

Language Independent Speech Compression using Devanagari Phonetics

M. Habibullah Pagarkar, Lakshmi Gopalakrishnan, Nimish Sheth, Rizwana Shaikh, Virag Shah

V.E.S. Institute of Technology

Sindhi Society, Chembur, Mumbai - 400071

mhabibp@ieee.org, laksh75@rediffmail.com, nimishjs@hotmail.com, rizwana111@rediffmail.com,
virag81@gmx.net

Abstract – This paper proposes a method to develop a complete speech compression system using Devanagari script. Any speech can be considered as a concatenation of basic sounds, called 'phonemes'. Most of this basic set is the same for all languages, with a few language-specific additions. If this set can be identified, then we can use these phonemes as the fundamental building block for all speech. Words in any language can then be represented using a Devanagari Transform, which expresses them as a combination of Devanagari phonemes. This lends the process its language-independence. This approach involves developing a phonetic recognizer which segments the words into a set of phoneme units, using Hidden Markov Models. These recognized units are mapped onto a labeled phonetic database containing about 4000 phoneme units. The uttered speech can then be expressed using only the pointers to these entries. This results in compression by a factor of up to 1000. The speech is reconstructed by applying concatenation synthesis for smoothing at segment boundaries. Additional prosodic information is also passed to render the synthetic speech more closer to the real speech. The use of Devanagari, here, has a distinct advantage over other languages since it is phonetic in nature.

I. INTRODUCTION

Even as modern communication methods continue to bring the world closer, there lurks in the background the age-old problem of the Quality vs. Bandwidth tradeoff. It has become more important now than ever before, to make optimum use of the bandwidth available to us. In this paper, we propose an alternative way to compress speech; a method which doesn't involve too much of statistical analysis, a method which borrows from the growing influence of Artificial Intelligence in our day today life. The basic idea here is the use of a phonetic language like Devanagari in speech compression.

The aim of speech compression is to produce a compact representation of speech sounds such that when reconstructed it is perceived to be close to the original. The two main measures of closeness are intelligibility and naturalness. There have been different approaches

towards speech compression where speech is treated as yet another type of signal; redundancies are identified and eliminated, thus effecting compression.

Traditionally, speech signals were sampled at regular time intervals and the amplitude was quantized. Majority of the losses during speech compression occur during the quantization phase. For recorded speech to be understood by humans, an 8 KHz sampling rate or more and at least 8 bit sampling is required. This produces poor quality, but understandable speech. If we group the samples into blocks called frames, we get a better representation for the signal making it possible to analyze the entire frame. Within each frame a vector of features is stored. The final step is vector quantization. Improvements can be achieved by increasing the number of bits in sampling to 12 bits or 16 bits, or by using a non-linear encoding technique such as μ -law or A-law.

However μ -law coding does not exploit the sample to sample correlations found in speech. ADPCM is the next family of speech coding techniques, and does exploit this redundancy by using a simple linear filter to predict the next sample of speech. The resulting prediction error is typically quantized to 4 bits thus giving a bit rate of 32 Kbps.

Our approach, on the other hand, is based on the very nature of speech, where we attempt to establish a representation for speech in terms of subunits. This representation encapsulates knowledge on multiple linguistic levels including morphology, syllabification, stress, phonemics and graphemics. This new paradigm can well be used to more efficiently model the words in a language. The information extracted can be utilized in a variety of different speech applications, thus providing enhanced performance.

At least two immediate applications of this representation exist. First, words can be composed from

the set of these finite units, much like a function can be composed from a basis set. A speech recognizer could operate with these underlying sub-word units, leading to unlimited vocabulary recognition. Second, this theorized “alphabet” could also be used in letter-to-sound/sound-to-letter generation. Once the correct sequence of these units is found for a word, the phonological information could be directly inferred from these units.

Devanagari is the language of choice for the implementation, over the more wide-spread English, for it is an inherently phonetic language and is spoken the way it is written. All human languages use a limited repertoire of about 40 to 50 sounds called phones. Combinations of characters in different contexts produce different phones. It has been found that all the commonly used words in Hindi language can be formed using a set of about 1500 phonemes. We believe that any spoken language can be represented using the basic phonemes in Devanagari along with a few additional phonemes specific to that particular language. For e.g. Australian English is heavily accented and Russian is very guttural. These introduce new phonemes into consideration.

With some language-specific additions, we get a basic set comprising of about 4000 phonemes. Words in any language can then be represented using a Devanagari Transform, which expresses them as a combination of Devanagari phonemes. This lends the process its language-independence.

A fixed dictionary of phonemes is maintained at both the sender and the receiver. Now, any speech belonging to a good number of languages can be represented as a pattern of the phonemes in this database. Thus, a small speech segment, which would conventionally have been represented as a wave file of size about 50 KB can be instead represented as a combination of about 10 phonemes, each of which requires a 12-bit pointer into our database. Even with the inculcation of some additional features such as prosody, the entire file can be represented with just 30 bytes of memory. Using concatenation synthesis, we can now reconstruct the entire sound sequence. This approach thus drastically reduces the storage and bandwidth requirements for the speech file, by factor of over 1000.

Our method for speech compression uses three blocks – segmentation, transmission and synthesis. The first block decomposes the input speech into its component phonemes, using a phonetic recognizer. This recognizer

is designed using Hidden Markov Models (HMM), which is a statistical process model based upon the Markov chain.

II. SPEECH RECOGNITION USING HMM

The recognition process is ultimately a pattern recognition problem. Automatic speech recognition has been recognized as one of the more valuable tools that could be developed as a computer interface. Speech is viewed as a signal that can be modeled as either a deterministic or stochastic processes. By developing different models for each speech segment of interest, incoming speech can then be compared with all available models and the speech segment recognized as the best match. Given the variability of speech by even a single speaker, as well as the numerous speech classes, models of the speech signal must be both flexible and fast. Currently, the most successful speech model is the Hidden Markov Model (HMM), a statistical process model based upon the Markov chain.

A) Language Modeling

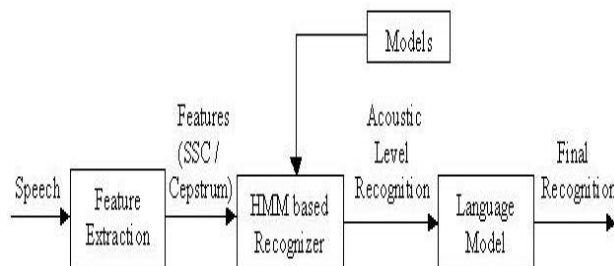


Fig. 1. General Speech Recognition Technique

The acoustic front-end of a good speech recognizer can just predict the sequence of spoken utterances with a probability and the error rate is often to the tune of 20-30% at the acoustic level. This error can be reduced by using a language model that uses the constraints of that language to refine the acoustic-level hypotheses. The acoustic level recognition is less language dependent. It is possible to use a commercially available recognizer and train it with the specific speaker’s voice. The scheme will work up to the acoustic level, but such a recognizer will not be able to utilize the constraints of the Devanagari language and hence the recognition accuracy will be low.

So in speech recognition using Devanagari, language modeling is a core issue. Language model may be based on either:

- 1) Grammatical or Semantics properties using finite state languages
- 2) Statistical characteristics of the word sequences in the given language using n-gram model.

For Indian languages statistical method is a better choice over the usage of Finite state languages. We now focus on the most standard method used for implementation of Hidden Markov Model (HMM). Let $P(W/A)$ be the probability of a given word sequence W , generating acoustic evidence A . Recognition algorithm must identify the word sequence that maximizes $P(W/A)$. According to Baye's theorem,

$$P(W/A) = \frac{P(A/W) * P(W)}{P(A)}$$

where

$P(W)$ is a priori probability that word W is uttered
 $P(A)$ is the average probability that acoustic evidence A is observed
 $P(A/W)$ is the probability that when W is uttered, A is observed.

While maximizing $P(W/A)$ with a given utterance A and $P(A)$ are constant. So our problem is to maximize $P(A/W) * P(W)$. Here $P(A/W)$ corresponds to the acoustic level recognition and $P(W)$ the language model.

$$P(W) = \prod_{i=1}^n P(w_i / w_1, w_2, w_3, \dots, w_{i-1})$$

Here, $P(w_i / w_1, w_2, w_3, \dots, w_{i-1})$ is the probability that w_i will be spoken given that words $w_1, w_2, w_3, \dots, w_{i-1}$ were said before n is the number of words taken in the sequence.

But estimating $P(w_i / w_1, w_2, w_3, \dots, w_{i-1})$ is not feasible even for a moderate vocabulary and a smaller value of i . An approximation adapted to limit calculations is to take the sequences with the same two words at the end as 'equivalent' and calculate probability:

$$P(W) = \prod_{i=1}^n P(w_i / w_{i-2}, w_{i-1})$$

This is the 'Trigram Model'. They aren't as sophisticated as PCFG but they manage to capture local syntax.

B) Recognition Algorithm

The standard method used is HMM, since it is easy to implement, well supported, properly documented and runs fast.

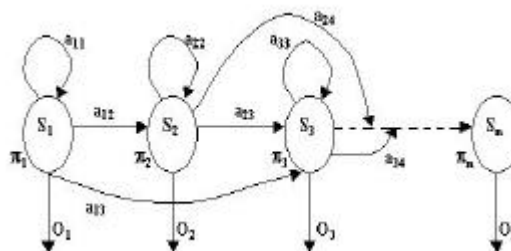


Fig. 2 – Hidden Markov Model

In HMM, each spoken unit of recognition, which maybe a word or a sub-word, is represented by a model: with several states and a sequence of transitions through these states. The model is represented by:

1. the initial transition probabilities of i^{th} states (π_i),
2. transition probabilities from state i to j (a_{ij}) and
3. observation sequences from state i (O_i).

Each state represents a spoken sub-unit. The state sequence to be traversed is probabilistic for any given unit, as it may be pronounced differently within permissible limits. We cannot see any state directly, rather observe them through features. Mapping from state to features too is probabilistic. The model is therefore doubly stochastic which states 'hidden'. By HMM, $P(A/W)$ i.e. the probability of a particular model generating the given observations, is found for each model and the product $P(W) * P(A/W)$ is to be maximized. The associated problems are:

- 1) Training i.e. evaluation of model parameters given the training data. This is an offline task. There is no analytical solution. But reasonably good training is possible with maximum likelihood and iterative procedures e.g. Baum Welch method.

2) Computation of Probability of a particular model producing the observed sequence. This corresponds to recognition and is Online. Here an analytical method exists. Direct probability computation leading to astronomical number of operations is not feasible. But methods to solve this problem neatly and analytically exist. Forward – Backward Procedure is often used or an objectively determined best state sequence may be found by Viterbi algorithm

The Viterbi algorithm takes an HMM and outputs an output sequence and returns the most probable path through the HMM that outputs the sequence. It also returns the probability of the path. It can be thought of to be an iterative algorithm that first finds all the paths that output the first symbol (phoneme in this case). For each of those paths it finds the most probable path that outputs the rest of the sequence, given that we have chosen a particular path for the first symbol. If the length of the sequence is n and there are M states, then it seems as if this algorithm will be $O(M^n)$.

The key point in the Viterbi algorithm is to use the Markov property to make it more efficient. The most probable path for the rest of any sequence depends on the state in which it starts. This means that all the paths need not be looked into. We need to keep track of the most probable path that ends in that state. It is an example of dynamic programming.

The HMM is used in speech recognition for 2 reasons – firstly, the Viterbi algorithm is linear with respect to the length of the input. Secondly, HMMs can be made to learn directly from a training set.

III. SPEECH SYNTHESIS

Speech synthesis programs convert written input to spoken output by automatically generating synthetic speech. Speech synthesis is often referred to a "Text-to-Speech" conversion (TTS).

Both concatenative and formant synthesis techniques can be used. A text processor first accepts text as input and outputs phonemes and prosody markers. Next, speech is generated by concatenating diphone-like segments of recorded natural speech in the former and by a production model, based on 'formant' (resonance) frequencies in the latter (fig 3). A major problem of concatenative method is discontinuity at segment

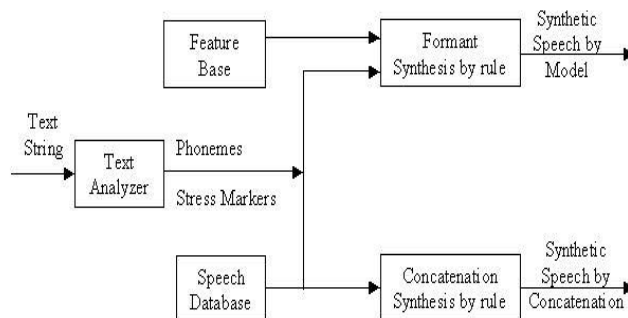


Fig. 3. Formant & Concatenation Synthesis

boundaries. In formant synthesis it is difficult to capture the stop consonant to vowel transitions by rules.

Our major concern is to map the received phoneme units into the labelled database with utmost accuracy. Once this is accomplished, concatenative methods would be fairly sufficient in most of the cases. However, it might be necessary to employ formant synthesis for certain word parts to get increased accuracy.

A) Segment Inventory and Selection

In concatenation synthesis, it is very important to have a well-selected and efficient inventory. The inventory being used by us is a minimal inventory, which is limited to the entries in the labeled phoneme database. Here, we have entries in which the phoneme is either preceded by or followed by a vowel. The inventory must also contain certain typical polyphone units for multiple consonant clusters.

Ideally, there should be a large speech database with enough redundancy, so that out of alternatives, a segment, most suitable in a given context, can be selected. Else, discontinuities at the segment boundaries will show up. However, there are certain trade-offs to be considered.

Manual segmentation of a large database needs considerable skilled man-hours and may delay the effort. Also, there should be some objective criteria to select the 'most suitable' segment, at a given context. Automatic labeling using HMM is an alternative. HMMs can also be used to automatically select optimal setoff diphone and polyphone units to be used, simply by checking for the spectral stability at the segment boundaries.

However, the job of segmentation can be simplified by keeping the size of the database (segment inventory) to a least possible value. As stated earlier, the price to be paid for simplicity is discontinuity at the segment boundaries. In the next section, we discuss a solution to this problem.

B) Smoothing at segment boundary

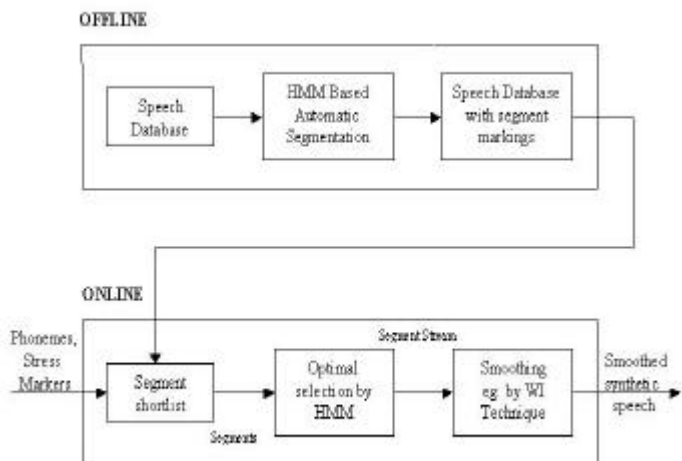


Fig. 4. Typical Concatenation Synthesis Scheme

Discontinuity at segment boundaries is a problem in concatenation synthesis, especially with modest segment inventory. Even with careful speaking, any two boundaries may not spectrally match, giving rise to a sudden feeling of discontinuity. This can be reduced by smoothing. Waveform interpolation (WI) is used for speech coding. The spectral parameters of speech change rather slowly over pitch cycles. It is thus possible to code speech by retaining frames at some intervals and reconstructing intermediate frames by interpolated spectral parameters.

As interpolation parameter, LP residuals are often used. The same method can be used for smoothing. Here a few frames at the boundary (1-3 pitch cycles on each side) are removed and reconstructed by WI. This ensures smooth transition between segments. Transitions between natural and coded speech creates a little buzzing at the boundary. But overall, the quality improves.

IV. PROSODY & FIDELITY CRITERIA

A possible problem with this approach is that since the reproduced speech consists of pre-recorded components,

it may not sound like the sender's voice. In order to overcome this problem, we make use of prosodic information, which is also transmitted as a part of the data structure for each phoneme unit.

Prosody has three elements: pitch, intensity and duration. This is an integral part of any speech synthesizer. To render the synthesized speech to be as natural as possible, prosodic criteria have to be noted and used during synthesis. Knowledge about the pitch helps to bring the voice quality closer to that of the sender's. The volume information tells us how loud the speech is at every instant. The duration of each phoneme allows us to capture the mannerisms of individual speakers such as emphasis on a particular sound or speed of talking, etc. Use of this information makes the reproduced synthetic speech remarkably similar to that of the sender's. Generally, various combinations of prosody and inventory are used for synthesizing speech of near-human quality.

Modulating duration and intensity is easy in both concatenation and formant syntheses. But in concatenation synthesis with time domain approach, pitch change is tricky. An easy way is to 'stretch' or 'compress' each pitch cycle in inverse ratio of pitch. The core problem of generating modulation contours for prosody elements, given the context, calls for prosody rules: to be formulated at sentence, phrase, word, syllable and phoneme levels. One possible scheme is to generate 'markers' for each prosody element (during text processing), preferably decided by easily detectable criterion. (eg., we can 'stress' the 'k' words 'kyaa', 'kaun' etc. in interrogative sentences and adjectives in exclamatory ones).

To provide language independence to the system, it is necessary that the prosody is not dependent on the labeled inventory. Thus, prosody is created during recognition and is kept intact throughout the system to achieve faithful reproduction during synthesis. This gives enhanced usability and functionality to the system – that of recognizing moods such as anger, happiness, sadness and neutral. More details of this implementation are given in [6].

For a state-of-art speech synthesis system, the comparison is with human speech. Hence, 3 features can be used to compare synthetic speech and human speech.

- 1) Intelligibility
- 2) Naturalness

3) Variability

In terms of intelligibility, i.e.: how understandable speech synthesis is to humans, recent research has shown that synthetic speech can reach the intelligibility levels of human speech. However, lack of variability, i.e.: changes in speech rate and voice quality, and naturalness, i.e.: how ‘human’ a synthesized speech sounds, has impeded the general acceptance of synthetic speech, especially for extended listening.

Since our major goal is to achieve compression, the only criterion we deal with seriously is intelligibility. Synthesized speech is of no use if the naive user cannot interpret it correctly. We have already agreed upon sacrificing the continuity at the segment boundaries due to the limited size of our inventory. Consequently, the testing routine of our synthesis module will be based upon subjective evaluation by a large number of naive users.

V. IMPLEMENTATION

The implementation of this project needs to be carried out in a sequence of three stages, namely, recognition (segmentation), transmission and synthesis. Compression can be achieved by passing pointers across between the source and the destination. But the two major challenges that have to be tackled include Recognition and Synthesis. The block diagram for the process is as follows:

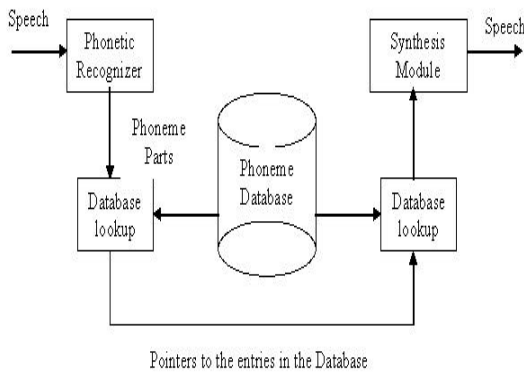


Fig. 5. – Speech Compression System Block Diagram

Speech recognition relevant to our system would primarily recognize phoneme-parts which form the basic elements of our system (entries in the database). We have already documented the implementation of the phoneme recognizer using Hidden Markov Models. The

output of this recognizer is an index, which represents the phoneme recognized.

Certain additions are made to the index to include prosodic information about the phoneme, and this data structure is transmitted for each phoneme. The file can also be stored in this format for reproduction at a later stage. The transmitted data structure then appears as follows:

Pointer to the Phoneme	Prosodic Information	Checksum
12	15	9

Fig. 6. Data Structure to be transmitted

At the receiver end, when the speech is required to be played back, the received data structures are used to get the indices of the phoneme units. A database look-up is then performed to find the actual phoneme. There is a pre-recorded sound file corresponding to each file in the phoneme database. The set of all these component phonemes is then concatenated using formant and concatenation synthesis methods to get the speech back at the receiver end. To incorporate fluency in the reproduced sound, we make use of waveform interpolation techniques.

Using the Database, we have been working on the development of synthesis system employing the principles of Concatenation Synthesis. This has led us to conclude that our system ought to be lossy. This is not a major concern as the main emphasis is on Compression rather than faithful reproduction of data. Conceptually, the reason behind the losses is the nature of our database, which doesn't have the provision to include the pitch, formants and stress, which are different for every individual.

The overall success of the system depends upon the development of an extensive knowledge base for the application and proper co-ordination of both the recognition and synthesis modules.

VI. APPLICATION AREAS

This project in an advanced form can have many far-reaching effects. Some of the promising applications are outlined below:

Telephones can be equipped with a processing entity and some storage for the phonemes. The incoming voice

stream can be digitized and converted using the technique outlined. The telephone can then output a stream of phonemes. The advantage here is that each voice channel can now occupy lesser bandwidth. This can be employed in cellular phones, which already have a powerful central processor. This can be an effective solution to the world's bandwidth crunch.

Since in the early implementations, exact fidelity cannot be obtained, another application comes to the fore: Voice activated devices. The ultimate aim of technology is to make human life more comfortable. Gadgets can be fitted with a rudimentary device that will allow them to understand what the user is saying. Not having to do a lot of statistical analysis means that we can use simpler and thus cheaper circuitry.

Text-to-speech synthesis can greatly enhance operations based on human-machine interaction and make them simpler and more appealing. This may also be used in applications such as a reader for a blind/illiterate person, etc. Other applications include reading out bank statements on a telephone, by an automated teller, automatic translation system between multilingual persons, etc. These and other such applications render our novel approach to speech compression a great deal of significance in the present scenario.

VII. REFERENCES

- [1] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, February 1989.
- [2] Aniruddha Sen, "Text to Speech Conversion in Indian English", *Proc. KBCS 2000*, pp.564-575, Mumbai, Dec 2000.
- [3] M. R. Schroeder, "Speech Processing."
- [4] Aniruddha Sen, "Speech Recognition and Synthesis in India: A few R & D Issues", Tata Institute of Fundamental Research.
- [5] Aarati Parmar, "A Semi-Automatic System for the Syllabification and Stress Assignment of Large Lexicons."
- [6] Murtaza Bulut, Shrikanth S. Narayanan, "Expressive Speech Synthesis Using a Concatenative Synthesizer", AT&T Labs-Research.
- [7] Karlheinz Stober et al, "Synthesis by Word Concatenation."
- [8] Romain Prudon & Christophe d'Alessandro, "A Selection/Concatenation text-to-speech synthesis system: Databases development, System Design, Comparative Evaluation."