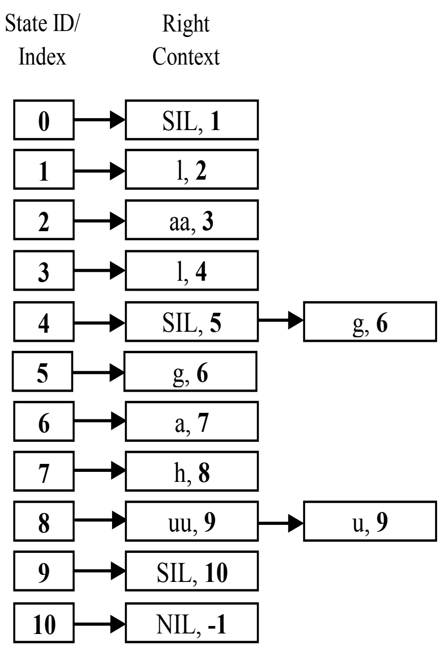

Fig. 3. Adjacency List of 'laal gahuu'

triphone(s) given the current triphone during Viterbi search.

Suppose the current triphone is 'l-aa-l' where the base phone is 'aa' and left and right context are both 'l'. From the Adjacency List shown in Fig. 3, index following the phone 'aa' (base phone) is 3. So we go to index 3 where the phone listed is'l'. So now our base phone becomes 'l' while 'aa' becomes the left context. To find the right context of 'l', we check the index following 'l' which is 4. At index 4 there are two possibilities i.e either 'SIL' or 'g'. Hence the next triphone can be either 'aa-l-SIL' or 'aa-l-g'. Index in the Adjacency List are the State-Ids of FST.

E. Viterbi algorithm implementation using tree

Viterbi algorithm is implemented using a tree data structure to obtain the best possible alignment. Unlike other standard implementations of forced aligner where the last state of transcription is assumed to be the best possible match with the last audio frame, Cyborg aligns the last frame of audio with the state actually having the highest forward probability at that time. This is similar to the work reported in [7].

Let $Q = q_1 q_2 \ldots q_N$ be a set of states corresponding to subphones. The probability to go from state i to state j is represented by a_{ij} and the emission probability i.e the probability of a cepstral feature vector (o_t) being generated from state i is represented by $b_i(o_t)$. The forward probability i.e. probability of being in state j after seeing the first t observations is [8]:

$$\alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

The observation vector is comprised of the 13 MFCCs computed earlier and augmented by delta and acceleration coefficients computed here. Since the classifier used is GMM the emission probability corresponding to observation vector $\bar{x}$ is given by [8]:

$$b(o_t) = \sum_{k=1}^{M} \prod_{i=1}^{D} c_k \frac{1}{\sqrt{2\pi\sigma_{ki}^2}} exp[-0.5 \frac{(x_i - \mu_{ki})^2}{\sigma_{ki}^2}]$$

where M is the number of Gaussians in the mixture and D is the number of elements in the feature vector; μ_{ki} and σ_{ki} refer to the mean variance of ith element of kth mixture respectively and c_k is the weight of each mixture. Since using probabilities can result in numeric underflow, log of both emission probability and forward probability is taken.

During Viterbi search, a simplification is applied to tree generation which consists of deleting the less probable duplicated hypotheses [9]. For example, in the Viterbi tree depicted in Fig. 4, we see that at t3

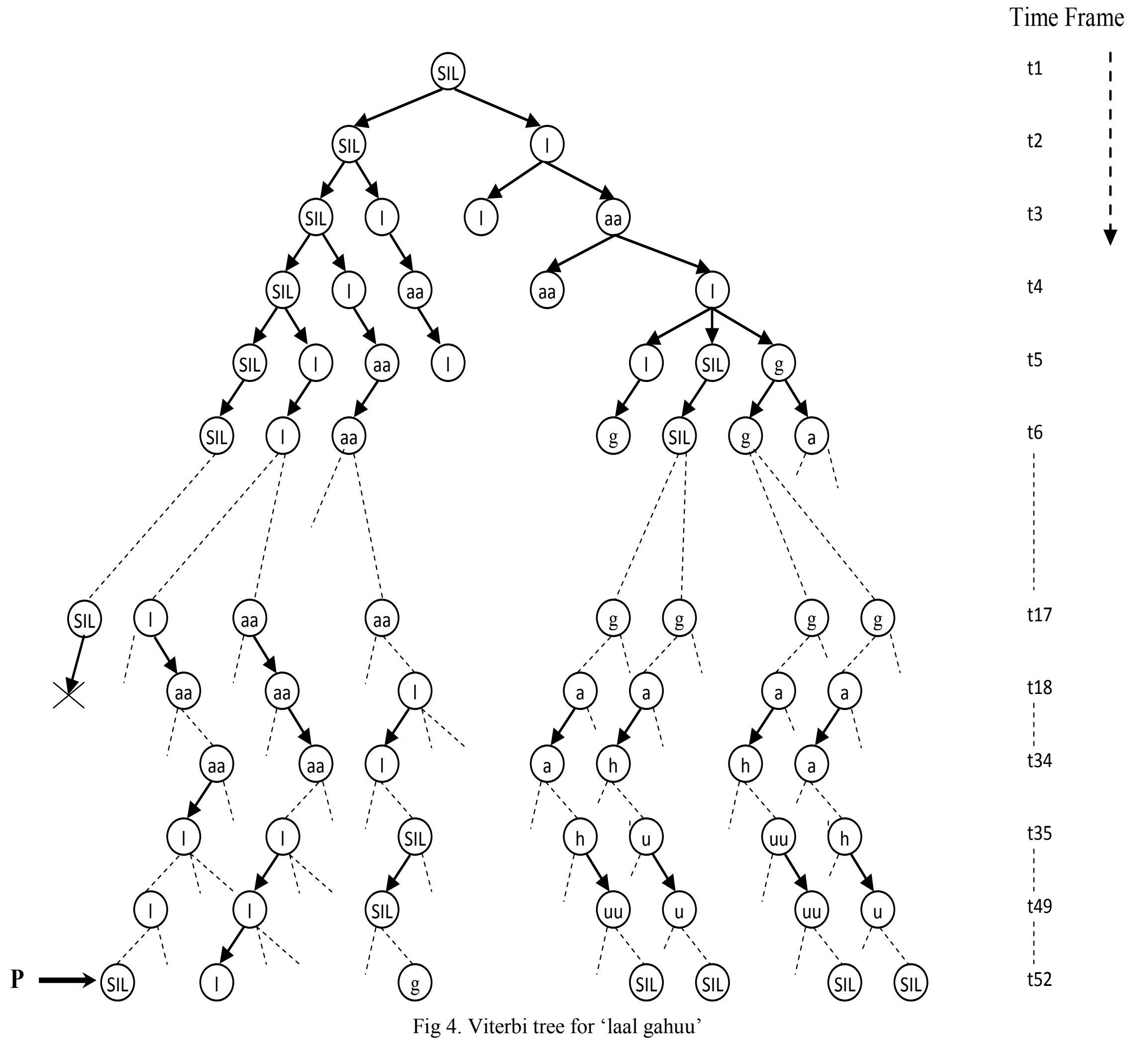

Fig 4. Viterbi tree for 'laal gahuu'

time frame, four states occur but ‘l’ is repeated. Hence the hypothesis having lower forward probability is deleted, which in this case happens to be generated from parent ‘l’. Like pruning (described next) this step simplifies the tree at the cost of possible loss of optimality in the search.

Further, pruning is carried out at each level using beam search to eliminate highly unlikely paths [9]. Log forward probabilities of all the active paths at a particular level are compared and maximum among them is obtained. This maximum log forward probability obtained is then added to log of beam search coefficient and all the paths whose log forward probability is less than beam width (beam width = log of beam search coefficient + Maximum log forward probability at particular level) are pruned. Pruning of one of the path after t17 time frame is shown in Fig. 4 with cross (×).

F. Back-tracing to find best path

The maximum forward probability is obtained and that path is back traced. Consider for example the utterance to be aligned is ‘laal’ but the transcription given is ‘laal gahuu’ having corresponding Viterbi search tree as shown in Fig. 4. As seen from tree, Viterbi search has generated various possible paths that can get aligned with the audio. Among all these paths some of the paths have phoneme sequence corresponding to ‘laal gahuu’, which is same as transcription, but since the utterance is only ‘laal’ the forward probability of that path would be low. On the other hand the path having phoneme sequence of ‘laal’ would have higher forward probability score and hence that path is chosen as the best path for aligning.

If instead the utterance to be aligned is ‘laal gahuu’ then one of the path with phoneme sequence corresponding to ‘laal gahuu’ would have the highest forward probability. In the same way if the utterance to be aligned was ‘laal gahuu deshii’ even then the path having the highest forward probability would be ‘laal gahuu’ but in that case filler would come in to play and ‘deshii’ would get replaced by some filler.

III. EXPERIMENTS

The accuracy of forced alignment can be tested via the performance of confidence measures that depend on phone segmentation. For example, a confidence measure computed from the forced alignment can exploit the fact that words which are incorrectly decoded typically have abnormal phone durations [3]. We present experiments on the recorded call data comprising commodity names where the decoder hypothesized transcription is aligned with the utterance. A phone duration based confidence measure is applied to detect incorrectly decoded words.

A. Performance of duration based confidence measure

Assuming that each phone has a duration distribution that is Gaussian about a mean value that is computed across all occurrences of that phone in the database, the CM was computed for a word as the fraction of phones whose durations fall outside the corresponding middle 75% percentile range of the distribution. A threshold is applied to the CM to identify correctly decoded words. On a dataset of 2000 words, a CM threshold of 0.4 discriminates the wrongly decoded words from the correctly decoded words as shown in Table 1. This indicates that the achieved phone-level forced alignment with Cyborg is reasonable. A more sophisticated duration based CM is expected to improve the results further.

TABLE 1. Classification of correct and wrong words by system

Ground Truth \ System Output	Correct	Wrong
Correct	86.21%	13.79%
Wrong	54.64%	45.36%

B. Handling incomplete utterances

One problem that is observed in the call data is the occurrence of incomplete utterances by speakers due to unexpected appearance of the recording end beep or due to the onset of a loud noise that masks out the latter part of a speaker’s utterance. In such cases, the decoder hypothesis may turn out to contain more words than actually present in the utterance due to the constraints of the N-gram LM. The forced alignment of the utterance with a longer state sequence than actual is problematic with traditional forced alignment generally failing. This is due to the last time frame being forced to align with the final state in the transcription states sequence as observed with the Sphinx forced aligner, for instance. On the other hand, Cyborg is expected to handle this condition gracefully due to its modified back-tracing algorithm as described in Sec. II. Fig. 5 shows an example of alignment of the incomplete utterance “laal” with the transcription corresponding to the commodity name “laal gahuu”. We note that Cyborg correctly identifies the actual words in the utterance while the Sphinx aligner hypothesizes more words than are present. In the former, the best path in the tree of Fig. 4 (marked as **P**) corresponded to the end state of phone SIL whose parent is /l/. Since there are no restrictions on the end state, a word can even end midway.

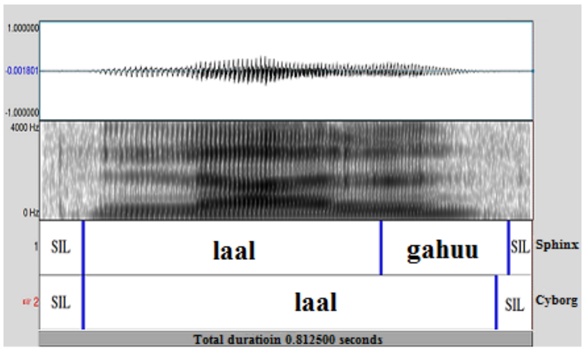

Fig. 5. Alignment using Sphinx and Cyborg aligner for utterance 'laal' (a) Waveform (top) (b) Spectrogram (centre) (c) Word level alignment using Sphinx and Cyborg (bottom)

An experimental demonstration of alignment performance on incomplete utterances appears in Table 2. A data set of about 200 utterances of commodity name was taken. But the transcription provided for aligning had 1 extra word concatenated at the end which was not present in utterance. Aligning was done using 2 different forced aligners: CMU Sphinx and Cyborg forced Aligners.

TABLE 2. Cyborg and Sphinx forced alignment comparison

	Sphinx (no. of words)	Cyborg (no. of words)
Not Aligned (Alignment Failed)	45	0
Wrongly Aligned	155	10
Correctly Aligned	0	190

Table 2 shows that Sphinx did not align 45 utterances because of mismatch between transcription and utterance and it wrongly aligned 155 utterances i.e it aligned the entire transcription with the utterance even though the transcription had an extra word. On the contrary, Cyborg aligned 190 out of 200 utterances correctly i.e it detected that part of the transcription which was actually present in the utterance while it wrongly aligned 10 utterances. Further it was observed that 176 of the 190 words in the aligned utterances satisfied the correctly decoded criterion on the phone duration based CM, indicating that the achieved phone-level alignment was accurate.

IV. Conclusion and Future Work

In this paper, we presented the development of a new FST based aligner 'Cyborg' to revalidate an available ASR decoder hypothesis via a confidence measure. A major goal of this work was to develop an independent system for phone alignment and likelihood computation based on FST. The phone duration based confidence measure did a reasonable job to classify correctly and wrongly decoded hypothesis indicating that a satisfactory phone-level alignment is obtained on real data including incomplete utterances. This work was an example of testing a specific aspect of the back-tracing algorithm. The larger goal is to exploit the available modular implementation of the Cyborg system to experiment with different acoustic features, model combinations, grammar constraints and pruning rules to come up with better rescoring strategies for the decoder hypotheses.

Acknowledgment

This work was supported by the project 'Speech-based access of agricultural commodity prices in six Indian languages' sponsored by Ministry of Communication and Information Technology, Government of India. The authors would like to express their gratitude to Dr. Samudravijaya, Mayur Jagtap, Amogh Garg and Tejas Godambe for their support and assistance in the project work.

References

[1] CMU-Sphinx: Open Source Toolkit for Speech Recognition. http://cmusphinx.sourceforge.net/.

[2] T. Godambe and K. Samudravijaya, "Speech data acquisition for voice based agricultural information retrieval", Proc. Of 39th All India DLA Conference, Punjabi University, Patiala, June 2011. http://speech.tifr.res.in/chief/publ/11DLA_agriSpeechDataAcquisition.pdf.

[3] J. Gada, P. Rao and K. Samudravijaya, "Confidence Measure for Detecting Speech Recognition Errors", Proc. of the National Conference on Communications (NCC), Feb 2013, IIT Delhi, India.

[4] CoMIRVA, Open Source Feature Extraction Library, http://www.cp.jku.at/people/schedl/Research/Development/CoMIRVA/webpage/CoMIRVA.html

[5] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-State Transducer Library", Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007), volume 4783 of Lecture Notes in Computer Science, pages 11-23. Springer, 2007. http://www.openfst.org.

[6] E. Stoimenov & T. Schultz (2009, November). "A multiplatform speech recognition decoder based on weighted finite-state transducers". In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*(pp. 293-298). IEEE.

[7] K.Prahallad and A.W.Black, "Handling Large Audio Files in Audio Books for Building Synthetic Voices", in Proc. of the 7th ISCA Tutorial and Research Workshop on Speech Synthesis(SSW7), Kyoto, Japan, p.148-153, 2010.

[8] Daniel Jurafsky & James H. Martin, *Speech and Language processing,* Upper Saddle River, New Jersey: Pearson, 2009, pp. 285-333.

[9] Claudio Becchetti and Lucio Prina Ricotti, Speech Recognition Theory and C++ Implementation, John Wiley & Sons Ltd, Baffins Lane, Chichester, 1999, pp. 17-18, 315-320.